

Realsoft 3D

Table of Contents

Readme for Realsoft3D Version 8 viii
1. Readme 1
New Global Illumination System 2
Distributed Post Processing 3
Modeling Tools 3
Progressive Image Rendering 4
HDRI Support 5
Improvements to Particle System 5
View Navigation 6
User Interface 6
Simulation 7
Import/Export 8
User Interface Improvements 9
User Manual xv
1. Introduction 1
Getting Started 1
Installing and Starting the Program 1
Default User Interface 2
Building Your First Scene 5
Modeling Your First Building 16
Building Walls 16
Editing the Walls 18
Windows and Doors 21
Decorating Rooms 23
The Roof 24
Furniture 25
Landscape 27
Project Harry 29
Introduction 29
Modeling the body 29
Adding Materials 34
Programming the Skin with VSL 35
Local UV Texture 42
Hair 44
Skeleton 48
Animating the Character 52
Rendering the Animation 57
2. Modeling 60
Using Curve, Circle and Rectangle Tools 60
Using Object Handles 66
Working with Multiple Projects 71
Snapping to Points Using Dragging 73
Using the Numeric Interface for Accurate Modeling 76
Grids 78
Boolean Operations 79
Advanced Boolean Operations 83
Lattice Mapping 85
Recording and Executing Macros 90
Particles 93
Creating Leaves Using 3D Particles 99
Camera 105
Fonts 109
Rotator Object 109
Tube Object 111

Instances 112
Constraints 113
Field Evaluator 114
Metaball Object 115
Link Object 120
3. Subdivision Surfaces 122
Subdivision Surface Modeling 122
Using Compass Menus in SDS Modeling 126
Using Point, Edge and Face Handles 128
Pointwise Creation of Arbitrary SDS Shapes 131
Modeling a Bird 133
Converting Objects to Subdivision Form 135
Defining Pointwise Properties 138
Creating Symmetric Objects with the Duplicate Tool 139
Adding New Points and Faces 142
Rational Subdivision Surfaces 144
Creases 146
Rounding 148
Assigning Materials to Faces 150
Using the Knife tools 153
4. NURBS Curves and Surfaces 155
NURBS Curves 155
NURBS Surfaces 164
Rotational Surfaces 170
Sweep and Bi-Rail Tools 173
Cross Section Surfaces 175
Extruded Surfaces 177
Trimming NURBS Surfaces 178
Welding 182
Fillets 185
The Font Tool 187
Vectorizing Curves from Images 188
Creating Curves from Surfaces 189
The Displace Tool 191
5. Modeling Buildings 192
Creating Parametric Objects 192
Modeling Roofs 204
Multiple Floors 207
Designing Kitchens 210
6. Materials 213
Introduction 213
Blending Materials 217
Texture Mapping 220
Material Classes 223
3D Painting 225
Visual Shading Language 233
Introduction to Visual Shading Language 233
VSL Wizards 238
Building an Interface for a VSL Material 238
Bump Mapping 239
Fog Examples 242
Illumination Examples 244
Miscellaneous Examples 247
Type Casting 251
Parameters 251
Assigning Multiple Materials to SDS Objects 252

Modeling a Pot 256
Modeling the shape 256
Texturing our pot 257
Scope mapping 260
Clip Mapping 262
Previewing Materials 264
Combining 2D and 3D 266
Adding Dirt 272
Adding Dirt 272
Dirty Wall 275
Color Control 276
7. Rendering 280
Introduction 280
Rendering Settings 282
Rendering to a File 282
Distributed Rendering 285
Lighting 288
Volumetric Rendering 293
Motion Blur 299
Field Rendering 300
Scanline Hairs 301
Alpha Channel 305
Advanced Illumination Examples 306
Outline Rendering 314
Caustics 317
Animation with Global Illumination 320
photon map Illumination 328
8. Post Effects 331
Post Image Effects 331
Post Particle Effects 334
Blur effect 337
Particle Smoke 338
Compositing 340
Animated Lens Flare 345
Post Shading 346
Technical Outline Drawing Style 350
9. Animations 353
Getting Started with Animation 353
Animation Controls 356
Choreography Basics 358
Choreography Essentials 364
Path Animations 368
Animating Materials 374
Animating Particle Effects 378
Animating Image Effects 378
Rotations 379
Texture Morphing 381
Rendering Animations 383
Skeletons 384
Choreography Parametrization 391
Batch Rendering 396
10. Simulations 397
Simulations Overview 397
Controlling Velocity and Spin 400
The Creator Object 401
Explosions 404

Gravity 406
Collision Detection 409
Electro Magnetism 413
Fluid Dynamics 413
Scripting in Simulations 417
11. User Interface 420
Customizing the Toolbar 420
Customizing the User Interface 421
Binding Macros to Keys 423
Binding Macros to Pull Down Menus 428
Binding Macros to Buttons 428
Binding Macros to Compass Menus 430
Localization 431
12. Scripting 433
Introduction to Scripting Languages 433
Controlling Geometric Objects 434
Binding Scripts to GUI Objects 434
JavaScript 434
Introduction 435
Creating User Interface Objects 435
Managing Geometric Objects 443
Vectors and Matrices 447
Managing Materials 451
Layers 453
Implementing New Classes 455
13. Miscellaneous 460
Hot Keys 460
Productivity Tips 467
Troubleshooting 469
14. Imprint 473
Index 474
Reference Manual ..... cdlxxviii
1. Realsoft 3D 1
Models 1
Channels 1
Choreography Classes 2
File Rendering Configurations 5
Geometric Objects 9
Grids 87
Images 87
Materials 88
Output Objects 120
Post Particle Effects 125
Post Image Effects 128
Rendering Settings 138
View Models 146
Animation Object 146
Startup options 147
Scripting 149
Built-in JavaScript Classes 149
Tools 176
3D painting Tools 176
Analytic Tools 181
Boolean Tools 188
Building Construction Tools 189
Construction Tools 192

Creation Tools 204
Particle Tools 207
Deformation Tools 210
Foot Print Tools 216
Geometry Manager Tools 217
View Control Tools 218
Lattice Mapping Tools 222
Light Source Creation Tools 226
Material Mapping Tools 228
Miscellaneous Tools 232
NURBS Curves and Surfaces 237
Plant Tools 255
Subdivision Surface Tools 256
Simulation Tools 273
Transformation Tools 277
Triset Tools 281
Unified Creation Tools 282
User Interface 291
Geometry Managers 291
Macro GUI gadgets 292
Windows 292
Windows 296
Advanced Selection Window 296
Animation Settings 297
Animation Window 298
Attributes Window 300
Choreography Window 300
Color Window 304
Curve Editor 305
File Rendering Window 308
Info Window 308
Material Browse Window 308
Numeric Window 309
Object Browse Window 310
Options Window 310
Property Window 312
Scripting Window 313
Select Window 313
Status Window 327
Tool Bar 327
Tool Control Bar 327
UV Editor 328
UV Editor Properties 330
View 330
View Control Bar 338
View Properties 341
2. BatchRen 350
Description 350
Startup Options 350
Examples 351
3. BatchRenW 352
Description 352
Using BatchRenW 352
4. RendD 353
Description 353
Startup Switches 353

Examples 355
5. RendW 357
Description 357
Pull-down Menus 357
Startup Options 357
The Control Window 358
6. ShowImg 360
Pull-down Menus 360
Startup Parameters 360
7. Software Updater 362
Description 362
Startup options 363
8. StartOpts 364
Description 364
Startup options 364
9. RsView 366
Description 366
Pull-down Menus 366
Startup Parameters 367
10. Plugins 368
Index 372
Chrono ccclxxxi
1. Chrono 1
Dynatomic ii
1. Dynatomic 1
Revolution ii
1. Revolution 1
Legal Notices ii
1. Imprint 1
2. End User Licenses 2
Realsoft 3D SOFTWARE LICENSE AGREEMENT 2
DynAtomic Software License 2
Chrono End User License 3
3. Third party software 4
ImageMagick License 4
GD Graphics Library 6
LIBPNG LICENSE 7
FFmpeg Licence 9
FreeType Licence 9
Open CASCADE Technology 11
SpiderMonkey JavaScript Engine 11
BugTrap License 12
Program Icons 12
CURL LICENSE 12
Index 13



Readme for Realsoft3D Version 8

Readme for Realsoft3D Version 8

Table of Contents

1. Readme 1
New Global Illumination System 2
Distributed Post Processing 3
Modeling Tools 3
Progressive Image Rendering 4
HDRI Support 5
Improvements to Particle System 5
View Navigation 6
User Interface 6
Simulation 7
Import/Export 8
User Interface Improvements 9



Chapter 1. Readme

Welcome to Realsoft 3D Version 8!

This readme file describes improvements and bug fixes in Realsoft 3D version 8. If this is the first time you use the software, the tutorials of this User Manual help you to get started. Remember to register at realsoft.com! It qualifies you for technical support, upgrade announcements, special customer discounts and an access to free software updates. After installing the program, you can send us your contact details easily using the pull down menu Help/ Realsoft in the Web/Register.

Thank you for choosing Realsoft 3D!

Version 8.2.56 (documentation updated 2019-12-23 00:36:11)

New Global Illumination System

Powerful global illumination solution based on photon level simulation added. The tool features new innovations such as baking of prefiltered illumination for extra fast animation renders.



Courtesy of Pixel Perfect

Some examples of the options:

- Baking tool prefilters photon illumination. This speeds up rendering dramatically.
- Direct and indirect photons can be captured selectively.
- Unforeseen photon supersampling feature makes it possible to store high quality, low noise illumination into a relatively small number of photons.
- Automatic photon sampling range and brightness estimation at map creation.
- Intensity control for caustics like photon effects.
- Photon capturing can be turned on/off for each individual object in the scene.

Distributed Post Processing

Post processing render engine now automatically distributes effects applied to fully composed images. Time consuming effects such as glow now render 2-8 times faster, depending on the number of cores in the CPU.



Modeling Tools

Rotate & Extend Tool

New transformation tool 'Rotate & Extend' from Real3D™ Version 3 plugged in. The tool allows you to both rotate and scale object in one go.



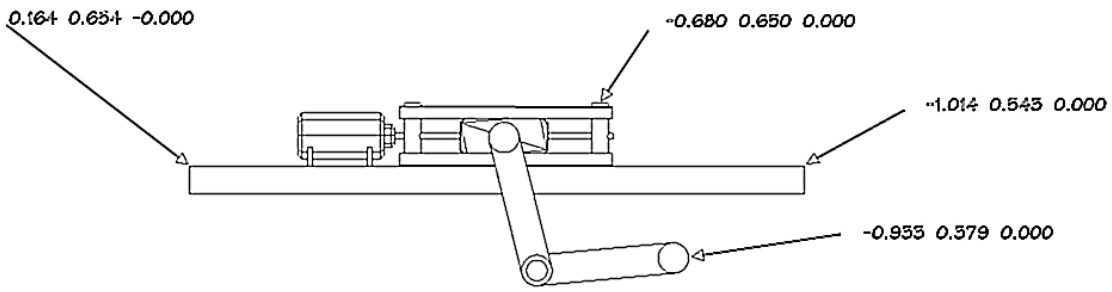


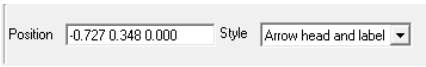
Move-to Transformation Tool

`Move-to` transformation tool added. Click a point and all the selected objects are moved into the specified position.

Position Mark Measurement Tools

Position Mark measurement tool lets the user to displace the label and the dot visualizing the actual position to be measured. To achieve this switch to edit mode, and drag the position handle with the mouse to displace it from the label. Furthermore, there is a 'Style' option to show only the label, both the label and the position dot, or in displaced case render arrow head line from label to the position mark.

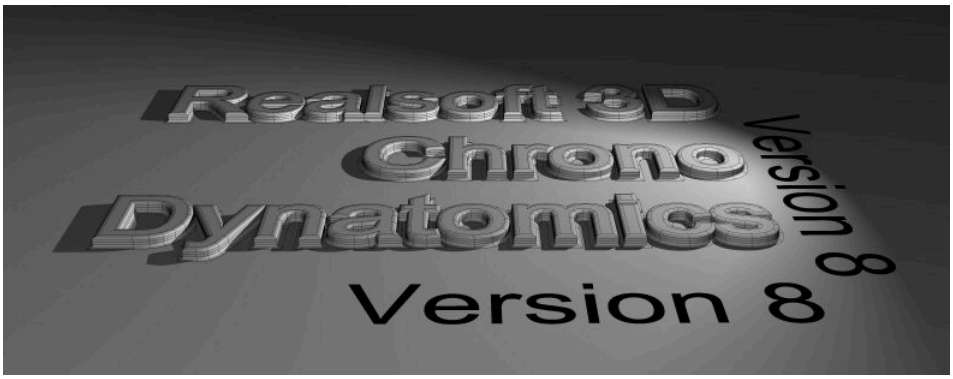






Nurbs Extrude Tool

NURBS Extrude tool optimized for objects with zero extrusion depth. When the depth is set to zero the tool creates only trimmed surfaces.



Editing Numeric Input Fields with Mouse Wheel

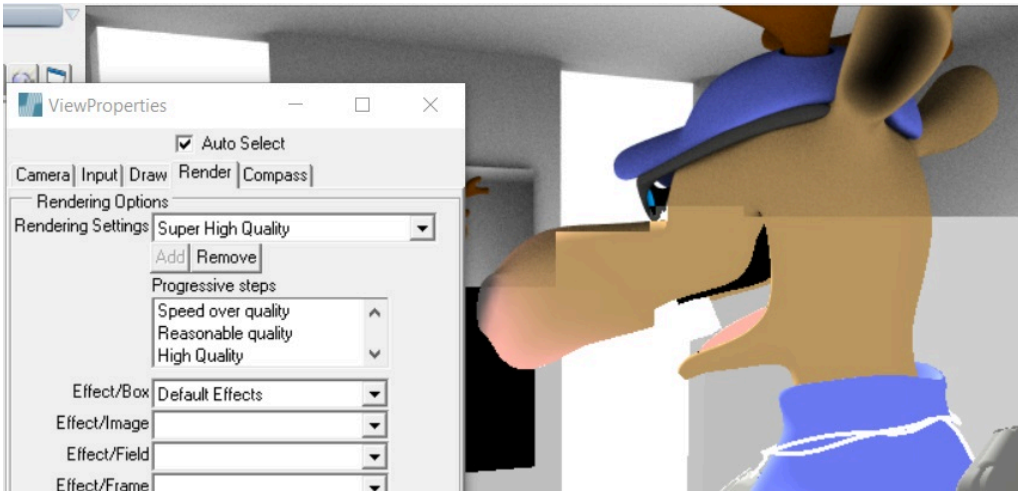
Mouse wheel support added for numeric input fields. The wheel changes the digit under the mouse. This provides a fast and productive way for editing the values without the keyboard.

Progressive Image Rendering

The user can now define a sequence of render settings for view windows, for example:

1. draft quality rendering (typically 1 seconds/view)
2. medium quality rendering (typically 10/seconds view)
3. good quality rendering (typically 1 minute/view)

View rendering automatically steps through the defined render quality sequence.



HDRI Support

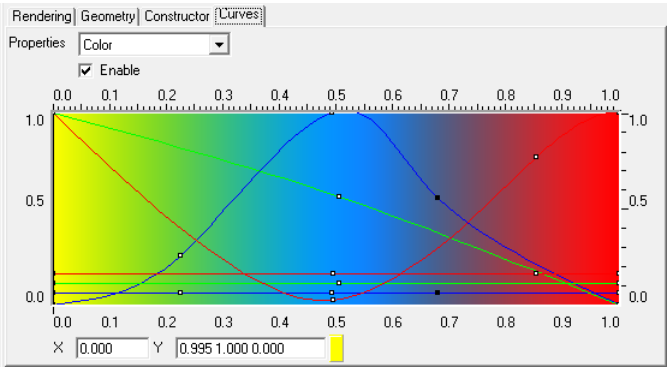
Support for HDRI plugged in. Version 8 should be able to read all common HDRI formats with full dynamic range. HDRI images can be rendered to Realsoft 3D Native image format and OpenEXR files.

Improvements to Particle System

Curve Control over Particle Properties

Particle system supports 'Curves' for easy control over pointwise properties.

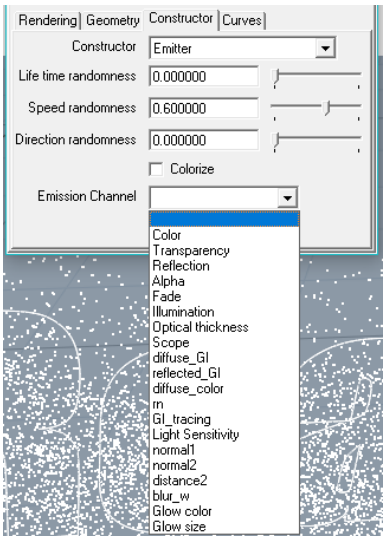
For example, make 'Color' a pointwise property and then enable curves for the color. All curves use particle's relative age (0.0 - 1.0) as input to define colors for particles. For example, the older the particle the dimmer its color.



Emission Channel

Particles support new attribute `Emission Channel`. Any attribute, such as color or transparency, can be used for determining now much particles gets emitted from objects.

For example, setting `Emission Channel Color` means that black areas of a surface won't emit particles at all where as surface areas with white color emit with maximum rate.



Solid Model Awareness

Particle `Emitter`, `Distribute over surface` and `Filler` features support solid models. Particles are emitted only from areas which are not trimmed out with boolean operations.

Position Randomness

This option controls how much randomness is added over the position of the new particles to be created. If zero then particles gets emitted precisely from the surface of the emitter object.

View Navigation

The mouse wheel now zooms the view about the mouse pointer. This reduces the need to pan after zoom.

Left and Right Shift modifiers control now different things. Left Shift + dragging with the Right Mouse Button zooms in / out. Right Shift + Right Mouse Button controls the distance (moves the camera closer to or away from the current aimpoint).

Basic 3DConnexion™ 3D-mouse support added for view navigation control. The View Control group in the Environment/Customization/Available Objects window contains four new tools for enabling panning, zooming and rotation effects on View windows.



Note that this toolset is loaded and available only when a 3D mouse device is plugged in.

User Interface

DPI Awareness for High Resolution Screens

Realsoft 3D V8 is DPI aware: it scales user interface elements to match the detected monitor resolution. Scaling applies to tool button icons, 3D handles and respective thresholds used for detecting when mouse is placed over the handle.

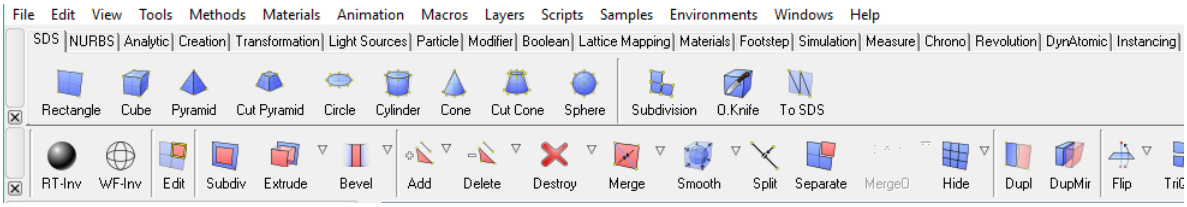
The default operating system detected scaling factor can be overridden with a new 'dpi' parameter. For example, to force DPI to 128, add `-dpi 128` to the Realsoft 3D launching shortcut.

```
c:\Program Files\Realsoft\Realsoft 3D_64_V8.exe -dpi 128
```

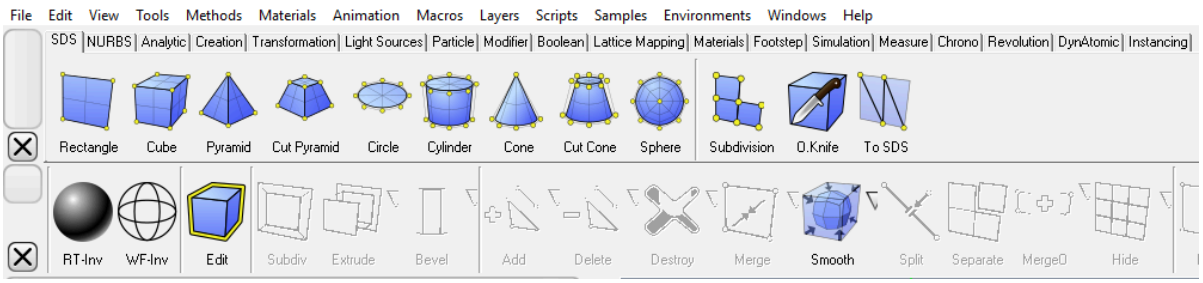
If the underlying operating system does not support DPI awareness 96 dots per inc is assumed.

SVG Icons

Version 8 includes a new icon set based on SVG (Scalable Vector Graphics) format. SVG icons can be freely scaled to any size without loss of quality.



Courtesy of Andy Jones, Shooting Pixels Photography



Courtesy of Andy Jones, Shooting Pixels Photography



Note

SVG is the default icon set for Windows operating system. Linux version relies on ImageMagick-7 package which is not available for all Linux distros yet. The linux version of Realsoft 3D therefore uses PNG based `pxl8` icons by default.

Desired icon set can be selected by passing the following startup parameter to the program:

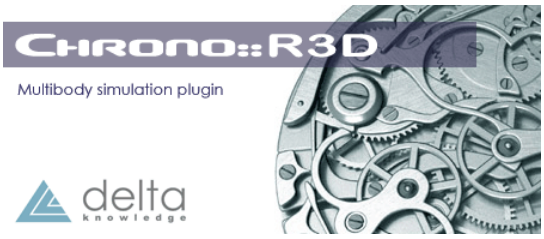
```
-icon ICONSET
```

where ICONSET can be either pxl8 or svg.

Simulation

Chrono Multi-body Dynamics

Chrono™ - the powerful multibody dynamics tool set is included with Realsoft 3D V8



Chrono integration has been improved. All Chrono objects now maintain initial state. There is a new tool 'Set Initial' for Chrono body object.

Dynatomic™ Toolset

Dynatomic™, an advanced dynamic simulation system for soft-semirigid interacting models, clothes and garments integrated with Realsoft 3D.





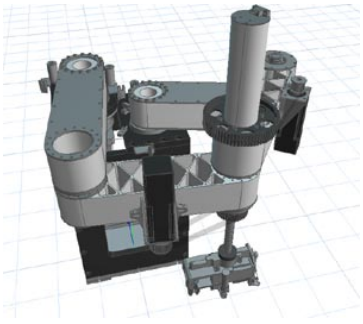
Note

Dynatomic™ is not included with the Linux version

Import/Export

Revolution4D Toolset

Revolution 4D plugin for importing and exporting various CAD formats, such as STEP and STL, included.



Import/Export Formats

FBX format introduces several bug fixes and a few improvements. It now supports native NURBS curves and can automatically convert analytic objects to NURBS meshes for export. Please note that this feature is still under development and does not support trim curves, materials nor animations yet.

Improvements to Wavefront OBJ format

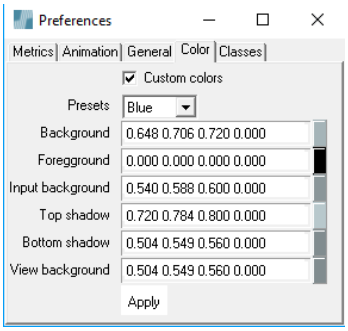
- Memory leak when exporting analytic objects fixed.
- Random crashes when exporting nurbs curves fixed.

Support for exporting STL format geometry for 3D printing.

User Interface Improvements

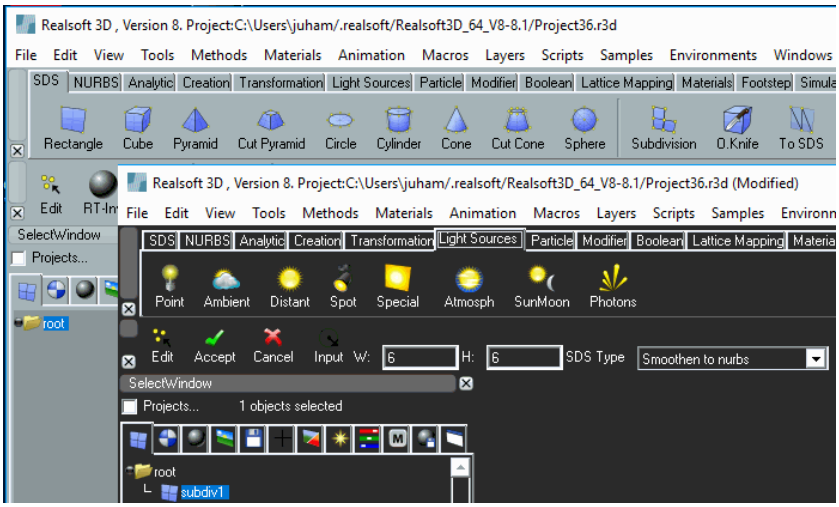
Many tools support now context sensitive options more comprehensively for maximally fluent work flow.

GUI colors can now be controlled in the Preferences window.



The colors can also be overriden using startup arguments:

-vbg 0xrrggbb	View background color
-bg 0xrrggbb	Background color for other windows
-fg 0xrrggbb	Foreground color (color for text)
-bg 0xrrggbb	Bottom shadow color
-ts 0xrrggbb	Top shadow color
-ibg 0xrrggbb	Input background color
-scheme name	for selecting predefined color sets. The parameter value 'name' can be black, dark, light, white, blue, green.



The default view window background color can also be overridden for a specific view by setting 'Custom background color' checkbox in the View Property Window's Draw tab.

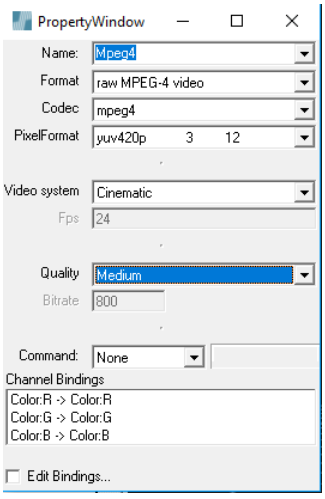


Ffmpeg

Ffmpeg plugin introduces several improvements.

Pixel-format control added.

Predefined quality and Video System options provide a more intuitive user interface.

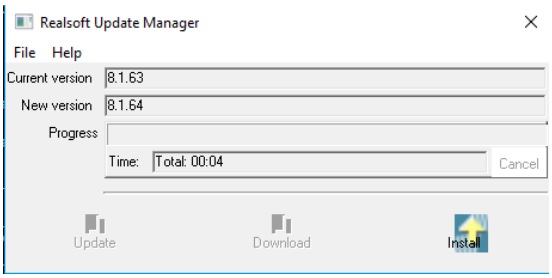


Ffmpeg libraries upgraded to the most recent versions available.



Software Updater

Realsoft 3D contains software updater which can detect, download and install new Realsoft 3D versions.



The software updater is configured for manual operation by default. It can be activated by selecting `Help/Realsoft on the Web/Check for updates` pull-down menu. You can also ask the operating system to start it automatically when booting up, in which case the software updater process runs on the background and uses task bar icon to notify you when new updates become available.

For more information consult the User Manual



Context Sensitive Control Bar

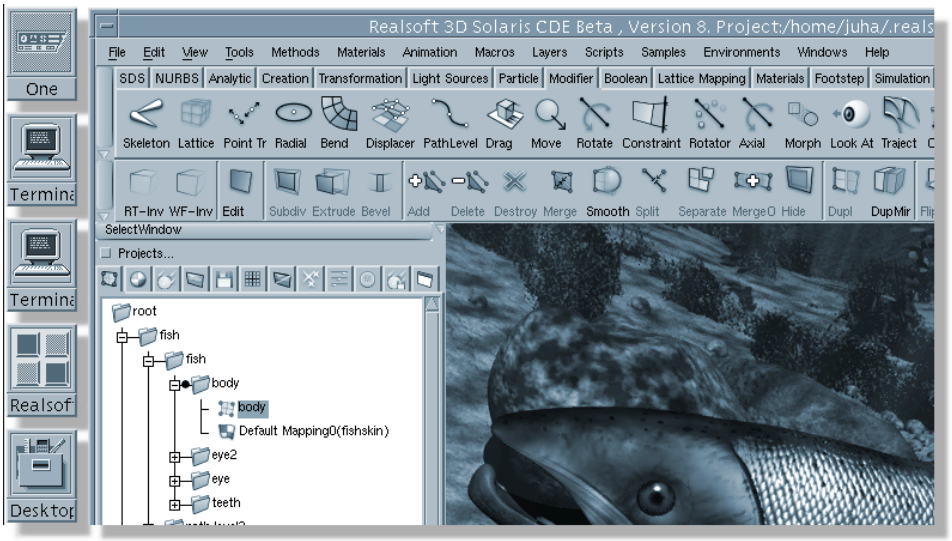
Context sensitive Control Bar has been extended to provide the user with frequently needed tools and options for most commonly needed geometric objects.

For example, select an 'extrude' object and the toolbar automatically shows options extrusion depth, beveling radius and other extusion specific controls.



Support for Solaris 10

Beta version for Solaris 10 operating systems is available.



Realsoft 3D on Solaris 10 CDE



Miscellaneous

- Known Windows 10 compatibility issues fixed.
- Rendering no longer crashes on systems with more than 32 CPU cores. Maximum number of CPUs/cores elevated from 32 to 256.
- Broken group object fixed.
- Raytrace VSL object's randomness control works correctly and intuitively when increasing randommness towards value 1.
- Backsurface culling optimization disabled when tracing rays through faded surfaces. The result is visually easier to understand.
- Point and spot light with a non-zero emitting radius now emit light from a spherical surface, not from a volume. The approach gives faster shadow computation with a smoother quality.
- V8 should run on most relatively modern Linux systems and distros. Currently Debian package is available.
- Support for ImageMagick libraries plugged in (allows reading of many new file formats, including SVG).
- File 'Save As' and 'Load' user interfaces use Combobox to clearly state the purpose of the field.

- Rendering of imported dense polygon models optimized to allow significantly higher model complexity.
- Raytrace the Position mark label text if 'Show name' option in the Wire tab of the Property Window is set.
- New matrix decomposition system plugged in for more compatible import/export.
- File/Export converts all transformation matrices to orthogonal right handed form.
- Several performance improvements througout the software.
- New option added to the Normalize tool: 'Preserve object space handedness'.
- 'Normalize' tool did not correctly eliminate skew for mirrored object spaces, fixed.
- Skeleton/Native pose was broken for animated skeletons. Fixed.
- Option to enable/disable automatic Nurbs extrude construction added. It helps to avoid continuous and slow reconstruction of extruded shapes in some real-time interactions.
- Memory corruption with 3d emitters containing zero particles fixed.
- Random crash bug with the fractal tree generator fixed.
- Rendering to .PNG format crashed. Fixed.
- Compass menus sometimes rendered with corrupted (typically black) background, fixed.
- Button consumed the Shift key, breaking Shift Down/Up detection code and compass menus. Fixed.
- Deleted tasks did not always remove themselves from the task instance list, leaving the task instance list into corrupted state. Fixed
- 'UV image' tool was broken in the 64 bit version of Realsoft 3D. Fixed.
- Buffer overflow bug in trashing GUI colors in exit fixed.
- Incomplete configuration when Double clicking .r3d project fixed (the project worked if loaded via File/Load).
- Potential crash bug during exit: macro controls were destroyed while still attached to models. Fixed
- 'UV View' / reset function did not support macro recording, fixed.
- Random buffer overflows with key-repeat Ctrl & Alt keys fixed.
- Shortcut 'p' no longer opens multiple property windows.
- Vertex painter is now auto-accepted after one drag event, for more fluent work flow.
- In GUI Edit mode, splitting a view operation used current directory for saving tmp files, resulting into 'Cannot access' error in Windows 7 or newer. Fixed
- Lattice mapping related selection bug fixed.
- Basic geometric tools included with the context sensitive toolbar.
- Scale related floating point precision issue resulted incorrect indirect lightning effects with imported scenes. Fixed.
- Morphing tool created incorrect results when applied to camera objects, fixed. Morphing angle attributes uses now shortest route interpolation.
- Polar angles were incorrectly recorded by the macro recording system. Fixed.

- OpenEXR (a high dynamic-range image file format developed by Industrial Light & Magic) upgraded.
- Linux: fixed Motif crash bug due to destroyed Motif label.
- Memory leak in icon class fixed.
- Memory leak and small performance issue with view navigation fixed.
- View backdrop image feature did not work correctly in the linux version. Fixed for both OpenGL and Xlib canvases.



User Manual

Seventh Edition

User Manual

Publication date 1985 ... 2019
Copyright © 1985 ... 2019 Realsoft Graphics Oy

All rights reserved. This documentation contains proprietary confidential information and trade secrets of Realsoft Graphics Oy.

Realsoft 3D™ and Real3D™ are trademarks of Realsoft Graphics Oy and Realsoft Oy. All the other trademarks are the property of their respective owners.

THIS SOFTWARE AND RELATED MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality, performance and result of the program is with you. Should the program prove defective, you assume the entire cost of all necessary servicing, repair or correction. IN NO EVENT SHALL Realsoft Graphics Oy BE LIABLE FOR ANY DAMAGES CAUSED USE OF THIS MATERIAL.

Table of Contents

1. Introduction 1
    - Getting Started 1
        - Installing and Starting the Program 1
        - Default User Interface 2
    - Building Your First Scene 5
    - Modeling Your First Building 16
        - Building Walls 16
        - Editing the Walls 18
        - Windows and Doors 21
        - Decorating Rooms 23
        - The Roof 24
        - Furniture 25
        - Landscape 27
    - Project Harry 29
        - Introduction 29
        - Modeling the body 29
        - Adding Materials 34
        - Programming the Skin with VSL 35
        - Local UV Texture 42
        - Hair 44
        - Skeleton 48
        - Animating the Character 52
        - Rendering the Animation 57
2. Modeling 60
    - Using Curve, Circle and Rectangle Tools 60
    - Using Object Handles 66
    - Working with Multiple Projects 71
    - Snapping to Points Using Dragging 73
    - Using the Numeric Interface for Accurate Modeling 76
    - Grids 78
    - Boolean Operations 79
    - Advanced Boolean Operations 83
    - Lattice Mapping 85
    - Recording and Executing Macros 90
    - Particles 93
    - Creating Leaves Using 3D Particles 99
    - Camera 105
    - Fonts 109
    - Rotator Object 109
    - Tube Object 111
    - Instances 112
    - Constraints 113
    - Field Evaluator 114
    - Metaball Object 115
    - Link Object 120
3. Subdivision Surfaces 122
    - Subdivision Surface Modeling 122
    - Using Compass Menus in SDS Modeling 126
    - Using Point, Edge and Face Handles 128
    - Pointwise Creation of Arbitrary SDS Shapes 131
    - Modeling a Bird 133
    - Converting Objects to Subdivision Form 135
    - Defining Pointwise Properties 138
    - Creating Symmetric Objects with the Duplicate Tool 139

Adding New Points and Faces 142
Rational Subdivision Surfaces 144
Creases 146
Rounding 148
Assigning Materials to Faces 150
Using the Knife tools 153
4. NURBS Curves and Surfaces 155
NURBS Curves 155
NURBS Surfaces 164
Rotational Surfaces 170
Sweep and Bi-Rail Tools 173
Cross Section Surfaces 175
Extruded Surfaces 177
Trimming NURBS Surfaces 178
Welding 182
Fillets 185
The Font Tool 187
Vectorizing Curves from Images 188
Creating Curves from Surfaces 189
The Displace Tool 191
5. Modeling Buildings 192
Creating Parametric Objects 192
Modeling Roofs 204
Multiple Floors 207
Designing Kitchens 210
6. Materials 213
Introduction 213
Blending Materials 217
Texture Mapping 220
Material Classes 223
3D Painting 225
Visual Shading Language 233
Introduction to Visual Shading Language 233
VSL Wizards 238
Building an Interface for a VSL Material 238
Bump Mapping 239
Fog Examples 242
Illumination Examples 244
Miscellaneous Examples 247
Type Casting 251
Parameters 251
Assigning Multiple Materials to SDS Objects 252
Modeling a Pot 256
Modeling the shape 256
Texturing our pot 257
Scope mapping 260
Clip Mapping 262
Previewing Materials 264
Combining 2D and 3D 266
Adding Dirt 272
Adding Dirt 272
Dirty Wall 275
Color Control 276
7. Rendering 280
Introduction 280
Rendering Settings 282

Rendering to a File 282
Distributed Rendering 285
Lighting 288
Volumetric Rendering 293
Motion Blur 299
Field Rendering 300
Scanline Hairs 301
Alpha Channel 305
Advanced Illumination Examples 306
Outline Rendering 314
Caustics 317
Animation with Global Illumination 320
photon map Illumination 328
8. Post Effects 331
Post Image Effects 331
Post Particle Effects 334
Blur effect 337
Particle Smoke 338
Compositing 340
Animated Lens Flare 345
Post Shading 346
Technical Outline Drawing Style 350
9. Animations 353
Getting Started with Animation 353
Animation Controls 356
Choreography Basics 358
Choreography Essentials 364
Path Animations 368
Animating Materials 374
Animating Particle Effects 378
Animating Image Effects 378
Rotations 379
Texture Morphing 381
Rendering Animations 383
Skeletons 384
Choreography Parametrization 391
Batch Rendering 396
10. Simulations 397
Simulations Overview 397
Controlling Velocity and Spin 400
The Creator Object 401
Explosions 404
Gravity 406
Collision Detection 409
Electro Magnetism 413
Fluid Dynamics 413
Scripting in Simulations 417
11. User Interface 420
Customizing the Toolbar 420
Customizing the User Interface 421
Binding Macros to Keys 423
Binding Macros to Pull Down Menus 428
Binding Macros to Buttons 428
Binding Macros to Compass Menus 430
Localization 431
12. Scripting 433

Introduction to Scripting Languages 433
Controlling Geometric Objects 434
Binding Scripts to GUI Objects 434
JavaScript 434
Introduction 435
Creating User Interface Objects 435
Managing Geometric Objects 443
Vectors and Matrices 447
Managing Materials 451
Layers 453
Implementing New Classes 455
13. Miscellaneous 460
Hot Keys 460
Productivity Tips 467
Troubleshooting 469
14. Imprint 473
Index 474



Chapter 1. Introduction

Getting Started

Installing and Starting the Program

Realsoft 3D ships with Debian and RPM packages. Just download the appropriate setup package and follow the standard installation procedure to install them.



Note
Realsoft 3D requires Motif package.

If your Linux distro is not either Debian or RPM based, then you can use the generic package, which can be run on any system with typical *nix* programs, such as tar, gzip and bash, installed.

To install the program using the generic .sh setup, enter the following commands in a terminal window:

```
su
sh ./realsoft3d-8.2.*-linux-i386.sh
exit
```

or

```
sudo sh realsoft3d-8.2.*-linux-i386.sh
```

If you have TCL installed a graphical installer is invoked. If you don't have TCL package installed then the setup runs in text mode. You may also force the install to run in text mode, by passing -t parameter to the setup:

```
sh ./realsoft3d-8.2.*-linux-i386.sh -t
```

If you have TCL installed a graphical installer is invoked. If you don't have TCL package installed then the setup runs in text mode. You may also force the install to run in text mode, by passing -t parameter to the setup:

```
sh ./realsoft3d-8.2.*-linux-i386.sh -t
```

Realsoft 3D can be started from Applications/Realsoft Graphics Oy/Realsoft 3D menu entry, or by clicking the Realsoft 3D desktop icon. You can also start it the same way as any standard Linux appication, by typing its name 'realsoft3d' in a terminal window.

If the program refuses to start, you most likely don't have the Motif libraries installed (libXm.so), or your need newer versions of them installed. Try starting the program in a terminal window to see the exact name of the missing library, and then install the package containing the library version in question.

After installation, you need to set the install key. Start the program and select the pull down menu Help/About. The opened About window includes an Install Key field. Type your install key into it exactly as written in the sticker of the install CD or in the license registration message, and then click the Set Key button. Note that characters are case sensitive. You can also find the install key after the download link in the Software Updates area of realsoft.com.



Note

The install key consists of your personal license key, and a platform specific extension. It should look as follows:

```
annnnnnnnnnnn.nnnnnnn
```

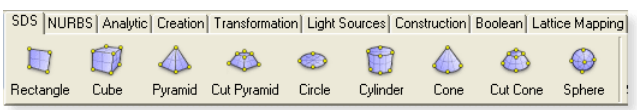
If you do not set the install key, the program will expire after 2 months. It will then run in demo mode, save functionality disabled.

If you are a new Realsoft 3D user, remember to register your license at realsoft.com! It qualifies you for technical support, upgrade announcements, special customer discounts and an access to free software updates.

Default User Interface

When you start the program, the default user interface is loaded from the startup project. The default interface consists of the following elements:

Toolbar

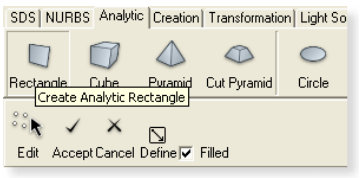


Tabbed Toolbar

The Toolbar allows you to activate such things as the creation and modification tools, as well as many others. The default configuration shows a 'tabbed' toolbar, with tool icons grouped under several tabs like SDS, Creation, Light Sources, and so on. You can hide/show the toolbar by clicking the header area at the top left corner of the toolbar.

Control Bar

The Control Bar is located right below the Toolbar. The control bar is context sensitive. When you select a tool (the Rectangle tool, for instance), the control bar contains the set of modifiers that are relevant to that tool. If you activate the Rectangle tool, valid rectangle-specific options are shown.



Rectangle tool selected. The control bar automatically shows the tool-specific controls

The Control Bar always shows options that are related to the action the user is currently performing. Therefore, its contents change frequently, and adapt to the workflow.

The Control Bar contain more tools than may fit into your screen. However, you can scroll the toolbar with the mouse wheel or by dragging it with the right mouse button while holding down the Ctrl key.

View Control Bar

The View Control Bar is a tall and narrow vertical window, located near the right edge of the main window. It allows you to control the view orientation, and other options specific to View Windows.



For example, the View Control Bar contains buttons for switching between Front/Side/Top views, activating different rendering methods, and rotating, zooming, and panning a view.



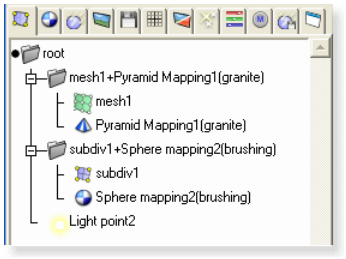
Note

Clicking the Front, Side and Top button while holding the **Shift** down will switch the view into the opposite direction.

The View Control Bar can be scrolled in the same way as the Toolbar if all the controls do not fit into the screen area.

Select Window

The Select Window is located near the left edge of the main window. It allows you to select objects based on their names. When working with large projects, this window is extremely useful for selecting all types of objects, from geometric ones to light sources and material maps. Through its various tabs, the Select window also allows you to manage such things as materials, rendering settings, post effects, and so on.



The Select Window

The Select Window allows you to manage the object hierarchy using a drag&drop interface. For example, you can drag one object inside another 'level' object. A drag box can be used for multi-selecting objects. You can also multi-select objects by holding down the **Shift** key while clicking objects.

If there are more objects than fit into the window, you can scroll the contents of the select window in the same way you can scroll the toolbar: just hold down the **Ctrl** key while dragging with the right mouse button.

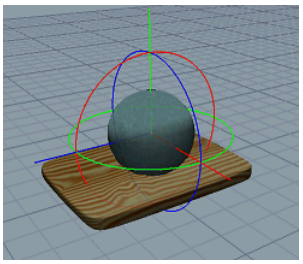
View Window

The View Window is perhaps one of the most often needed windows. It renders geometric 3D objects using various rendering methods, such as wire frame, shaded OpenGL, or ray tracing. You can enter 3D data through the View Window using mouse or some other input device, such as a digital tablet.

To Pan a view window (scroll it): **Ctrl** + drag right mouse.

Zoom a view window: **Shift** + drag right mouse vertically

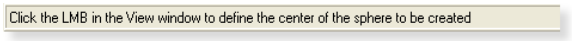
To Rotate a view window: drag middle mouse button.



View in Shaded OpenGL mode

Status Window

The Status Window is where the program prints out messages and other feedback.



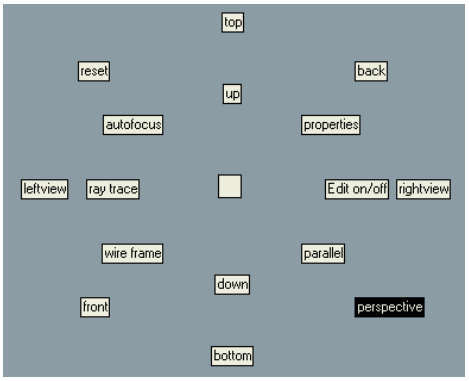
Status window

In addition to these, there are several other windows included with the standard startup project. You can open any of these windows through the View pull down menu.

Compass Menus

View window also support special purpose menus called Compass Menus. They are context sensitive. The contents of the compass are usually set to include those functions that are most likely needed in each situation. For example, if a curve object is selected, the compass menu includes the most frequently needed curve editing functions.

You access a Compass Menu by dragging the right mouse button in the view window. Where you drag, the mouse determines which compass menu is displayed. When you have moved the mouse more than a certain threshold distance, the compass menu is opened.



Drag with the right mouse button to select far south-east compass menu

For example, to enable the perspective view projection, drag while pressing the right mouse button in the View Window. The compass menu is displayed. Keep moving the mouse until the perspective label is highlighted and release the mouse button.

The big idea behind the compass menus is that your muscles can learn to use them. Selecting functions in this manner can become very fast after some practice, because it is not necessary to watch the menu while using it

(unlike when using traditional menus, or buttons where you have to accurately position the mouse pointer over the desired item). Your hand remembers the necessary motions to pick an item.

Compass menus also can be bound to keys. For example, the View window compass menu (shown in the previous illustration) is bound to **v** key. Holding down the **v** key while dragging with the right mouse button overrides the default context sensitivity rules, allowing you to access the view specific options whenever you need.

Compass menus are an extremely powerful way to access tools in Realsoft 3D and we strongly suggest you learn to use them.

Building Your First Scene

This first tutorial project introduces you the basic workflow for building and rendering a simple animation in Realsoft 3D. It is intended for first time users including users familiar with previous Realsoft products, because it introduces several features not available in previous program versions.

In this tutorial, you will do the following:

- Model a wooden table with a marbled sphere on it
- Add a light source
- Create a camera
- Animate the sphere to move along the table
- Render the animation to file.

Tutorial level: Beginner

Modeling

To prepare the View Window for modeling the table, make sure OpenGL Shading and Perspective options are set in the View Control Bar. Depending on the screen resolution, you may have to scroll the View Control Bar all the way down. You can either drag with the LMB when the mouse is not over any of the gadgets of the View Controls bar (the cursor changes to an up/down arrow) or hold down the Ctrl while dragging with the right mouse button anywhere on the View Controls bar. Or simply use the Mouse Wheel for scrolling



The first task is to model a simple rectangular table with a simple set of legs. All five of the pieces needed for the table can be constructed using the Analytic Cube tool. Let us start by creating the tabletop first.

We create the tabletop by defining it from the top view. To switch the view to the top view, click the Top button near the top edge of the View Controls Bar.



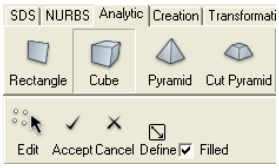


Tip

Shift modidier swaps the direction 180 degrees, so that you can also switch to Back, Right and Bottom views using the same view controls.

Make sure you are viewing the Analytic tab in the Toolbar, then click the Cube icon with the left mouse button to activate the tool.

The Control bar now displays options related to creating cubes. The default settings are fine for the current task.

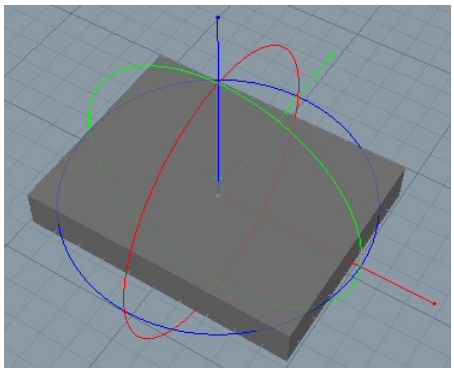


The Cube Tool activated

The Definition type is Corner-Corner, meaning you define the upper left corner and the lower right corner of the rectangle. Filled is checked, so a closed cube with six sides will be constructed.

Click the Left Mouse Button once near the top left corner of the view window to define the upper right corner (do NOT hold down the LMB, just click). Move the mouse down to the right bottom corner of the view. Click with the LMB button again. A cube is created.

Controlling the Orientation of the View Window



Rotate the view by dragging with the right mouse button while holding down the Alt key.

To verify that you really created a cube, you might want to rotate the view window a bit.

You can rotate the view window interactively, using the mouse and a key modifier. To rotate, zoom, or pan the View Window, hold down either the Alt (rotate), Ctrl (pan), or Shift (zoom) key and drag with the right mouse button.

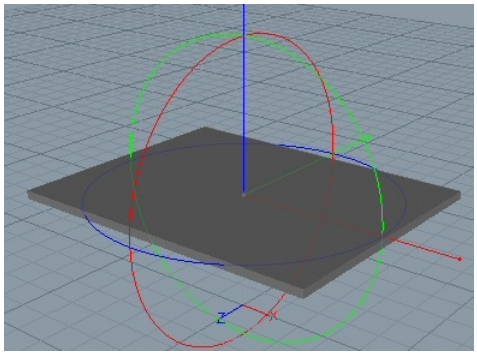
There are several other ways for you to alter the View window or change its orientation. The view window's compass menu is one alternative. Hold down the 'v' key and drag the RMB in the view window until the compass menu appears. The outermost items are view orientation settings. Here, in addition to the Front, Top, and Left-side views, you can also find menus to rotate the current view left, right, up and down 90 degrees.



Compass menu

Transforming

The three circles and axes associated with the created cube are so called transformation handles. These handles are shown for all selected object and you can drag them to move, rotate or scale the object.



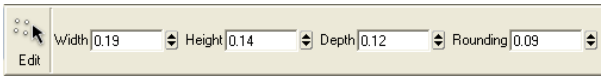
Cube translated and scaled in 'y' direction

The created cube appears to be slightly too thick, so let's scale it in its 'y' direction. To do this, move the mouse over the end point of the vertical 'y' axis handle. Then press the left mouse button key down to start scaling. Move the mouse downwards to see how the object gets scaled in its y direction. Release the mouse button when you are happy with the thickness of the tabletop.

The tabletop also lies now on the ground plane, so let's try moving it up a bit. To do this, move the mouse over the actual 'y' axis handle. Then press the left mouse button down and move the mouse upwards to lift the tabletop. Release the mouse to exit the move tool.

Similary, you would be able to rotate objects using the three cirular rotation handles.

When you select an object, the Control Bar (the context sensitive tool bar) shows you the most commonly needed tools for editing the selected object. In this case, it shows you the controls for editing the size and the edge roundings of the cube.



Cube specific controls

Most real world objects have slightly rounded edges. By default, the program displays measures using meter scale. Therefore, set the Rounding field to a small value, such as 0.005 = 5 mm. The preferred measuring unit can be selected using File/Preferences pull down menu.

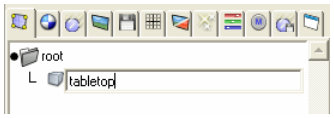
Note that the view window doesn't show you the rounding in any way. The rounding will show up later when we create an image using photo realistic ray tracing.

Renaming

The Select window at the left side of the screen has several tabs on it. Currently, it is displaying the Geometric Objects tab, which shows the hierarchical structure of the object geometry of your project.

In general, selecting objects from the Select window is more accurate and far less error prone than trying to select objects in a view window.

The default name for created objects is typically the name of the object type (cube for instance) followed by a number. Since this is not a good name name for our tabletop, we'll change it.



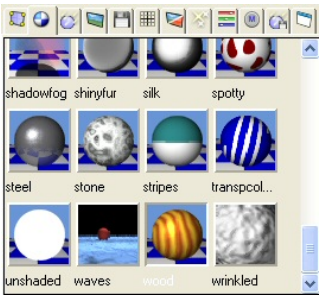
Rename the cube as tabletop

Move the mouse over the cube object in the select window, and then left click the name slowly two times. You can also select Rename from the pop-up menu of the select window. The default name of the level is highlighted and placed in an edit box. Type the word "tabletop" and press the Enter key.

Using Materials

The next step is to assign a 'wood' material with our tabletop object. Make sure the tabletop object is still selected and then click on the Materials tab. The select window now displays the current material library.

Click the Wood material to select it (you may need to scroll the material list first). Next, click the right mouse button over the wood material icon to open a popup menu. This popup menu is different from the menu that was available from the Select window. (The popup menu is context sensitive, and its contents depend on the place where you open it.)



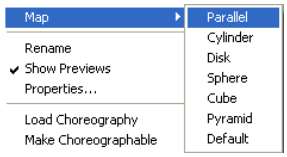
Select the 'wood' material



Note

If you see only a list of material names instead of the preview images, click with the right mouse button and select Show Previews from the popup menu.

There is a Map submenu at the bottom of the popup menu. Move the cursor down the list to highlight the Map option. A submenu opens that displays the available mapping object types.

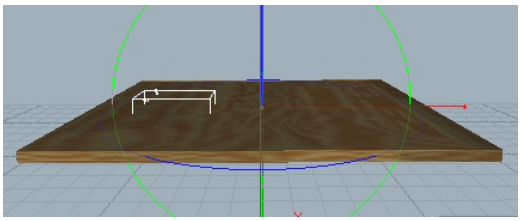


Select Parallel Map Tool

A mapping object wraps a material to a target object according to a specific map geometry, which can be cube-like, disk-like, cylindrical, spherical, etc. Select Parallel mapping from the menu.

Which one is most suitable depends on both the characteristics of the material and the shape of the target object. The Wood material is designed so that it works best with parallel mapping.

Then in the view window, switch to the Side view and define the parallel map geometry with two mouse clicks. Just make the parallel mapping object smaller to achieve suitably dense wood grains.



The white parallel map box defines wood grain projection



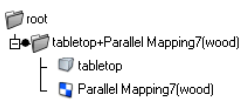
Note

The direction, from which you draw the parallel map (the Side view above) defines how wood grains travel trough the tabletob.

The Hierarchy

Click the Geometric Objects tab (first on the left) of the Select window. Notice that the structure has changed; now, instead of a "tabletop" object under the root, you have a new folder called "tabletop + Parallel Mapping(wood)". Open this folder by clicking the little box at the left side of the name.

You can see that the new folder contains the tabletop object and a parallel mapping object. When you drew the parallel map of the Wood material for the tabletop object, the mapping tool created a new hierarchy level and placed the wood mapping and the target cube into that level.



Tabletop consists of a cube and a wood mapping

Realsoft 3D uses the concept of hierarchy extensively to manage the data a user places in the scene. For example, a human body consists of several parts: arms and legs, a head, a torso, etc. In turn, an arm has several subparts like upper arm, forearm, and hand; a hand consists of several more parts, and so on. The object hierarchy is designed to categorize and collect object groups, in the same way that the file system on your computer does.

Similarly, the wooden tabletop in this example consists of a tabletop shape and a wood mapping object. Select the 'parallel mapping(wood)' object. If you want bigger wood grains, just scale the mapping object bigger. If you want to control the direction of the wood grain, rotate the mapping object. After adjusting the mapping, click the WF-Inv button in the MiscTools bar below the select window to hide the wireframe of the parallel map. We do not need to see it any more, so hiding it helps to keep the scene visually cleaner and simpler.



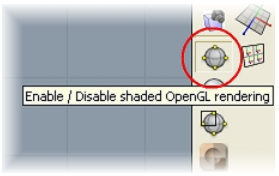
Tip

A mapping object only affects other objects that are in the same hierarchy level!

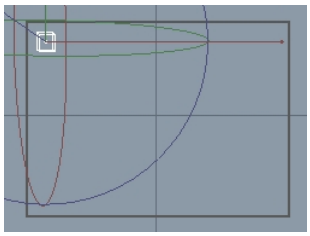
The next step is to create four legs for the table.

Modeling a wooden leg for our table is easy, now that we know how to create a wooden tabletop so we repeat the steps very briefly now.

Switch to the top view. The legs will be created under the tabletop. The shaded tabletop now blocks our view under the table, making further modeling hard. Therefore, turn Shaded OpenGL drawing off using the view control icon in the lowest group of the view control bar.



Turn Shaded OpenGL off



Create a cube representing a leg

Look at the select window; you should see so called Current Level marker (by default, a black dot) at the left side of the 'tabletop+parallel mapping(wood)' name. The Current Level defines where the program places new objects. A leg is not a part of the tabletop, so click the root folder to make it the current level.

Enable the Cube tool. Then enter two points in the view window, somewhere close a tabletop corner, to create a cube representing a leg.

Rename the created cube as 'leg'.

You may find it quite difficult to place the cube properly using perspective view. As intuitive as the perspective view is, it makes accurate modeling with the mouse difficult. Turn off the Perspective option either using View popup/Camera/Perspective menu or from the View control bar's Perpective icon. Then switch to the front view. Use the transformation handles to scale the leg longer and fit it under the tabletop. The parallel view projection makes these steps easy and accurate.

Switch back to the top view. Attach wood the material to the leg exactly the same way as you attached it to the tabletop. This time use the top view when drawing the parallel map, to align the wood grains with the leg. If you used the side view, the wood grain would be perpendicular to the leg, which is quite unnatural.

Now the 'root' object consists of two sub objects: a wooden tabletop and a wooden leg.

Duplicating

Rather than making three more legs from scratch, we can just duplicate the one we already have. To do this, make first sure the 'leg + parallel map(wood)' object is selected, then click the RMB in the Select window, and choose Duplicate from the popup menu.

Then drag one of the translation handles in the view window to move the newly created leg to the opposite edge of the tabletop.

Rather than duplicating the remaining legs one by one, we can multi select both legs, duplicate and translate them in one simple operation.



To multi select the two legs, hold down the Shift key and click each leg object with the mouse. You can click them either in the select window or in the view window.

Legs multiselected



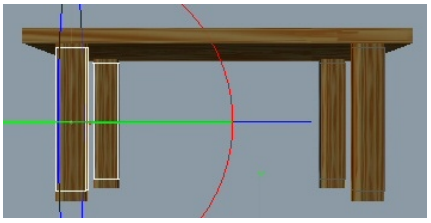
Tip

Realsoft 3D supports the following selection modifiers:

- Shift - multi select
- Ctrl - select range
- Alt - subtract from the current selection

We could again use the Duplicate menu to get more legs. However, there is even easier way: transformation handles support automatic duplication via Ctrl key. Let's try this.

Switch to the side view. Then press the Ctrl key down and drag the translation handle. The selected legs are instantly duplicated and the move operation is applied to them. Move the two new legs to the opposite edge of the tabletop.



Legs duplicated and translated

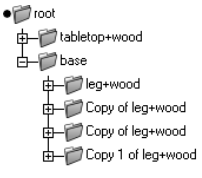
We have now created all the geometric objects we need for our table. Enable Perspective view and OpenGL shading again. Use the same controls as before - they toggle the mentioned View properties on and off.

Manipulating the Hierarchy

We have created all the table parts under the root level. It would be much better to put the legs inside a single dedicated level object 'base'. Such a structure has many advantages. For example, we can then modify all the four legs easily, simply by modifying the 'base' object.

To create a new level, open the popup menu of the Select window with a right mouse button click. Select New/Level from the popup menu. Rename the created new level object as 'base'.

Then multi select the four legs and drag & drop them inside the base level.



Legs moved inside a new level

In addition to drag & drop, you can group objects into new levels using the 'Drop to a level' tool. To move both the tabletop and the base in a separate level called 'table', multi select the objects and then select the item Drop to level from the pop-up menu of the select window. Rename the new level which now contains all modeled parts as 'table'. The root object nown consists of only one sub object, the table.

Creating Light Sources

We now have a wooden table, but the scene is quite dark. Only the dim flashlight of the View camera illuminates the objects. The next task is to add a light source to the scene.

Select the Light Sources section of the Toolbar at the top of the screen and click the Point light source icon.



Activate the point light tool

Switch to the side view and enter two points in the view window - somewhere above the table - to define the position and radius for the point light object. The distance between the two clicks will define the falloff rate of light intensity.

Now, there is also a Point light object in the hierarchy. The tool control bar displays some useful controls for it. Let's experiment with them. For example, change the Falloff property from the default value None to Distance, so that light intensity decreases by distance just like in the real world. Then increase the Intensity value to 1.0.



Note

Unlike the material mapping objects, which affect only objects at the same hierarchy level, light source objects, by default, illuminate all objects in the model regardless of the hierarchy.

Editing Geometry

Earlier, you transformed the tabletop cube using the the transformation handles, which automatically show up when you select an object.

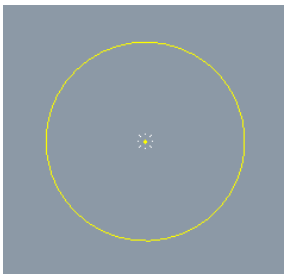
These transformation handles are common to all objects: for example, the rotate handle consisting of three perpendicular circles can be used for rotating all kinds of objects.

Some handles are object specific. You can ask the selected object to show its object specific handles by clicking the Edit button in the Control Bar.

We created the light source by entering two points, which defined the center and the fall-of radius for light source. Both properties can be edited after creation using the Edit mode.

Select the light source and click the Edit button, or simply hit the space bar key, which does the same thing.

A circular handle shows up around the light source. This circle represents the fall-off radius, which specifies the distance where the intensity decreases to 50%. If the light source doesn't seem to illuminate objects of you scene enough, drag this intensity handle bigger.



Point light specific handles

Only the point light source has this kind of handle. For example, a cube defines other kind of handles (six face handles), which can be simpy dragged around to edit the size of the cube.

Ray Tracing

So far, we have worked with our table in shaded mode, so let's see what our table looks like when it is ray traced. Make any adjustments you want to the view orientation using Alt+dragRMB to get a nice view of the table. You can then render the view window by clicking the Render View button, which is at the bottom group of the View Control bar.

Note that all icons have tool tips. If you do not know what a particular button represents, move the cursor over the icon, and in a moment a short message explaining the icon is displayed. You can also activate rendering using the view popup menu Render/Ray Trace.

Shadows and Reflections

Light sources cast shadows by default. In order to see shadows, we need something the table can shadow. So let us use create a floor.

The floor we need can be represented by a large rectangle. Now, you should already have a pretty good idea how to do this: switch the view window to the Top view, use Shift+RMB to zoom out so that you can create quite large floor. Then enable the Rectangle tool in the Analytic tab of the tool bar and enter two points in the view window to define the size for the rectangle. Take a side view and move the floor just below the table.

To make the floor more interesting, you might want to attach a nice material to it too.

Some of the materials define reflectivity, in which case they automatically reflect the surrounding world in a physically correct way. Let's see how this work.

Switch to the front view, then select the Analytic Sphere tool from the Analytic tab of the Toolbar. Click twice in the view window above the table to define the center and the radius for the sphere.

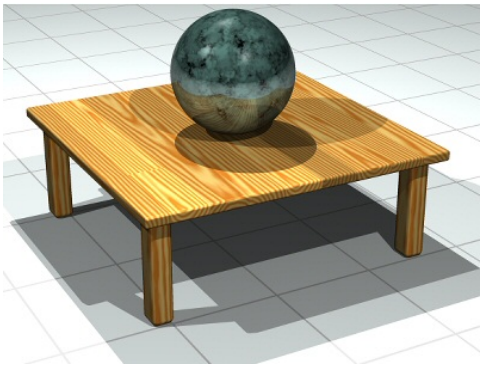
A gray sphere is quite boring, so let's apply a material to the sphere, but this time using the drag&drop interface.

Make sure that the sphere is selected, then switch to Materials tab of the Select window. Move the mouse pointer over the dark 'marble1' material, and press and hold down the left mouse button. You have just "grabbed" the marble material. Now drag the mouse into the view window and release the mouse button. The sphere has been assigned the marble material. To verify this, switch back to the Geometric Objects tab of the Select window and look at your hierarchy.

Drag&Drop is another way to map materials to objects and very useful when you don't have to define a specific mapping geometry. Earlier, when assigning wood to table legs, we wanted to make sure that grains follow the leg direction,and therefore explictly drew the required parallel map. Now we did not worry about the orientation of the marble pattern, and therefore the quickest way to assign a material was quite suitable.

Rotate the view upwards (Alt + dragRMB) and click the Ray Trace button in the View Controls bar to check the result.

The image is starting to look more interesting. The table casts shadows on the floor. The marble sphere should reflect the table.



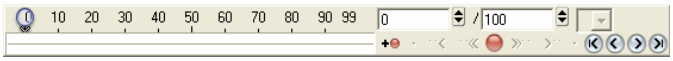
A marbled sphere on a wooden table

Creating an Animation

The next part of our task is to animate the sphere so it rolls across the table. There is much to say about animation, and we will get to that in later tutorials. For now, what you need to know is that you "key" the position, rotation, etc. of objects at particular frames, and let the computer fill in the in-between frames (often called "tweens"). So how do you do it?

Realsoft 3D has the normal main menu below the program title bar. It contains all the general-purpose tools, and if the icon of some tool is not included in the toolbar (because of lack of screen space, etc.), you can find it from here. The main menu also includes the functions that are needed for project management and changing the working environment.

We need to display the Animation window. It appears at the bottom of the main window, below the view window, and looks like the following.

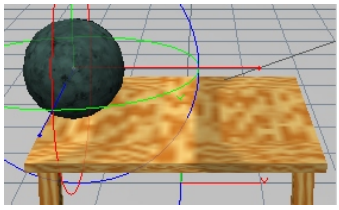


The Animation Window

If this window is not visible already, go to the View menu (main menu bar) and select Animation window from the list. If you don't need animation controls, you can hide the window to save some screen space.

We want to animate the marbled sphere, so select the 'sphere+marble' object from the hierarchy tree of the Select window.

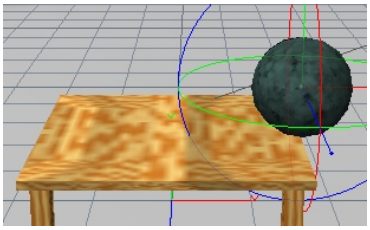
Place the sphere at the position where you want the animation to start (in other words, your initial scene). For example, use the 'x' translation handle to place the sphere at the back of the table.



Sphere in frame 0

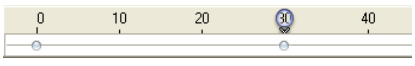
At the right end of the Animation window, there is a red circular button. Click this button. Now animation recording is on and all modifications to the table scene applied while the recording is on are stored as into the animation.

Move the time slider in the Animation window to frame 30, which represents 1 second in NTSC format. If you prefer, move the time slider to frame 25 for 1 second in PAL format.



Sphere in frame 30

Drag the 'x' translation handle to move the sphere to the right edge of the table. Congratulations - you have just created an animation!



Key frames

The Animation Window should now show you two knots, placed at frames 0 and 30 correspondingly. These represent the sphere position in time.

Try moving the time slider between these two knot to verify that the sphere really moves between the two edges of the table when you move the time slider between the two knots in the time line.

You can move these knots to adjust timing. For example, if the sphere moves too slowly on the table, drag the second keyframe from the frame 30 to frame 20 so that the sphere will reach the edge of the table in frame 20.

Let's create a few more key frames. Move the time slider to the frame 60, drag the sphere further over the table edge and then drag the 'y' translation handle to move the sphere down to the floor. Move the time slider to the frame 90. Drag the 'y' translation handle to move the sphere up a bit, as if it had bounced from the floor.

You have now created an animation where the sphere moves on the table, then falls down to the floor and bounce up once.

Click the red animation Record button again to turn off the recording mode.

- Click the Play button in the Animation Window to see your animation.

Animation may not look that realistic yet, as only a couple of key frames is not sufficient to represent a realistic bouncing motion. You can fine tune the animation the same way you created it in the first place: enter animation recording mode and modify the sphere in desired frames to fine tune the motions.



Note

You can also modify the animated sphere animation recording mode disabled. For example, if you animate a sphere to bounce on a table, and you then move it down the floor recording turned off, the sphere bounces on the floor. In other words, if you transform the object in animation recording mode, you edit/create key frames. If you modify the object without animation recording on, then the transformation affects uniformly all motions.

Saving Projects

At this point, we must also perform one important step: save the work done so far. Therefore, select File/Save As from the main menu. Choose a suitable folder and file name for saving the project using the browse button. Press OK to finish the Save operation.

Rendering the Animation

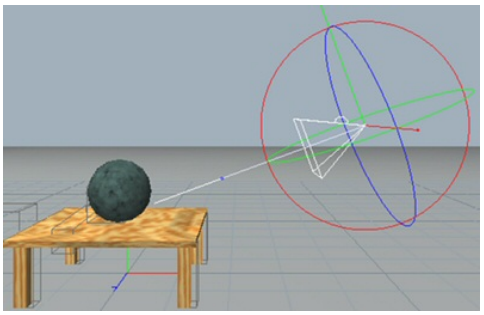
In most cases, the created animation must be rendered to a suitable file format, so that it can be presented as a part of a multimedia show or in a home page, etc. This is the final step for this tutorial.

When rendering images to disk files, it is necessary to create a camera object for the scene. The camera defines how the scene is viewed in the rendered pictures. The view window cannot define this, because there may be many views opened simultaneously, or no views at all.



Click the Camera tool icon on the Creation section of the Toolbar. Then enter three points through the view window to specify the position, the aimpoint and the viewing angle for the camera.

As you can see, the created camera has the standard transformation handles. The camera object is also added to your object hierarchy. It can be selected, modified and animated similar to any geometric object, such as the bouncing sphere above.



Camera object created



Tip

If you click the Accept button in the Control Bar without entering any points in the view window, the orientation of the current view window will be used for determining the orientation of the created camera.

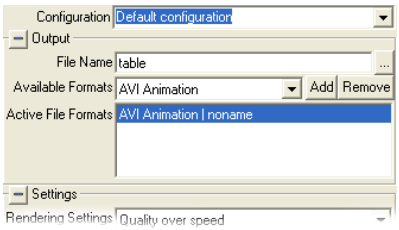
If you want to see how things look from the cameras stand point, simply drag&drop the camera object from the select window into the view window. If you lost your sphere or table in the view window (for example, you accidentally panned the view too much) just drag the desired object from the select window into the view window. This turns the view to show the dropped object.

Adjust the camera position and other attributes until the camera shows the scene in a suitable way.

Select File/Render from the main menu. This opens a File Rendering window with settings for controlling the final rendering of the animation. Do the following:

In the Output section, enter a suitable path and filename (e.g. 'images/myanim') in the File Name field. Use the browse gadget if necessary.

By default the file rendering window renders images and animations to .BMP format, which is not what we want now. Left click to highlight BMP from the Active File Formats list, then press the Remove button to remove the BMP format from the list.



The Rendering Window

Press the downward pointing arrow to the right of the Available Formats field. Select 'MPEG Animation' from the list, and then click the Add button.

In the Settings section, select the value 'Reasonable quality' for the Rendering Settings field.

In the Resolution section, set the Width field to 400 and the Height field to 400. The height and width should be the same, because the camera we created above has a symmetric image aspect ratio 1.0 (=square). If the resolution ratio h/w and the image aspect of the camera do not match, rendered pixels will not be perfect squares but stretched rectangles.

Press the Render Animation button to start rendering the scene. This button renders the entire animation, whereas the Render Image button renders a single still image, based on the current position of the time slider.

Wait until the progress indicator shows 100% and the Render Animation button and Render Image button become enabled again. This indicates that rendering has completed. You can now play the animation using your favorite media player. On many systems, all you need to do is double-click the AVI file from the File Manager.

Summary

This first tutorial has introduced several of the basic concepts of Realsoft 3D . An important element of this tutorial was to show you that there are a number of ways to activate various functions in Realsoft 3D.

All these different methods of performing the same function may seem confusing at first, but it allows for much flexibility as you learn the program. Beginners may use simple methods like tool icons or menus, whereas experienced users may use fast methods like keyboard shortcuts and compass menus. The program includes thousands of features, and all the input devices - mouse, keyboard, etc. - are loaded with power features. As you learn to use the program, it becomes possible to execute a complicated workflow surprisingly fast by combining these techniques.

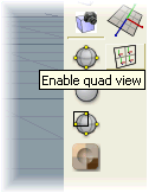
This marks the end of the first tutorial. Congratulations on creating your first scene. Hopefully, this first lesson has whetted your appetite, and demonstrated some of the power of Realsoft 3D. Happily, there is much, much more. Further lessons introduce and explore more specific features and techniques. All that can be said at this point is: be patient. 3D modeling is a huge topic, but isn't learning fun!

Modeling Your First Building

Building Walls

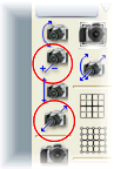
Start the program. Change the view window to the quadview mode by clicking the quadview button. The mentioned button, as well as many other useful view navigation and control tools are grouped into the vertical view control bar at the right side of the view window.

Click the top left quadrant of the view (labeled as 'Top) to activate it. Then click the Quadview button again to maximize the selected top view. Alternatively, you can press Alt key down and double click the selected view quadrant to maximize it.



The view control bar is located at the right side of the view window. The Quadview button is highlighted and shows the tool tip.

Use the navigation controls to zoom out and move the view camera far enough from the origin, so that you will see enough space for modeling a complete building. Note that the status window displays changed camera parameters while you use the navigation controls. The view window should display about at least 15 meters wide area of space, and the camera should be moved at least 10 meters away from the origin. Undesired clipping problems arise, if parts of the building reach out all the way behind the camera.



Zoom and Distance controls



Note

In this software, the XZ plane is the ground plane and Y axis points upwards. You can see the orientation of world from the coordinate system drawn in the view window. The direction convention can be changed using the Space selector of the File/Preferences window.



Activate the Building tool. It is located in the Creation tab of the toolbar.

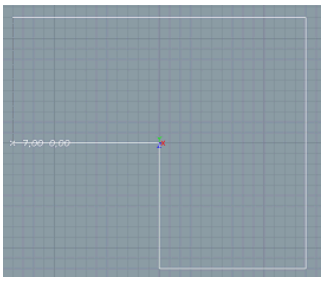
First you need to draw a closed loop, which defines the outer walls of the house. Click somewhere on the top left corner of the view window to start drawing the walls.



Note

Click the left button - do not drag the mouse while holding the left button down. Dragging operation is reserved for point snapping.

Move the mouse to see a rectangular shape drawn on the view window. By default, the building tool is in automatic mode, which draws a frequently needed rectangular base for the building. However, if the second click is exactly in a vertical or horizontal line from the first click, the tool continues accepting an arbitrary sequence of wall segments. Let's use the latter alternative.



Draw a line sequence, which defines the walls

Move the mouse to the opposite side of the view window horizontally. To easily find an exact horizontal level, just hold the Shift key down. When the first wall segment is 14 meters wide, click second time. Then continue drawing the base of the house until you have an L shaped form shown in the example picture. To finish the drawing, either click the Accept button in the toolbar or click again on top of the first wall point to close the loop.



Note

If you accidentally enter a point in a wrong position, hit the backspace key to step back in drawing.

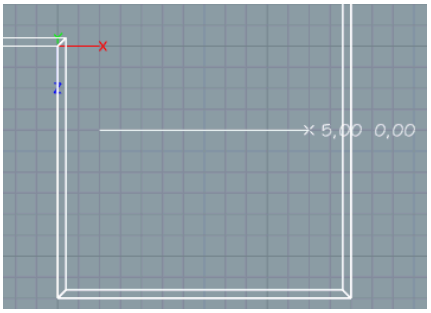
A new object called 'buildingX' appears into the select window, which is located at the left side of the screen. The select window shows you the structure of your current scene. It is very important tool and we will learn more about it later.

Let's draw some inner walls next. While a building object is selected, the second row of the toolbar automatically shows you some frequently needed controls and tools, which are useful for building construction. Click the Sub Wall tool.



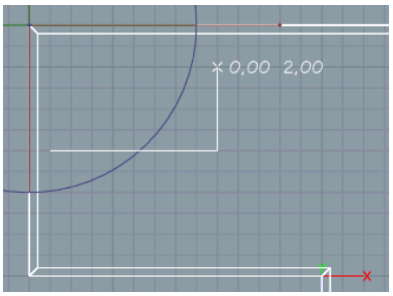
Sub Wall

Click two times to draw a horizontal wall as shown in the picture Note that you do not have to aim carefully to match the inner wall with outer walls. After activating the Sub Wall tool, take a look at the second row of the toolbar; it now shows the options of the Sub Wall tool. Snap Start and Snap End options are active by default, indicating that the tool will automatically fill the gaps. Finish wall drawing by clicking Accept or hitting the Enter key.

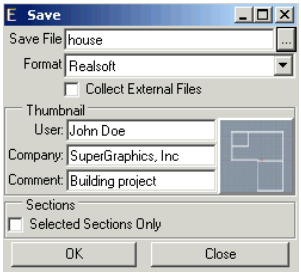


Draw the first sub wall with 2 mouse clicks. Length does not matter.

Activate the Sub Wall tool again and draw a second set of inner walls as shown in the picture. As you can see, sub walls can have multiple points and turns.



The second sub wall has 3 points



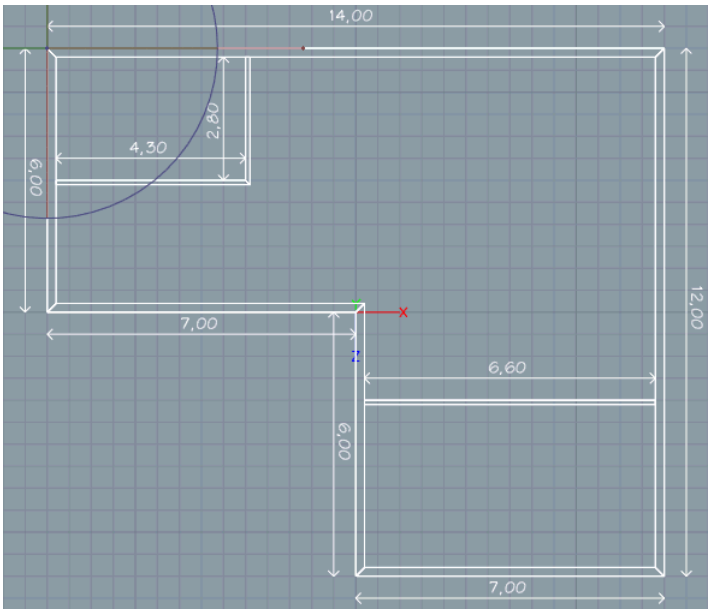
Save the project at regular intervals

It's time to save the work. Select the pull down menu File/Save As. After the save window has opened, define a suitable file name for saving, such as 'house'. You can also fill in some information to the Thumbnail fields. By clicking the thumbnail image, you can record the contents of the view into the thumbnail. Click OK to save the project.

Editing the Walls

The dimensions and walls of the building can be changed easily. All parameters can be found from the hierarchical structure of the building. The building system is fully parametric. When any part of a building definition is modified, the required parts of the geometry are rebuilt again.

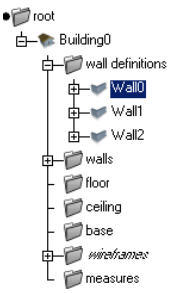
Make sure that the building object is still selected (its name is highlighted in the select window). You can see the dimensions of the walls by selecting Measure Lines=Left from the tool bar.



Measuring lines enabled

Click the little square which has a + sign inside it at the left side of the building item on the select window. The building folder opens and shows its sub objects. Open the first sub level called wall definitions. It contains three wall objects. The first one is what we drew first, the outer walls. Select it.

Note how the second row of the toolbar immediately shows some options for the selected wall. Change the thickness of the wall from 0.2 meters to 0.25 meters. Then change the extrusion side from Right to Left. The polyline for outer walls originally defined the outer dimensions of the building, but now the walls are built outside the polyline. While you made the changes, sub walls automatically adjusted their length, thanks to the auto snap option.



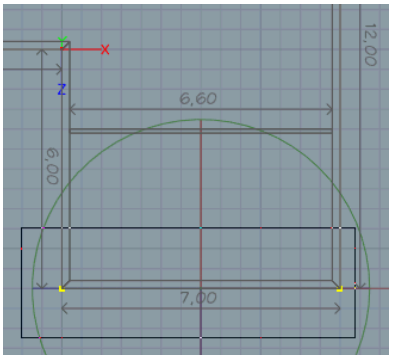
The hierarchy of the building opened. Outer wall definition selected.



Note

You can select the number of displayed decimals using File/Preferences pull down menu, and the Metrics tab in the opened window. 2 decimals is suitable for this tutorial, assuming that the Metrics/Distance is set to use 'm' = meter unit. You can then view the dimensions with a centimeter accuracy, which is enough in building scale modeling.

Open the up most wall level on the select window, and select the polyline object called 'nurbsXX' inside it. Hit Spacebar or click the Edit button in the toolbar to put it into the edit mode. The line displays some little red squares - its point edit handles - on the view window. Drag-select (move the mouse holding the left button down) a point pair on the bottom part of the wall, as seen on the picture.



A point pair selected. The drag selection box is also included in the picture.

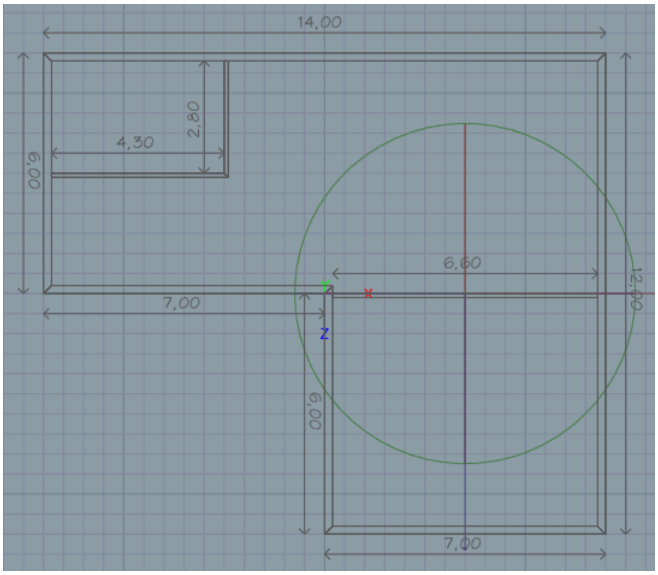
A circle and some axis lines show up around the point pair. The circle is the rotation handle, and lines are translation handles. The end knobs of axis lines are for scaling. Try modifying the point pair by dragging these handles with the mouse. Note that you can cancel an experiment by hitting Esc key, and undo the action afterwards using the Exit/Undo pull down menu. After experiments, expand the building to Z axis direction by dragging the blue axis line 2 grid units downwards.



Note

When you select the nurbs polyline, the toolbar displays a large set of tools for editing the line. With these tools, you can for example add more points or delete existing ones.

Above, we used point editing to modify the walls. A wall can be modified as a complete object, too. Select the second wall object from the select window. Then drag blue axis line on the view window to move the sub wall. Note how the sub wall automatically expands while you move it. Place the wall exactly in the corner as shown in the picture below.



Sub wall moved to a new position

Now we can stop experimenting with wall editing. Close the level 'wall definitions' in the select window to keep the hierarchy display simple. Then select the whole building object from the select window. Turn measuring lines off using the control in the tool bar.

So far, the results of our modeling actions were seen in a simple wireframe model. Next we will experiment how to model in perspective shaded view. But before that, let's add a light source. Alt-double click the view to see the quadview, then Alt-double click the Side view quadrant to maximize it.

The Zoom button is located in the top part of the view control window.

Zoom Tool

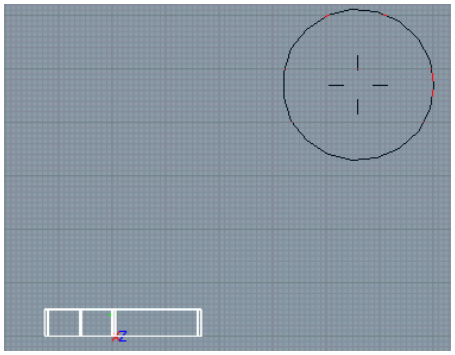
Move the mouse on top of it, press LMB down and drag downwards, so that you will see a larger area of space around the house.



Tip

The quickest way to zoom in and out is to hold the Shift key and the right mouse button down and move the mouse up and down above the view window.

Select the root level. Switch to the Light Sources tab and activate the Point Light tool. Click somewhere above and in front of the house. The falloff radius defined by the second click does not matter this time, so click a second time anywhere on the view.



Adding a point light

Go back to Quadview (Alt-doubleclick) and maximize the bottom right quadrant, which shows the perspective shaded view.

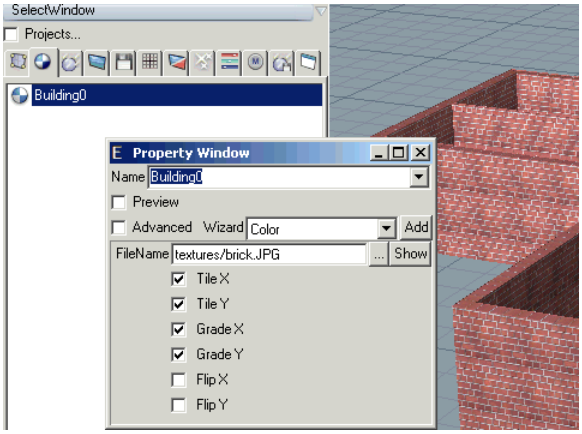
Switch the select window to the second tab from the left. It shows the current material library. The default startup library includes a large collection of materials, but we will need only some of them. So, select Purge from the popup menu of the select window. Only one material called building remains in the material library. Double click it select it and to open the property window.

The building material was actually created by the building tool, when we applied it in the beginning of the tutorial. So called Auto Texture option does this unless the user explicitly turns it off. So, all we need to do now is to click the browse button at the right side of the Filename control, and select a texture map using the file browser. Pick for example bricks.jpg from the textures folder. Then close the property window.



Note

The sub folder 'Walls' of the building contains a texture mapping object 'Default mapping(building)'. It assigns the material to the walls. If you select this material map object, you can change its target material to another alternative using the material selector gadget, which appears in the tool bar as soon as the material map object becomes selected.



Select a brick texture for the walls of the building

Go back to the leftmost tab of the Select Window. Click the building object to select it. Currently the building has only walls. To add a base plate under it, go to the second row of the toolbar and set Base=Fitted to Walls.



Note

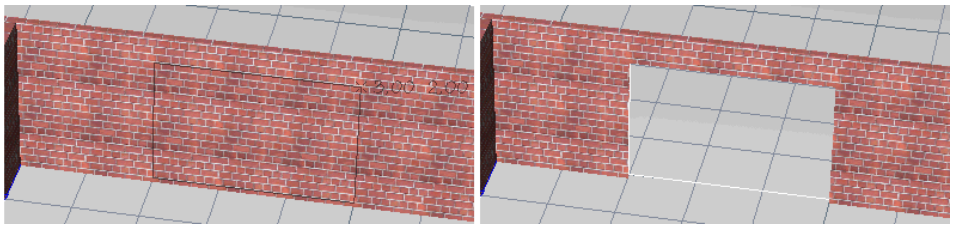
The narrow toolbar has room for only the basic options. For more detailed control, select the building object and open the Property Window (from the popup menu of the select window, by double clicking the object or using the 'p' hotkey). Then, in the spec tab of the property window, you can adjust the height of the base plate and other building properties

Rotate the view around the house using the view rotate button at the top of the view control bar. Another way is to hold the Alt key down and drag in the view window with the right mouse button. The view rotates to the direction where you move the mouse. Another useful navigation control of the view control bar is the pan tool (available also via Ctrl+RMB dragging). Try these controls to find a view where you see the straight inner wall.

Let's cut away part of the inner wall. Click the Hole tool in the toolbar.

 *Hole Tool*

Move the mouse pointer to a position where the wall meets the floor and click to define the bottom left corner of the hole. Move the mouse to the left and upward. Note how the hole snaps to the 0.5 meter grid units. Click second time when the hole is 3 meters wide and 2 meters high.



Cutting a wall with the hole tool



Note

Just like the wall definitions, you can find the new hole object from the hierarchy of the building. It is located under a sub level called holes. Open the level and select the hole. Go to the tool bar and change its Width to a new value, say, 2.5 meters. You can also move the hole around using its transformation handles.

It's again time to save the project. Select File/Save As from the pull down menu. Change the file name for example to 'house2'. This way you will get multiple backups of the project, representing different stages of the modeling process.

Windows and Doors

We could add all windows and doors to the building using the hole tool. However, there is even quicker method available: drag and drop from object libraries.

Modeling in shaded perspective view is fun but often the best accuracy is obtained by using parallel projection and a wireframe view. So, alt-double click the view to get back to the quad view and then alt-doubleclick the top view to maximize it.

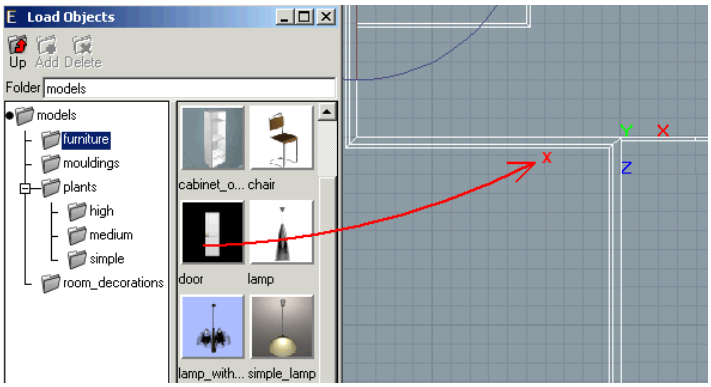
To find a good zoom scale for further modeling, switch to the leftmost geometry tab of the select window, drag the building object and drop it into the view window. Dropping an object to view performs an Auto Focus operation. The view window will zoom exactly around the selected target. You can also use the Auto Focus tool in the view control bar after selecting the building.

 *Auto Focus*

The loaded window and door items should go under the building hierarchy. Therefore, open the 'Building' level's hierarchy in the Select Window from little box at the left side of the name. The current level marker jumps from the

root to the building level. Then select Browse Object Library from the popup menu. This opens the loader window, which shows disk library objects conveniently as small thumbnails.

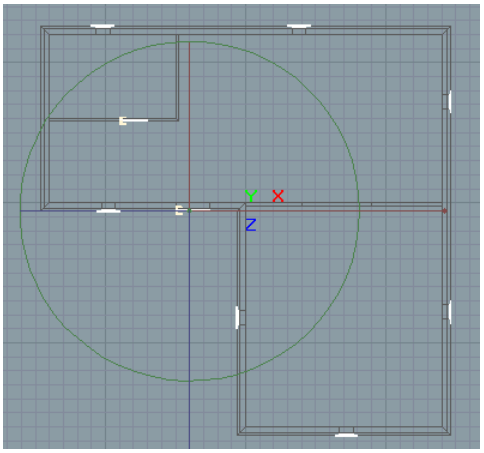
Click the library folder called furniture, and find the item door. Drag the door icon over the view window and drop it just outside the point where you want to insert the main door of the building. Do not drop it too far away from the wall; the door has auto-snapping properties, which become active if a wall surface is found closer than one meter. If the drop position was correct, the door snaps exactly to the wall and rotates itself to the direction of the wall.



A door snaps to the wall when dropped close enough

The loaded door object appears in the view window and in the select window's hierarchy tree. It is automatically selected and displays useful attributes in the toolbar. Try changing the width to 1.2 meters.

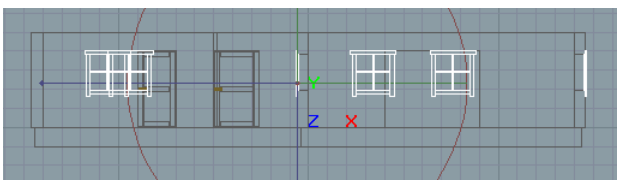
Next drag and drop another door for the small closed room in the upper corner of the house. The width of this door can be smaller, say 0.8 meters - change it using the toolbar control. Then drag and drop several windows so that every wall and room gets at least one window. Note that you have to drop the windows outside the building; otherwise the outer mouldings of the window may rotate to the wrong side of the wall.



Two doors and seven windows added

Close the loader window and take a front view. The windows are quite small; let's make them bigger. In the select window, drag a selection box around all objects called window. You can also select them all by clicking them one by one holding down the Shift key, which is the multi selection modifier.

The windows are parametric objects, which automatically display the dimensions in the tool bar. Change width and height to 1.2 meters. If the upper edge of the windows got too high, simply drag the green translation axis downwards on the view window. Note that holding down the Shift key releases grid snapping temporarily and allows you to align the windows with the upper edge of the door.



Windows resized and aligned with the door

Before proceeding into other steps of building construction, let's clean up the hierarchy. Multi select all doors and windows, and select Drop to a level from the popup menu of the select window. The selected objects are placed under a new folder. Use the Rename tool of the popup menu to rename the folder as doors_and_windows and close the folder. This action keeps your scene hierarchy more compact and better manageable. You can naturally later open the folder, select and edit any door or window if you like. Then save the project with a new name.

Decorating Rooms

Now all walls are made of solid brick, and the floor is just the base plate under the building. The room object allows individual decoration of rooms. This modeling stage happens best in the shaded perspective view. Turn it on as before and rotate the view so that you see the building from above.

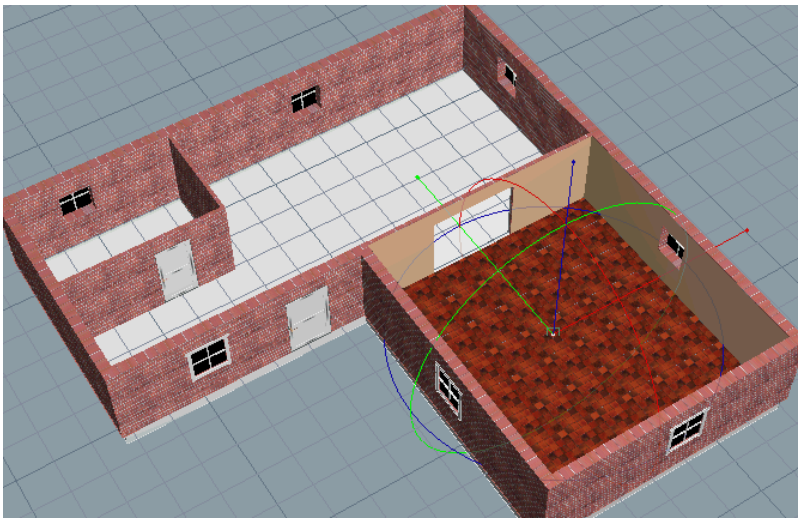
Select the building object. As usual, some useful tools for this type of object appear on the toolbar. Click the Room tool button 

Room Tool

When the room tool is activated, the tool bar shows some tool options. Change the name of the room to 'kitchen'. Click in the middle of the room, which is located in the lower right end of the house. The tool adds a new room object into the hierarchy, and white surfaces enclose the space around the clicked point. Select the room object from hierarchy, and turn off Ceiling option from the tool bar. Now we can see again into the room.

Go to the second tab of the select window, which includes the material library. The room tool has inserted a set of new material templates. Double click the material building_kitchen_floor to open the property window. Open the file selector using the filename browse button and select a texture, for example plates.jpg.

Select a second material, building_kitchen_walls. All surfaces need not to have a pattern. The Wizard selector in the property window shows Color; just click the Add button and a color selector appears on the bottom of the property window. Pick a constant color for walls, for example some nice shade of light brown.



One room decorated

Switch back to the geometry tab of the select window. Open the hierarchy of the kitchen folder. Select a folder called floormouldingsample. Apply Make Current from the select window's popup menu to make it the current level. Then pick the popup menu Browse Object Library. The loader window, which we already used for doors and windows opens. Select the mouldings folder from the browser window. Then drag and drop the item floormoulding_pine somewhere in the view window. Drop position does not matter, as long as it is in the view window. A moulding made of pinewood appears.

Then select the folder ceilingmouldingsample from the select window and apply Make Current popup tool to it. Then drag and drop the item ceilingmoulding_birch from the browser window to the view. Now the decoration of the first

room is ready. Select the kitchen object. Just for fun, drag the kitchen object around using the axis handles in the view window. As soon as the indicator reaches another room, the selected surface materials spread around the new room! Easy and fun, isn't it! Finally put the decoration back to its original place and check the Ceiling option in the tool bar. We will leave the ceiling plain white this time.



Note

Instead of loading some predefined mouldings, you can model the profile of the moulding yourself using the toolkit of the Nurbs tab of the toolbar.

We will decorate another room using a method that is even quicker and easier. Select the building object from the select window. Rotate the view so that you can see inside the little corner room. In the object browser window, select the folder called room_decorations. Drag the item 'bathroom' and drop it inside the little room. The room gets fully decorated right away. After viewing the result, just check the Ceiling option in the tool bar to add the top surface to the bathroom. It was initially disabled so that you could immediately check the result after loading the decoration.

If you look at the object hierarchy, you can see that the dropped bathroom object is placed into the building folder. If you do not like the decoration it defines or want to try another alternative, just delete the object.

Decorate the third room either step by step or using predefined decorations. Note that you can easily customize predefined library decorations by changing the textures or mouldings. When all rooms are decorated, close the browser window. You can also close the room object folders in the select window hierarchy to hide the details, which are no longer under our attention. Save the project.

The Roof

To add a roof above the building, select the building object and click the Roof tool in the toolbar.

 *Roof Tool*

By default, the roof has a rectangular base shape and a pyramid like structure. The shaded view shows that again, all surfaces are plain white by default. Switch to the material tab of the select window and double click the material building_roof. After the property window has opened, use the file browser to select a texture map called roof1.jpg.



Textured roof

The roof properties, such as the height, width of eaves etc. can be adjusted from the tool control bar and in more detail using the property window. The actual roof shape can be changed using the Roofline tool.

Go to the unshaded top view. Make sure that the roof object is selected and click the roof line tool.

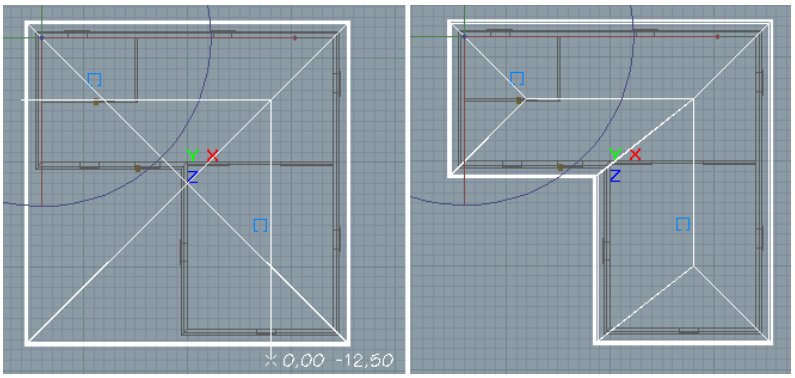
 *Roof Line tool*

Click three times to draw a line that travels in the middle of the L-shape of the building. It does not matter how far the lines reach outside the building. After the third point, click Accept or hit Enter.



Note

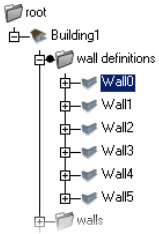
You can use Shift, x and y modifier keys to easily define exact horizontal and vertical lines.



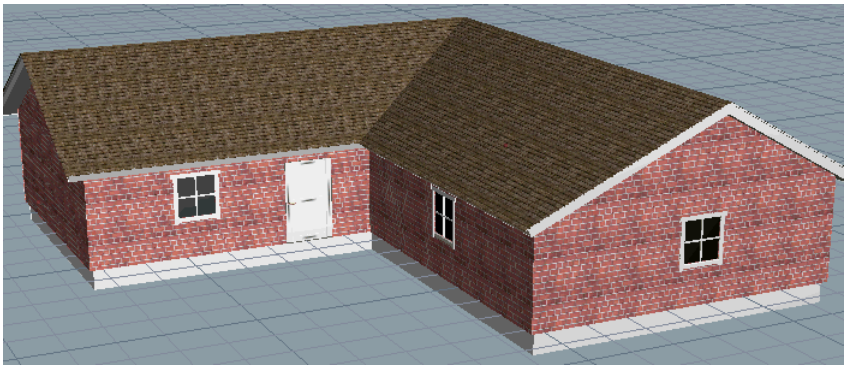
Defining the roof line

The building now has a hip roof. Change it to a gable roof by selecting Roof End = Straight from the tool bar.

Switch back to shaded perspective view and rotate around the house. As you can see, there are triangular holes below the roof at both ends of the house. To fix that, open the wall definitions folder under the building. Select the first wall object (it defines the outer walls). Then activate the Extend to Roof option from the tool bar. The roof is now ready. This is a good moment to save your project.



Select the outer wall object



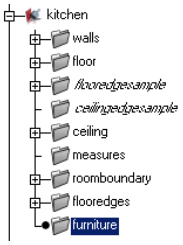
The gable roof finished

Furniture

Furniture can be modeled manually using the general modeling tools. The easiest way is to use the intelligent drag and drop tools and predefined furniture libraries.

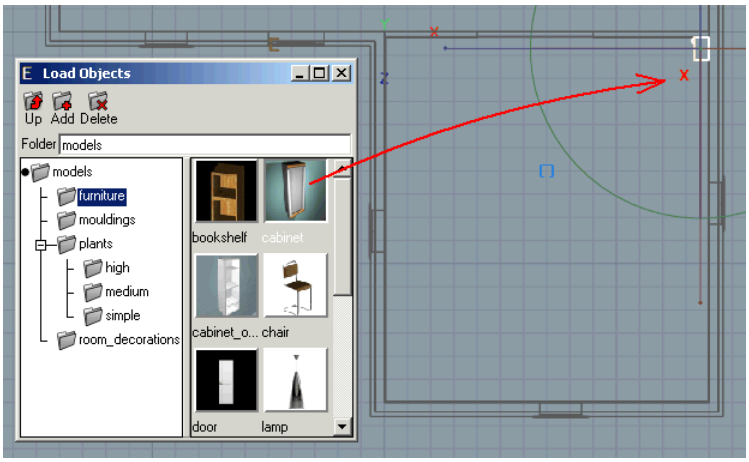
Take the top view (OpenGL shading switched off). Let's handle the kitchen first. Zoom in so that the kitchen fills the view window. Select kitchen from the select window and open its hierarchy, if not open yet. Make sure the kitchen is the current level; note that a small indicator at the left side of the hierarchy level shows what the current level is.

To keep the scene hierarchy in order, all pieces of furniture will be collected to a new level. Select New/Level from the popup menu of the select window. A new hierarchy folder appears to the end of the sub object list of the kitchen. Rename the level as 'furniture'. Then make the level current so that all loaded items will be placed there automatically.



A new level for furniture added into the kitchen and made current

Pick Browse Object Library from the popup menu of the select window. Click the furniture folder in the browser window. Find the item called cabinet. Drag and drop it to the top right corner of the kitchen, 2 grid units down from the corner and one grid unit away from the wall. The drop position is marked with a red cross in the picture below. If the drop position was correct, the cabinet snaps into the corner and aligns with the back wall of the kitchen, which was closest to the drop position.



Drop position of the cabinet and the loaded object. The cabinet aligns itself with the closest wall.

Drag the same cabinet and drop to the same position - 2 grid units down and 1 unit away from the wall. The new cabinet snaps to the end of the first cabinet.

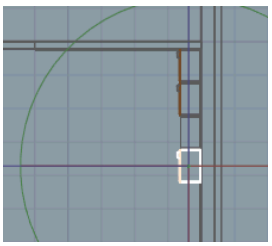


Note

Do not drop the cabinet inside the wall. It may align to the wrong side of the wallpaper!

Check the tool bar after you dropped the second cabinet. It is a parametric object, and you can easily change its width from 0.4 meters to, say, 0.6 meters.

Next drop the cabinet_open to the end of the cabinet row. This time you must shift the drag position one grid unit further. Then add one more cabinet to finish the cabinet group.



Four cabinets loaded



Note

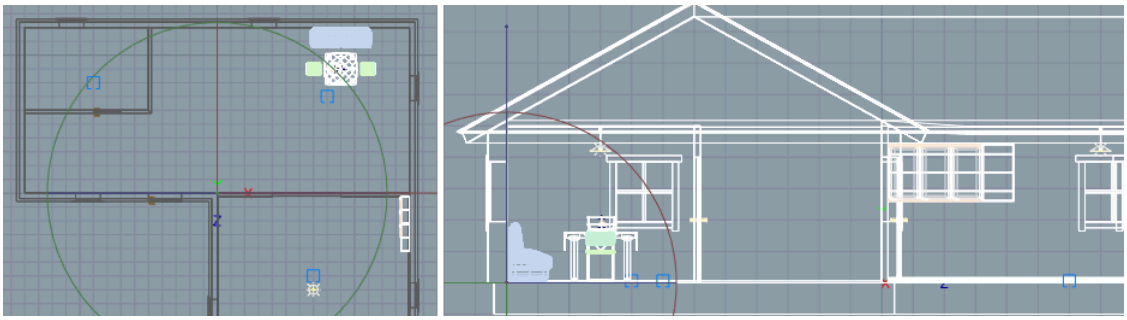
If you accidentally drop a cabinet to a wrong place, just move the cabinet to its intended place or apply Edit/Undo pull down menu and try again.

To finish the experiments in the kitchen, drop the object simple_lamp to the middle of the kitchen. Lamp does not snap to any wall, but it moves in view's orthogonal direction to find the ceiling. If you forgot to enable Ceiling after decorating the kitchen, lamp will travel all the way up to the roof.

Add also some furniture to the middle room. First make it the current level, then add a new sub level called furniture, then make the furniture level the current level. Add the furniture as follows:

- Drop a sofa object close to a wall so that it snaps to the wall.
- Drop a table object in front of the sofa. The table snaps only in Y direction to the floor.
- Drop a chair by the table. Rotate it with the circular rotation handle if necessary.
- Ctrl-drag the chair to the opposite side of the table. Dragging an object with Ctrl key automatically duplicates the object. Rotate the copy 180 degrees.
- Drop a lamp object on top of the table.

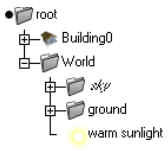
After these steps, take a side view to verify that all objects have snapped vertically to a correct height.



Some pieces of furniture added. Top and side view.

Landscape

The house is floating in empty space. You can model and texture the environment manually, or use predefined world models. We will now use the latter approach. Select the pull down menu Layers/World/flat_ground. New objects appear into the select window hierarchy. The loaded world model also includes a sun, so select the original light point from the select window and hit Del key to delete it.

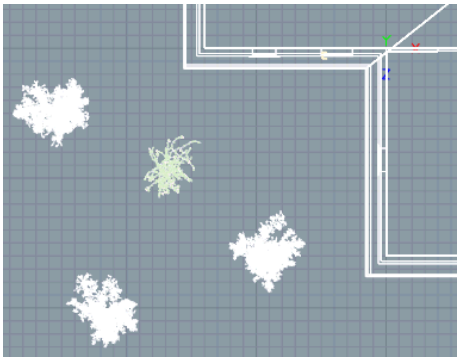


Add a world around the house

Take the front view. Make sure that the base of the house meets the ground level. If not, you can fix it easily:

- Method 1: Select the ground level and move it upwards using the translation handle (hold Shift to release grid snapping)
- Method 2: Select the building and move it up or down
- Method 3: Select the building, open the property window, go to the Spec tab and change the height of the base with the numeric slider

Go to the top view. Select the level called World. Make it the current level. From now on, new items will be loaded to the world level. Zoom out so that you can see the whole building and the surrounding area. We will plant some trees next.



Some trees planted

Open the object browser window from the popup menu of the select window. Go to the folder models/Plants/simple and drag and drop an object called flowertree in front of the house. Then drop three camelia trees to the front yard.

Close the object browser window. The 3 camelia trees are identical, which is quite unnatural. Multi select the trees and click the Randomize tool on the tool bar. Each tree becomes unique.

Close the world level on the select window and click the root level. Then click the Camera tool on Creation tab of the toolbar. First click on the position where you want to place the camera, for example in the front yard. Click second time where you want to aim the camera - in the middle of the house. The third click defines the opening angle of the camera.

Create a second camera inside the house, for example into the kitchen. Take a front view. Move the cameras upwards to a natural height, say 1.5 meters from the ground level.

Drag and drop the first camera to the view window. The view takes the projection parameters from the dropped camera object. Click the render button, which is located in the lowest icon group of the view control bar. Note that you can scroll the bar with the mouse wheel in case the button is hidden below the visible area. The last step is to save the finished project.



Ray traced image of the house and its environment



Note

You can save the contents of the view to an image file using the view popup menu Render/Save to File. To create print resolution images, use the File/Render pull down menu.

Project Harry

Introduction



In this tutorial, we will model, texture and animate a funny little character called Harry. The tutorial is a complete example and includes all usual steps of making an animation. Although the subject is quite advanced, the workflow is described in detail. Even new users who are already familiar with 3D graphics can try out this tutorial. If you find a certain part of the tutorial difficult, you should examine other beginner level tutorials of the same issue and continue again with improved skills.

Tutorial level: Advanced

Example project: tutorprojects/harry/finished.r3d

Modeling the body

The model will be built using subdivision surface modeling. This technique allows stepwise refinement of a rough control cage into a detailed model. It is a very powerful way to build organic freeform shapes.

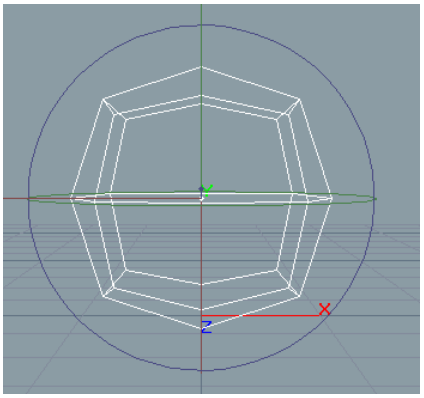
First some terminology: the left mouse button will be abbreviated as LMB, the right button as RMB. Moving the mouse while holding a mouse button down is called dragging. 'Click' means LMB click unless specified otherwise. Similarly, dragging means LMB dragging.

Start Realsoft 3D. You may close the animation bar on the bottom of the screen by hitting the function key F7. This maximizes working space on the view window.

The initial control cage for Harry's body is a simple cube. Switch to the leftmost tab (SDS) of the toolbar. Click on the Cube button to activate the SDS cube tool.

 *SDS cube tool*

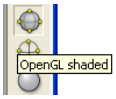
Move the mouse over the blue View window. Click once to define one corner of the cube. Press Shift key down to force a symmetric shape (width=height=depth) and move the mouse diagonally away from the first point.



SDS object created

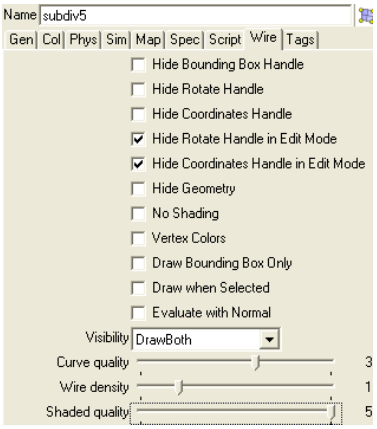
When you see a shape that has a suitable size for the body, click second time. The simple, somewhat rounded shape is the surface generated by a cube like control cage.

Let's modify our modeling environment a bit. To get a shaded real time view of the shape, click the OpenGL rendering icon. It is the fourth icon from the bottom in the vertical view control bar, located at the right side of the view.



The shaded SDS object may look a bit blocky. To see the actual smooth surface better, double click the object name 'subdivXX' on the leftmost Select window. A quick double click opens the object property sheet (you can also hit a lowercase 'p' hotkey).

Switch to the 'Wire' tab of the opened window. There you can find a slider 'Shaded Quality'. Set it to value 4 or 5, and then close the property window.



Property Window

In shaded OpenGL modeling, light sources usually help to visualize the shape. Let's add one. Zoom out the view window a bit by holding down both the Shift key and the right mouse button and by moving the mouse downwards. You will now see a larger area of space. Similarly, RMB dragging with Alt rotates the view, and RMB with Ctrl key pans the view.

Fluent space navigation is an important part of modeling in every 3D software package, so it might be a good idea to practice these controls until you feel comfortable with them. If you get lost in space, just hit the blueish globe icon in the view control bar. It brings you back to the front view towards space origin.

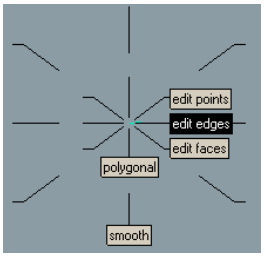


Point light

Then back to the light source issue: the toolbar on the top includes a tab called 'Light Sources'. Bring it to the front. Click the 'Point' light icon. Then click on the view window, for example on the top left corner. Move the mouse a bit and click second time to finish the tool.

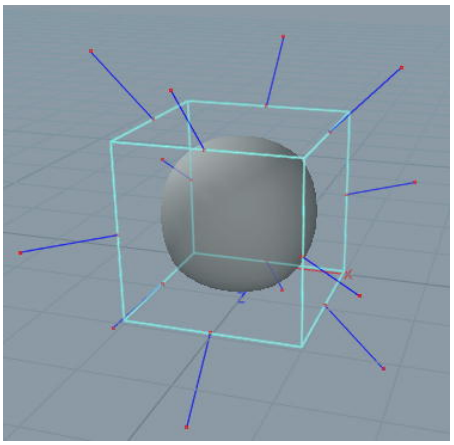
After these preliminary steps, we can start the actual control cage editing. Select the SDS object, which we created above, from the leftmost select window by clicking its name (SubdivXX).

We can quickly set the object into so-called edit modes using the View window's compass menu.



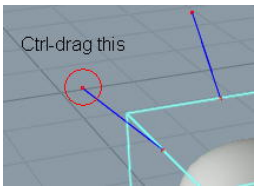
So, move the mouse over the View window, press the right mouse button down and move the mouse. This opens the context sensitive compass menu. When a SDS object is selected, the compass includes among other things 3 edit modes: edit points, edit edges and edit faces. Move the mouse to the right until 'edit edges' becomes highlighted. Then release the right mouse button.

The idea behind the compass menu is that, because it is based on clearly distinctive directions and distances, your hand can learn the selections and make them very quickly. There's no need to aim slowly and precisely.

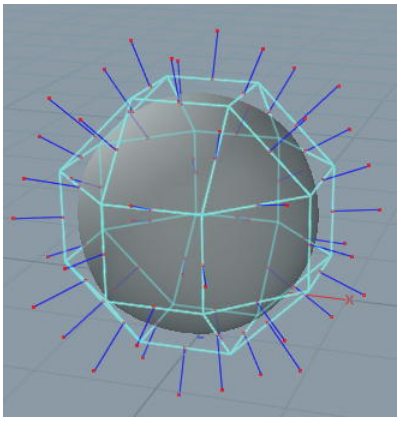


Edges selected

Now the SDS object is in edge edit state. You can see its control cage as a wire frame around the shaded surface. If you move the pointer over the shape, you can see how edge handles become highlighted. Select all edge handles by pressing LMB down and dragging a selection box around the whole SDS object. Selected edges display a blue edge normal handle line. Make sure all the edges are selected. If not, you can add them to selection by Shift clicking them one by one.



Press the Ctrl key down. Move the pointer over a red knob at the outer end of one of the blue edge normal handles. Press LMB down and drag the knob outwards, Ctrl held down. Ctrl dragging an end knob performs the edge bevel operation. Release the LMB when the bevels cut the object into equally sized parts as shown in the figure on the right.



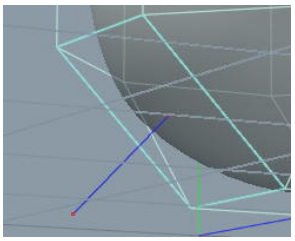
Edges beveled



Note

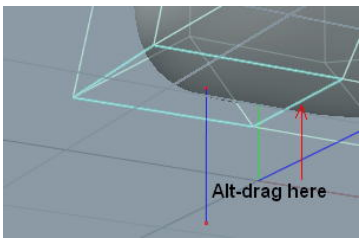
If the operation fails (not all edges were selected etc.), just apply Undo (Ctrl-z) to step back in modeling history.

Now switch to 'Edit Faces' mode; you can do this with the compass menu. Rotate the view (Alt+ RMB drag) so that you see the object from below. Note that the grid on the ground plane helps to visualize rotations. Select a face at the bottom left side by clicking it. This face will be the starting point of a leg.



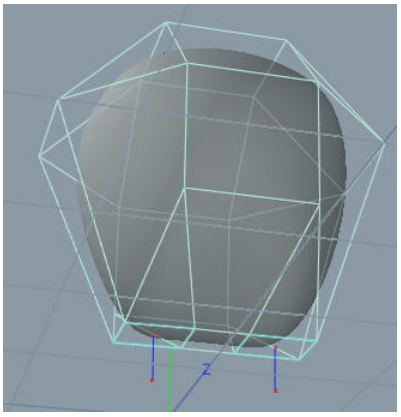
Face selected

Press the Alt key down, move the mouse over the bottom edge of the selected face and LMB drag. This activates rotation of the face around the edge. When the face is fully horizontal, release LMB.



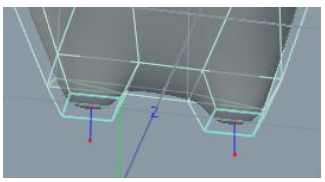
Click the face at the bottom right side to change the face handle selection and rotate it to the horizontal plane, too. Then, press the Shift key down and click the face, which you already rotated. Shift clicking does multi-selection and now you have both faces selected. If you accidentally select wrong faces, you can Alt click them to deselect.

As already indicated, we will 'branch out' two legs from these faces. Before doing that, you may decrease the size of the faces slightly, to make the leg diameter smaller. This happens by LMB dragging the end knob of either of the face normal handles. Note that if you hit a wrong handle, just hit esc key to cancel, or Undo (Ctrl z) afterwards. You may also move the faces closer to each other, by dragging an appropriate face edge. This moves the face in the edge direction. After these adjustments, make sure both faces are selected.

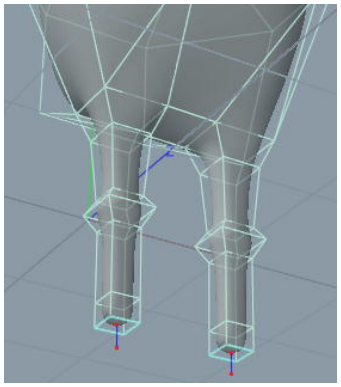


At this point, it is a good idea to save the project. Undo has a limited depth, and you may later want to reload a certain work phase. So, select 'File/ Save As' from the pull down menu, type in a file name and hit OK to save. Remember to repeat saving at regular intervals. It is also a good idea to change the file name occasionally so that you will get several backup versions: harry1, harry2, ... etc.

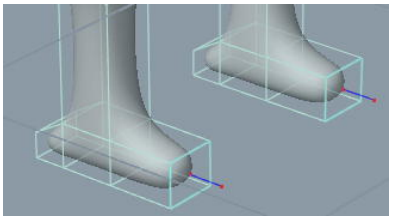
Now we start leg branching. Press Ctrl key down and LMB-drag either of the blue face normal handles from the middle of the line downwards. Two extrusions appear. Leave the first extrusion quite short, so that there will be enough geometry detail to allow leg bending at pelvis area. Release LMB to finish extrusion. Then drag an outer end knob of a blue normal handle line to shrink the leg diameter slightly.



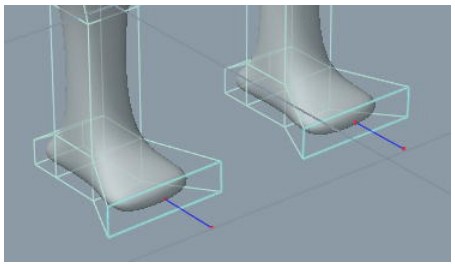
Just keep both end faces of leg extrusions selected and repeat extrusions and diameter adjustments until you reach the heel of legs. Put some extra cross-sections into the knee area, to allow knee bending and better control of surface detail. Feel free to use your imagination here! Learning the keyboard modifiers, such as the Ctrl-drag for extrusion, takes some practicing before it feels easy, but once you learn it, you can change the model this way efficiently. The handles are loaded with all kind of useful functionality, which you can access very quickly.



Select the faces behind the leg ends and extrude the heels (Ctrl-drag a normal handle) backwards. Then extrude first foot sections using the two opposite faces. Remember that you can build both legs at the same time, just by keeping the respective face handle selections on both legs.

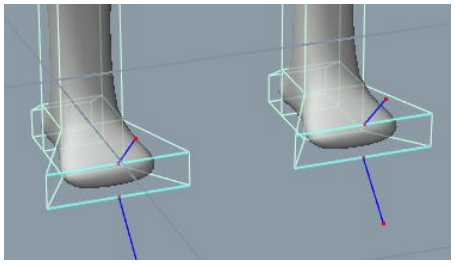


Increase the foot diameter by dragging the end knob of a face handle. Then make the faces flatter (lower) by Shift-dragging a vertical edge. To summarize edge drag operations we have used so far: plain dragging moves, shift dragging scales and alt-dragging rotates.

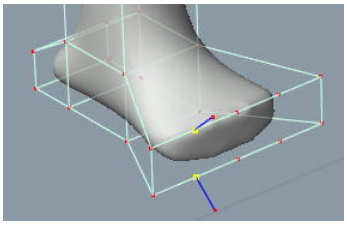


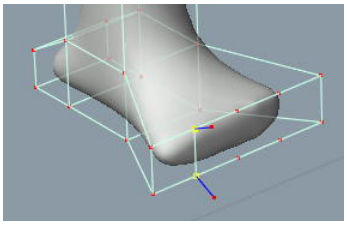
Now its time to extrude some toes. Use the compass menu to switch to the edge edit mode. Click an edge at the top of a foot extrusion. Then Shift click the remaining 3 horizontal edges at the end of the foot extrusions, to add them to the selection.

The second row of the toolbar contains a tool icon Split. Press the icon. Enter 3 to the Splits field and click Accept.



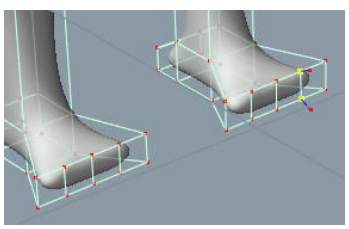
Use the compass menu to activate edit points. Click the first split point on the top edge of the left foot to select it, then Shift click the corresponding point below. Now a point pair is selected and you can apply the Add tool of the second toolbar row to create a vertical connecting edge.



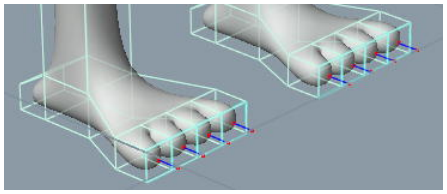


Select the next point pair and apply Add. Do this until all edge point pairs are connected. These steps prepare 4 new faces per foot, each one suitable for branching out a toe for Harry.

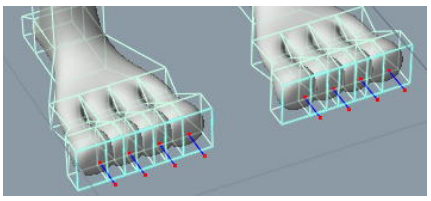
Open the view compass and select Edit faces mode. Click the leftmost base face of a toe to select it. Then shift click the remaining 7 toe faces to multi select them all. Next we will extrude toes.



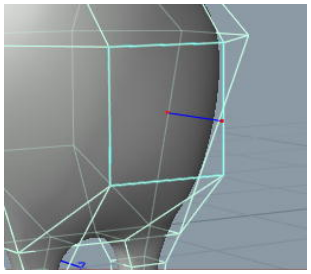
Since we are extruding connected faces, we need to specify some additional options. So, instead of Ctrl-dragging face normals, activate the Extrude tool from the second row of the toolbar. The toolbar changes to show the extrude tool options. Set the Region selector to Separately. Click anywhere on the view window, move the mouse and click again when the toe extrusion length is about one half of the full toe.

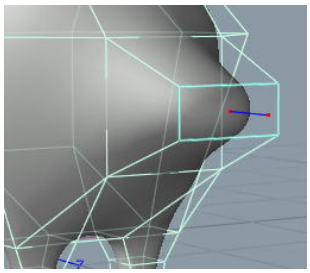


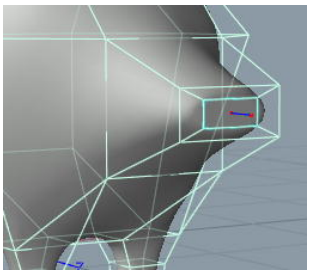
Shrink toe diameters using a red end knob of a face normal, and then extrude a new thicker section out. Again, you can build all 8 toes at the same time. If toes start to overlap, you can move them apart by dragging horizontal edges of selected faces. The final foot is shown in the example image.

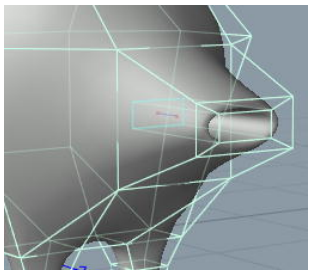


Modeling the beak and mouth takes some simple extrusions as shown in the pictures below. Use the normal handle with the Ctrl modifier to perform them easily. Note that the extrude operation works also inwards - you can create the mouth with it.

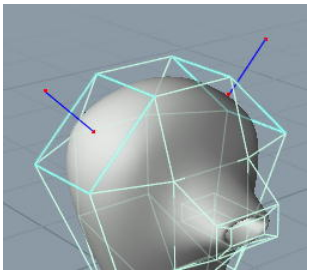




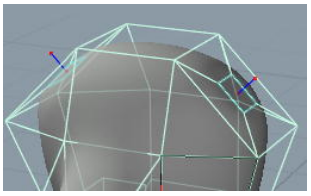




The figures show a very simple mouth construction, but feel free to put some additional geometry detail to the mouth area. This can be done, for example, by splitting the mouth edges and connecting the new points to the surrounding vertices. The more geometry there is, the more flexible expressions the mouth can show.



The two 'horns' or 'eye arms' are extruded from the head. First select the two faces on both sides of the head as the image on the left shows.

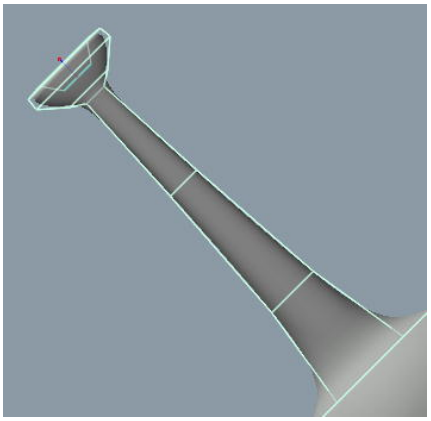


Then press the Ctrl key down and drag the outer end knob of a face normal handle.

Ctrl dragging the end knob does something similar to the extrude operation, except that now the new face appears inside the original one. This way you can set up a smaller area for the actual horn branching.

The remaining modeling steps for horns are just applying extrusion and diameter scaling repeatedly - see the leg modeling instructions above. Add enough cross sections to horns so that you can later bend them in a flexible way and set up interesting poses. One cross section in the middle already allows bending, but with 2 or 3 cross sections more you can achieve multiple 'S' like bends.

At the end of the horns, you should extrude little cup like nests for the eyeballs.



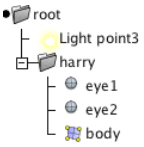
Now the actual body mesh is ready. Fine tune the shape using point, edge and face handles if you like. Then turn edit mode off by selecting 'object' editing from the compass menu (it's the compass selection straight to the left). You can also hit the spacebar to toggle edit mode on/off.

To model the eyeballs, switch to the Analytic tab of the toolbar and activate the Sphere tool. Analytic sphere is a very simple and efficient way to model the eyes.

The Analytic Sphere tool

Hit the globe icon in the view control bar at the right edge of the screen to get a straight front view. Click in the middle position of the left eyeball. Then move the mouse, and click again when the size is suitable. Then select 'Edit/Duplicate' from the pull down menu, or press Ctrl + d to make a duplicate sphere. Hit 'm' key to activate the move tool. Click the first eye to start moving, move the mouse until the duplicate sphere is located in the proper position in the second horn and click again to finish moving. Take a top view by hitting the camera like icon above the globe like reset icon in the view control bar. Make sure that the eyeballs are in the middle of the horn also in Z direction. If not, apply the move tool again. Now the eyes are ready.

Let's clean up the hierarchy a bit. Select the body mesh called subdivXX, click the RMB and select Rename from the popup menu. Change the name to 'body' and hit Enter. Select the first sphere and rename it as 'eye1', then rename the second sphere as 'eye2'. Multi-select all three objects (for example: first click the body, then Shift-click eye1 and eye2 on the select window), click the RMB and select the tool Drop to a level from the popup menu. This creates a new hierarchy level, which includes the three objects. Rename the level as 'Harry'. Now we have a clean, simple structure for the character with easily recognizable component names. Organizing and naming the scene parts carefully is a useful modeling practice, and essential in managing complex scenes.



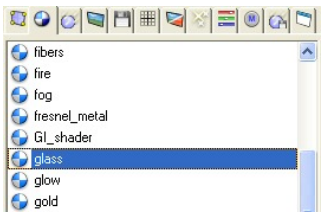
The body is now ready. Save the project again.

Adding Materials

The next step is to apply some materials to the character. You can continue from the previous chapter or start from scratch by loading the example project 'tutorprojects/harry/body.r3d'.

Select both spheres (eye1 and eye2) from the select window; you can do this by dragging a selection box around the two object names, or by Shift clicking them. Keyboard modifiers work on the select window exactly the same way as in the view window. You can also click the spheres in the view window to perform the selection there, although it can be harder there if other objects overlap the desired targets. Note how objects become highlighted at selection.

Switch to the second tab of the select window, which contains a list of materials. By default, material names are displayed, but you can also get their preview icons by selecting 'Show Previews' from the popup menu. Drag and drop a material called 'glass', from the select window into the view window. This assigns the dropped material to the selected objects. Note that object selection, not drop position, defines the targets for material assignment.

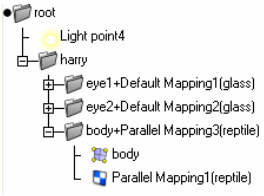


If glass is not in the library, select New/From Template from the popup menu and use the file browser to find and load it in (file 'materials/combined/glass').

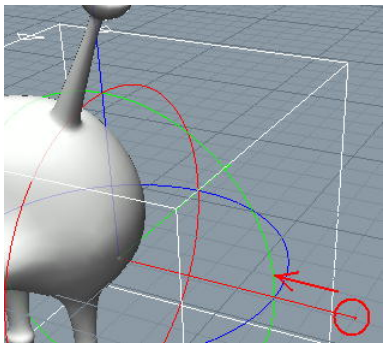
Click the body mesh on the view window to select it. Make sure that the mesh is no longer in Edit mode. If it is, hit the spacebar to turn edit mode off. This is important because dropping a material to a SDS object in edit state may apply the material only locally. Drag and drop a material 'reptile' to assign a bump map to the body. The assigned material shows up as a wireframe cube around the body.

Let's do a test render. Rotate (Alt-RMB drag) and zoom (Shift-RMB drag) the view to find an appropriate viewing angle. Then click the ray trace icon (third from the bottom right corner of the screen) or hit the zero hotkey '0'.

Most likely the bump map pattern, which makes the skin wrinkled, needs some scale adjustments. Perhaps the bumps are too big? Switch to the leftmost geometry tab of the select window. The body mesh has been dropped into a new level with a parallel map object. Select the parallel mapping - it is the object, which attaches the reptile shader to the body geometry.



On the view window, move the mouse pointer on top of the little scale knob at the end of red axis line (other knobs will do as well). Press Shift key down and drag the knob towards the handle center. Release LMB when the map geometry has shrunk sufficiently.



Render the view. Now bumps are smaller. However, they appear much steeper because the height did not change in density scaling. To adjust bump height, switch back to the material library on the select window, double click the Reptile material to open the property window and change the Bump Height value (default 0.01 m = 1 cm) to a smaller value, such as 0.002 = 2 mm. So, type 0.002 in the Bump height box and hit Enter. You can now close the property window.

Now bumpiness is OK but there is one problem: bump pattern is tied to the body geometry along a rigid parameter grid, defined by the parallel map box. The simple box cannot follow when Harry for example bends its leg. Therefore, we must convert parallel map based pattern to a pattern that is fixed to the actual vertices defining the body.

There is a tool for this purpose. On the geometry tab of the select window, select both the body mesh and the parallel mapping. Go to the Materials tab of the toolbar and activate Map2Obj tool. The default options, Target=Points, Source =Map Coords and Destination=UV Coords are suitable. This will evaluate and store parallel map coordinates into every point of the SDS mesh. Hit Accept.

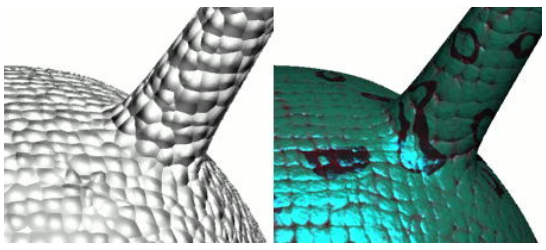
Now you can delete the parallel map object (use Delete hotkey or Edit/Delete pull down menu). Then select the parent level of the body mesh. Switch to the materials tab ob the select window. Make sure that Reptile is still highlighted, and select Map/Default from the popup menu. This creates a new kind of material map, which uses UV coordinates. UV coordinate values are stored into vertices, so they will not change when the model deforms.

Save the project.

Programming the Skin with VSL

The reptile skin material is now attached to the body. It defines only bumpiness, no color patterns or other shading information. Let's make the skin material more interesting:

- More variations to bumps
- Color patterns
- Metallic shine
- Efficient antialiasing



Original and improved skin

The example project to start this chapter is 'tutorprojects/harry/body_and_materials.r3d'. You can also load the improved materials from the example project 'body_and_skin.r3d' if you wish to skip this VSL oriented chapter.

Surface Geometry

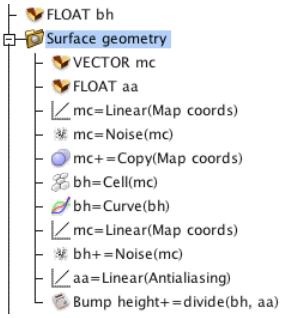
Switch to the material tab of the select window, make sure the reptile material is selected and open the property window. Check the Advanced option to see the structure of the material. It is very simple:

```
Surface geometry
    Bump height += Cell (Map coords)
```

A VSL material consist of top level blocks called shaders: Surface geometry, Surface properties etc. Shaders define what happens in each step of the rendering pipeline. As the name suggests, a Surface geometry shader defines geometrical modifications for target surfaces. Although you can add shaders into a material in an arbitrary order, shaders have a fixed, natural order of evaluation. For example, a Surface geometry shader is always evaluated before a Surface illumination shader. Within shader levels, the evaluation order is the order of the objects in the VSL tree.

The first step is to make bumps more interesting. A surface geometry shader that does this looks as follows:

Below is a description how to build the shader. Creating a finished shader from scratch may look like a complicated task especially for users with no programming experience. However, in a real modeling situation, such shaders are created one small step at a time. For example, you could first solve how to control the bump shape only. While adjusting the shader and rendering previews of it, that part of the shader becomes very familiar to you and it is easier to proceed into the next part.



Let's start: drop a Variable object to the root level of the material. You can add VSL objects either by selecting the target level in the VSL hierarchy and choosing a VSL object from the popup menu, or by drag and dropping VSL object icons from the VSL Objects frame. The latter alternative has the advantage that you can place the object exactly at the desired location. Popup menu adds new objects to the end of the selected level, and you often have to drag the new item to the right place.

The new VSL object - Variable - becomes automatically selected. The bottom part of the VSL window shows a property gadget for it. Set the Type of variable to Float and rename it as 'bh'. VSL supports 3 kinds of variables: one dimensional float and 3 dimensional color and vector variables. The rendering system handles type conversions automatically, but it is wise to use the most appropriate type. For example, when you assign a value to color variable, VSL editor shows you a RGB color selector instead of a numeric vector gadget. Bump height is clearly a one dimensional value, so we use a float variable here.

This root level variable 'bh' will store bump height information and pass it to the Surface properties shader - we will use it there to define color from the bumpiness. Root variables are visible in all shaders, and therefore they can be used to pass information between shaders. Adding a new channel is another alternative, but this is a lighter, only locally visible alternative. One must be careful when passing information in root variables. Because of ray trace recursion, the material can be evaluated several times between shading steps. For example, do not expect that a root variable initialized in Surface properties still contains the same value in Surface illumination. Shadow computation may evaluate the same material on another position to find out object's transparency. The method can be used here because there are no shading events between Surface geometry and Surface properties shaders in our scene.

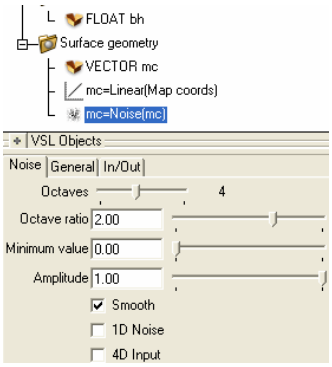
The original bumps were quite regular. A standard solution to make a pattern irregular is to add some noise to the parameter channel. The channel that contains the 3D space related parameter (suitably warped by a geometry of a material map object such as a parallel or a spherical map) is Map coords. Its original value may be needed later, and therefore we compute the modified parameter and store the result to a local variable. So, drop a new

variable object to the beginning of the Surface geometry shader. Set the type to Vector and rename it as 'mc'. Add a Linear object after the variable, set its Input0=Surface:Map coords and Output=mc (you can change Input and Output from the popup menu of the shader tree gadget, or using the In/Out tab at the bottom of the VSL window). Set the Multiply value of the Linear object to (0.5 0.5 0.5) - you can find the mentioned value from the Linear tab of the material property window. This value controls the density of bump shape irregularities. Density should be in a suitable proportion to the bump size. If the noise is clearly denser as the skin cell pattern, individual cells get distorted. If the noise density is lower, cells keep their original shape but their positions change and dimensions twist along the noise curvature.



Original skin, distortion by a low density noise and distortion by a high density noise

Add a Noise object after the Linear object, set Input=mc, Output=mc, Octave ratio=2, Smooth option enabled and Amplitude=0.1. Amplitude is the most important value here - it defines how large the bump shape irregularities become. You can find suitable values by experimenting, but 0.1 is a good starting point. The Octaves value of the noise controls how much detail the noise has. If you plan to make an animation, which shows the skin from a very close distance, you should use a high number of octaves so that the magnified skin shows enough details. For this example, 3 or 4 octaves is enough.



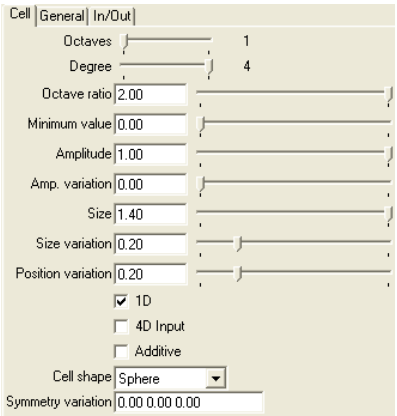
Noise properties

Now mc contains noise values for the bump shape distortion. To get the original parameterization included, add a Copy object after the Noise object, set Destination to mc from the popup menu, Source to Surface:Map coords and Operation from the General tab of the property window to +.

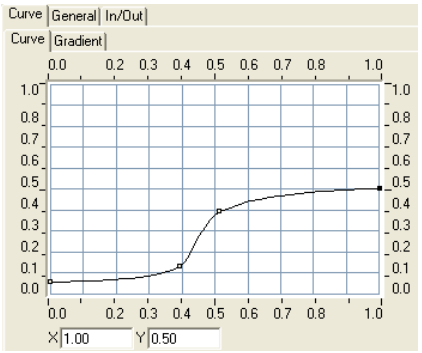
Now mc is a sum of irregular noise and regular Map coords. Using pure noise to define bumps would create a totally distorted bump pattern. Therefore we will use a sum of noise and Map coords. To summarize the computation with some clarifying comments:

```
mc=Linear(Map coords) // Define noise density as mc = 0.5 * Map coords
mc=Noise(mc) // Compute noise value.
mc+=Copy(Map coords) // Add regular parameter
```

Next activate the Cell object, which is already in the shading tree. It should be the last object in the Surface geometry shader. If not, drag it to the end of the object list. Change its Input0 to mc and Output to bh. Change Base value to 0 and amplitude to 1. We will modify the bump shape with a curve object, and the 0...1 range is a suitable parameter interval for it. In the General tab, change the operation from '+' to '='. The Cell object now initializes the new bh variable instead of modifying the actual Bump height channel. Other Cell settings can be as in the figure below.



We can adjust the shape of bumps with a curve object. The purpose is to define grooves between relatively flat bumps. Add a Curve object after the Cell object and set Input0=bh and Output=bh. At the Curve tab, right click on the curve and from the pop-up select 'set minimum and maximum values'. This opens a small window where you can enter minimum and maximum values for the X and Y axis of the curve. The default Curve scale, 0-1 meters, is definitely not suitable for skin details. Set Max Y = 0.01 meters in the opened dialog, set Rescale and hit O.K. Then edit the curve as shown in the picture below.



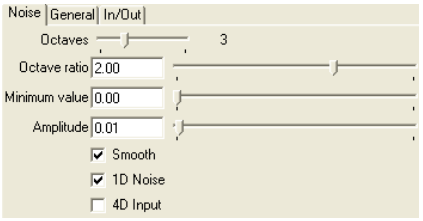
If you want to preview the bump pattern now, go to the In/Out tab and change the Output of the Curve to 'Surface:Bump height' and render. When the result is OK, change Output back to bh to continue the example. The curve controls the profile of bumps, and you can change it here easily and intuitively. Just note that the curve defines only one half of the bump, from the edge to the middle point.



The bumps have now somewhat irregular shape, but the surface does not have very fine detail. The bumps are actually quite smooth. The standard solution is to add a suitably dense noise field. Because the fine details must be clearly denser as 'large scale' bumps, we need again a modified parameter. So, drop a new Linear object to the end of the Surface geometry shader and set its Output to mc. We use the already defined mc variable here again, because its old value is no longer needed. At the linear tab, set Multiply to 5 5 5. This will give roughly 5 times denser bump pattern as the base pattern consisting of cells.

Now it's time to save the project before proceeding into new experiments.

Add a new Noise object to the end of the shader. This is very fine detail noise, so set Amplitude to 0.002 (2 millimeters) by typing 0.002 in the Amplitude box. Activate the Smooth option (there are no sharp edges on the skin) and set the 1D Noise option, because Bump height is a one-dimensional value. 1D noise is faster to compute as 3D or 4D noise. Set Input0 to mc, Output to bh and Operation from the General tab to '+'.

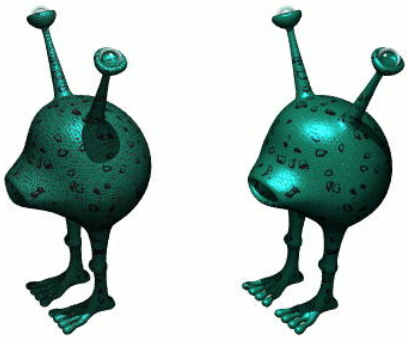


The final step is to move the computed bh value to the actual Bump height channel. We do this in an intelligent way, which improves antialiasing quality efficiently.

First, drop a new variable to the Surface geometry shader. You can place it to the beginning of the shader to make the VLS code nicely structured. Change the type of the variable to Float and rename it, for example, as 'aa'. Then add a new Linear object to the end of the shader, set Input0=Surface:Antialiasing and Output=aa. The Multiply value of the Linear object should be relatively high, 500 or so. The value of the Add field should be 1.

The last VSL object is an Operation object - add it to the end. Set its Type to Divide. Input0 should be bh, Input1 = aa and Output = Surface:Bump height. In the General tab, change the assignment Operation from '=' to '+'. It is a good practice to use + operation when modifying the bump height channel. All materials coded like this can be multi mapped to a same target object and bump maps become automatically summed up.

Let's consider the last VSL line a bit more: the computed bump height value is divided by an antialiasing factor, which describes 'information density' around the examined point. The density is very high if you view the object from very far away or in a low angle. When we divide the bump height with a high-density factor, bumps will gradually fade away. This is exactly what we need: small specular highlights from bump peaks, dark grooves and other details which would cause distracting flickering in animations unless a very high ray trace sampling rate were used, now automatically blend towards a nicely behaving smooth surface.



No antialiasing on the left, shader based antialiasing on the right

The Surface geometry shader is now ready; make some test renders and play with the Multiply value of the Linear object which scales Antialiasing, to find a suitable value, which does not reduce orthogonal bumps close to the camera too much.



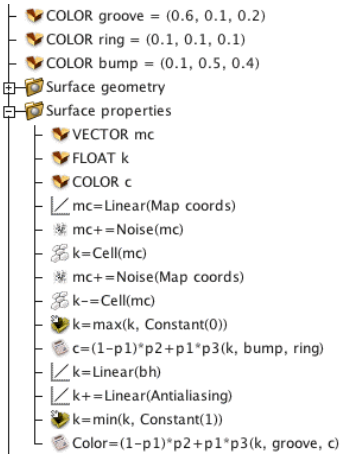
Note

If you intend to use displacement mapping with these antialiased bumps, remember to turn antialiasing off by setting Multiply=0 for the Linear VSL object which scales antialiasing channel. The shader computes a different bump height value for a camera view and for light sources. In other words: light sources see the object shape in another way as the camera does. Inconsistent geometry definition by surface displacement will create strange shadows, which almost certainly spoil the image.

Note for advanced users: the antialiasing channel has a 3 dimensional value. The third Z component contains a base value which depends on camera projection and focal length, distance from the examined point and the surface angle against the viewing ray. The first two sub components X and Y incorporate the contribution of the material mapping geometry. Shrinking a parallel map dimensions to one half doubles the antialiasing X and Y factors. This option is useful for example when mapping bitmap textures to surfaces. In our example, parameter is defined using a Default map object, which simply moves the value of UV coords (stored in vertices) to the Map coords channel. The Default map has no 3D space geometry of its own, and therefore it sets all three sub values of the antialiasing channel to equal. In the example above, we could use any of the sub channels; now the first X component is used because of the type conversion rules of VSL.

Surface Properties

Next we will define some color patterns for the skin. The goal is: bumps have a certain base color and the grooves between them have a different color. In addition to this, bumps have some large dark irregular rings on them. Rings do not color grooves.



At the top of the VSL hierarchy, click the minus sign to collapse the now finished Surface geometry level to get more display space for new properties. Then, at the root level of the material, add three variables, and rename them to Groove, Bump and Ring. For each Variable, object check the Initialize option and set the color value as you find suitable.

In the example scene, grooves are dark bluish red (0.6 0.1 0.2). Bumps are turquoise green (0.1 0.5 0.4) and rings are almost black (0.1 0.1 0.1). Initializing the values into root variables is an efficient solution: it is done only once at the beginning of rendering, not in every examined point. You can also initialize the values efficiently using Constant objects in a Material Initialization shader (bump = Constant(0.1 0.5 0.4) etc.).

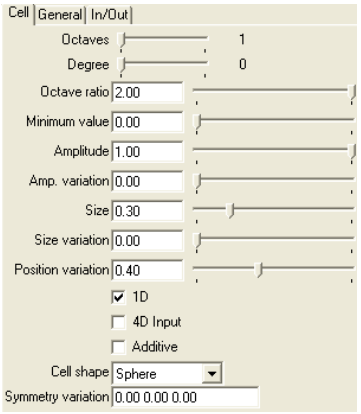
Add a new shader object to the material. Its default type, Surface properties, is just what we need. First we compute a value, which adds the rings. Add two variables to the beginning of the Surface properties shader. The first one is

a vector variable 'mc' and the second is a float variable 'k'. Set the types and names as before. The variable mc is used the same way as in the Surface geometry shader: we compute a noise distorted parameter into it, so that we can generate irregular patterns. Computation goes now as:

```
mc = Linear(Map coords)
mc += Noise(mc)
```

So drop a Linear and a Noise object to the Surface properties shader, and set Input and Output channels as shown above. The Multiply factor of the Linear object should be now much smaller, for example (0.2 0.2 0.2). The variable mc will be used for ring patterns, which are much larger as the skin bumps. When we multiply the Map coords with a number smaller than one, the product changes slower in 3D space and hence the patterns become larger. Noise Amplitude can be for example 0.07. The bigger the value, the more irregular rings you will get. Change also the assignment operation of the Noise from = to + in the General tab.

Next add a Cell object to the end of the Surface properties shader. It defines where rings appear. Set Output=k and Input0=mc. Cell options are shown in the image below. It is important that Base value is 0 and amplitude is 1, and amplitude variation is 0 - we will use k variable to 'key' the black rings to the skin. Also, cell Degree must be 0 so that the output value jumps straight up to 1 from the base value 0. A higher degree would produce gradient-edged rings. Set also 1D option for faster computation. Size variation can be 0.



The Cell object now produces spots, not rings. Subtracting a slightly modified smaller Cell signal turns spots into rings. To modify the inner spot shape, we add more noise to mc. So, drop a new Noise object to the end of Surface properties. Set its Output to mc and Operation to +:

```
mc += Noise(Map coords)
```

Noise Amplitude must be quite small, for example 0.002 - otherwise rings will easily break. Press Ctrl key down and drag the already added Cell object to the end of the Surface properties shader. This creates a duplicate of the Cell object. Decrease the Size from 0.3 to 0.2. Change the Operation from = to -.

To ensure that the key value k does not get negative, drop a Constant object to the end of the Surface properties shader. Set its Output to k. Change the Operation to max:

```
k = max(k, Constant(0))
```

Now the key value is set. We need one more color variable. Drop a Variable object to the beginning of the Surface properties shader and rename it as 'c' (you can also use longer names if you like). Add an Operation object to the end of the Surface properties shader. Set the Type of the operation to the formula labeled as (1-p1)*p2 + p1*p3. This cryptic looking formula actually does simple linear interpolation. With such an operation, you can blend a color over another color using 'alpha key'. Set Output to the color variable c, Input0 to k, Input1 to bump and Input2 to ring. Remember that the two last input parameters were the colors defined in the root level.

Finally we will key color c - combination of bump and ring colors - with the groove color using the bump height as a parameter. In the Surface geometry shader, we stored non-antialiased bump height to a root level variable bh. Drop a new Linear object to the end, set its Output to k and Input0 to bh:

```
k = Linear(bh)
```

Because the bump height was maximally about 0.005 meters, the Multiply value of the Linear object should be 200. That scaling gives us the suitable keying range 0...1.

We can antialiase also the color pattern efficiently within the shader, just like we did with the bump height. The idea is to blend the color pattern towards the average color. Especially fine details, such as the thin grooves between bumps, need careful antialiasing. We can incorporate the antialiasing step easily into the final color interpolation as follows: drop a new Linear object to the end of the Surface properties shader. Set Output=k, Input0=Surface:Antialiasing and Multiply=100. The Multiply value may need some adjusting later - this is just a guess to get started. Set Operation in the General tab to +. Now the interpolation key value grows by the bump height and by the antialias channel value. Therefore, the groove color will gradually fade away at object edges or when the object gets further away from the camera.

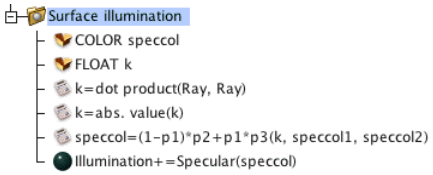
To ensure that the key value - a sum of two factors - does not exceed 1, add a Constant object to the end. Set its output to k and the Constant value to 1.0, and change the General tab's Operation to 'min':

```
k = min(k, constant(1))
```

The last shading action in Surface properties is color interpolation. Ctrl -drag a copy of the already existing Operation object, which computes c, to the end of the shader. Change the Output to Surface:Color, Input1 to groove and Input2 to c. The definition of surface's color is now ready.

Customized Specular Illumination

The third shader will control specular illumination. A metallic shine is very easy to add, so let's make the shine more interesting by controlling its color with a viewing angle dependency. Shine color will change from green to blue, depending on how we look at the surface.



Add two new root level variables to the beginning of the material, where you already placed some other root variables. Both new variables are color variables. Rename the first as speccol1 and the second speccol2. Check Initialize option for both variables and choose the color you like; in our example scene, speccol1 is light green (0.1 1 0.5) and speccol2 light blue (0.1 0.5 1).

Select Specular Color from the wizard selector at the top of the VSL window and hit the Add button. This initializes the illumination shader, which we will customize. Add two variables to the beginning of the new Surface illumination shader. Rename the first color variable as speccol. Change the second variable to type Float and rename it as k. The latter variable is again an interpolation key.

Drop a new Operation object to the Surface illumination shader, just above the last Specular object. Change the operation Type to dot product. Set Output=k, Input0=Light:Ray and Input1=Surface:Ray. Dot product is a quick way to measure angle between two directions: k=0 when rays are orthogonal, -1 or 1 when rays have opposite or parallel direction.

To ensure that the key is positive, drop another Operation object after the previous one. Set the Type to Abs.value, Input0=k and Output=k. Now k is suitable for interpolation. Add a third Operation object above the Specular object and select the interpolation formula (1-p1)*p2+p1*p3 to the Type field. Set Output to speccol, Input0 to k, Input1 to

speccol1 and Input2 to speccol2. This operation will interpolate a specular highlight color using the surface viewing angle from two base colors. The result, varying metallic shine, will look interesting especially in animations. As the final step, select the last Specular object and set its Input0 to speccol.

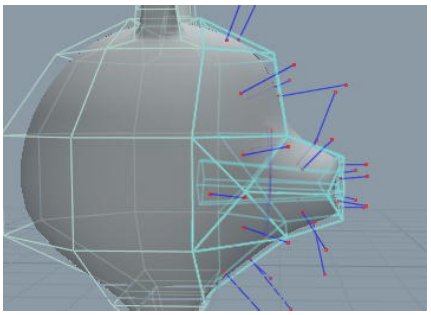
At this point, you should save the project again.

Local UV Texture

The body has now a procedurally defined color texture that covers the surface everywhere. This is a good starting point. However, details, which follow the geometry in a unique, irregular way - for example color bands around the mouth - can be hard to add procedurally. The easiest solution is to use a painted texture, which is wrapped onto the face of the character. We will do this next using so called UV Set object, which defines the texture only locally.

Start situation for this chapter: tutorprojects/harry/body_and_skin.r3d

Take a side view. Select the body mesh. In the screen shots below, we have turned off OpenGL shading of the skin textures. OpenGL visualization will not be accurate enough for estimating texture continuity. Also, when later pasting hair and animating the character, non-textured body renders much faster in OpenGL. To turn off the skin textures, open the property window, go to the Wire tab and set Texture quality to None.



Put the SDS mesh to the face edit mode. Activate all SDS faces at the face area of the body.

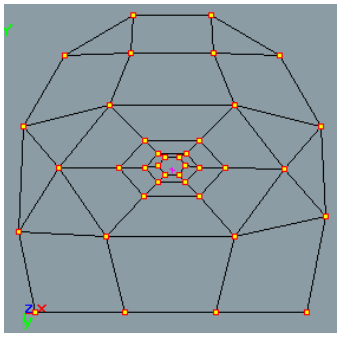
The tool control bar includes a tool called UV Set. It is at the right edge of the control bar, so you may have to Ctrl-RMB drag the control bar to find it. Click the tool. A new item 'SDS UV Set' appears to the hierarchy.



UV Set created

By default, this UV set object takes the UV coordinates from the parent SDS mesh. Those were defined using a parallel projection, which we adjusted for the reptile skin. Such UV coordinates are probably not suitable for mapping a texture image to the face.

To improve this, select the SDS UV Set and take a front view (=view against the UV set area), zoom the view so that the UV set fills the view area and click the From View tool (while SDS UV Set is selected) in the tool control bar. This tool uses the current view projection to define UV coordinates.

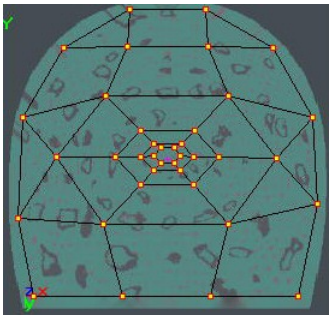


Open the UV editor from the pull down menu Windows/UV Editor. You can now see the defined UV mesh; hit the spacebar to set it to the edit mode. If the object is in face edit mode, choose Edit Points from the UV Editor's pop-up menu. Point edit vertices to adjust the result. For example, the mouth area may contain overlapping points - move them apart. The Unwrap UV tool can be helpful in fixing dense areas and folded surfaces. Close the UV editor when UV mesh is good enough.

We can 'bake' the existing color pattern to a texture map. This will make it easier to join the painted area to other skin. To initialize the texture, make sure that UV Set is still selected, go to the Materials tab of the tool bar and activate the UVImage tool. Set a suitable resolution to X and Y fields, for example 400*400 pixels. Change Type to Evaluate and change the file name to a more suitable form, such as 'facemap'. Hit Accept and wait until the evaluation is finished.

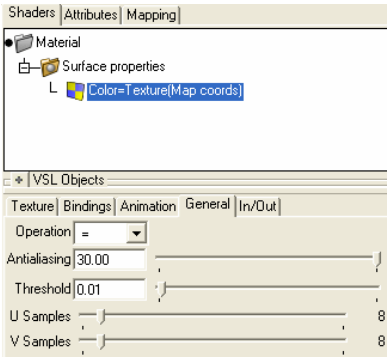


The tool adds a new material mapping under the UV Set object: Default Mapping(SDS UV Set Colormap)



In the material library, you can find the respective material, which textures the UV set area. Open the UV editor; you can see the created texture map as a background image. It may look grainy because of a low default resolution, but you can increase the quality by selecting the UV Set and by increasing the Texture quality value in the Wire tab of the property window. Then UV editor will display the background image accurately.

A test rendering shows that the edges of the new UV Set texture match the surrounding area quite well. However, the new texture mapping material does not include the same antialiasing mechanism as the procedural shaders created earlier in this tutorial. We can add antialiasing to the texture map object, too. Switch to the material library of the select window, select the new material 'SDS_UV_Set_Colormap', open the property window and check the advanced option. Select the Texture object from the shader tree and switch to the General tab. Set Antialiasing to a high value, for example 30, and the Threshold to 0.01. U and V samples can be for example 8. The higher the value, the better antialiasing you will get. 8*8 samples out of a 400*400 pixel image is a reasonable compromise between speed and quality.

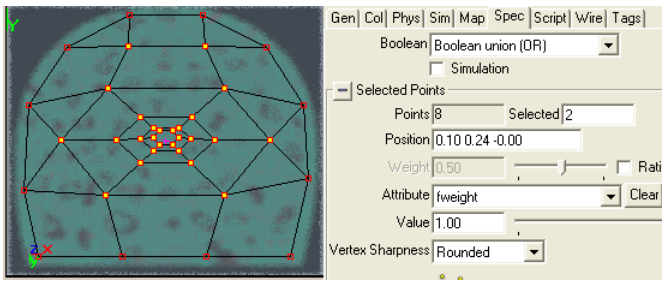


You can test antialiasing factors by making a couple of test rendering cycles. Skin patterns should appear sharp when you view the object from a close distance and fade away when you move camera further away.

Both the procedural skin and the painted skin have now antialiasing enabled. This does not ensure continuity at the border, because mechanisms are quite different. Therefore, we will next set up a gradual fading of the texture map towards its boundary. We need a new channel for that purpose. Switch to the Channels tab of the select window and create a new channel using the popup menu New/Float Channel. Rename the channel as 'fweight' or something similarly descriptive - the channel is the weight for the face texture map.

Switch back to the geometry tab of the select window. Open both the UV editor and the property window. Go to the Spec tab of the property window. Select the UV Set and put it into point edit mode using 'Edit Points' popup menu of the UV Editor. Use for example the lasso selector (hotkey '4' + LMB drag) to select only the interior points of the UV Set.

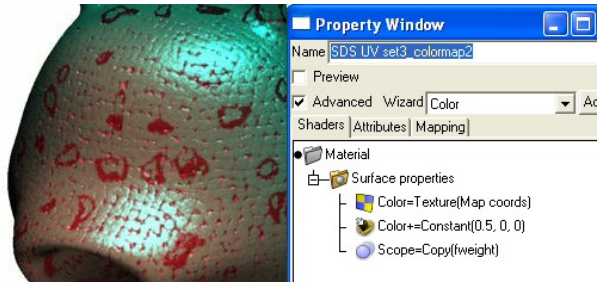
On the property window, choose fweight to the Selected Points/Attribute field. UV Set has no values assigned to this channel, so the value field shows red conflict color. Type the value 1.0 and hit enter. Now the interior has full weight, but vertices at the UV Set boundary got a default channel value 0. You can verify this by activating one of the boundary points.



Close the UV editor. The UV Set is ready, so you can turn off its wireframe using the WF-Invisible button in the toolbar or the 'i' hotkey.

Switch to the material library of the select window. The last material, SDS UV Set Colormap, should be still active and in the Advanced edit state. To enhance visibility of the blending process, add a Constant object after the Texture object, set its value to e.g. medium red (0.5 0 0) and change General/Operation to +. In other words, we make the face area redder as other parts of the body. A test render shows now a very clear boundary between the materials.

In the VSL editor, drop a Copy object to the end of the Surface properties shader. Click the In/Out tab and set Source to Surface:fweight and Destination to Surface:Scope. The blending setup is now ready - test render to see that materials blend smoothly.



The Scope channel is the system channel, which defines how strongly a material affects the target. Scope 0 makes the material invisible and Scope 1 gives a full effect. In the configuration we created above, Scope values were taken from a vertex channel, which gives us is a simple and intuitive control.

Advanced users might wonder why we did not assign the scope channel directly to the UV Set vertices but used another fweight channel to carry the value. The reason is simply the order of material evaluation: the Default Map under the UV Set is evaluated last. Its default scope 1 would overwrite the scope values from UV Set vertices. Therefore we use another, private channel to carry the values to the required shading level.

Now you can finish the face texturing. Paint some details to the created texture map (realsoft3d/facemap in the example above) in your favorite paint software, or use the 3D painting tools of Realsoft 3D. Then save the project.

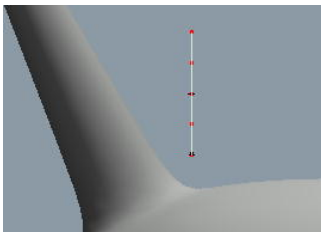
Hair

Next we will add some hair to Harry's head. You can continue the tutorial from the previous section by loading the sample file 'tutorprojects/harry/body_and_uvskin.r3d'.

We start hair modeling by drawing a single hair element: a NURBS curve.

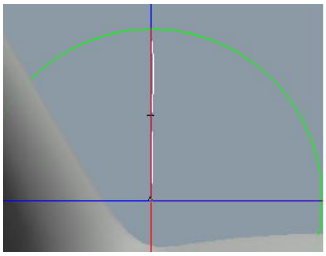
So, go to the NURBS tab of the toolbar. Activate the Curve tool and draw a single hair into the view window.

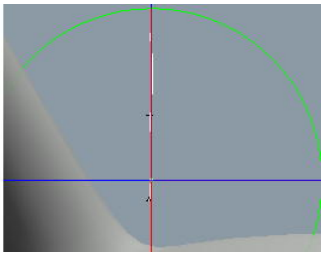
If you want to create long hair, use a sufficient amount of points so that hair bending, curly hair etc. details can be represented by the geometry. In this example, we will make relatively short hair, so, 5 or 6 points are quite enough. You can place the points at regular intervals. Position of the curve or drawing direction does not matter. Its shape should be quite a straight line - hairs will be bent and made curly later. After the last point, select Accept or hit the Enter key to finish the curve tool. Then rename the curve as Hair.



Each hair will be positioned to the skin using the pivot point - the center point of red, green and blue object handles. To ensure that hair will properly stick into the skin surface, not float above it, we have to move the pivot point along the hair.

You can do this by carefully clicking the center point of the object handles, and then a second time, say, 10% towards the opposite end of the hair. The pivot point can be moved also by activating the move tool ('m' hotkey) and selecting Target=Handles from the tool control bar.



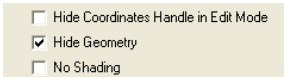




Note

The hair will be added using the Paste tool. The tool rotates the pasted objects so that their blue object space axis matches the surface normal. Therefore, make sure that the hair object you created has a correct object space orientation. The points should approximately follow the blue move handle line. A

new NURBS curve has the correct orientation after its creation, but the orientation can change if you point edit the curve heavily or apply tools, which change the orientation, such as the Normalize Object Space tool.

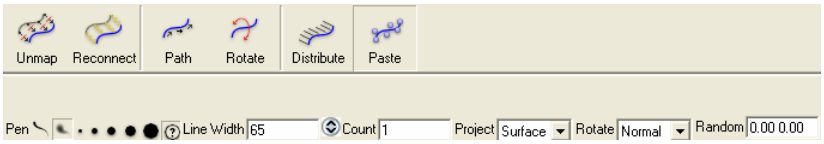


To ensure quick drawing of a large number of hair objects, select the hair curve, open the property window and go to the Wire tab. Check the Hide Geometry option and set Show Control Polygon=Always. Control polygon is the simplest way to draw the curve geometry. It is not totally accurate but good enough for our purposes. Clear the Draw Knots option.

While the curve is still selected, switch to the Spec tab of the property window. Set both Head and Tail radius to a suitable value - you may try 1 mm first. It's quite thick, but we can adjust it later. Now the hair will have a constant radius, just like human hair. Using a lower tail radius to get a tapered shape is naturally possible. Because of antialiasing related reasons, infinitely thin zero tail radius is not recommended for animated hair. It is wiser to use 'almost zero' radius such as 10 % of the Head radius value. Decrease the Quality from the default value 24 (suitable for scanline) to the minimal value 2. That is most likely sufficient and renders much quicker. Then switch to the leftmost Gen tab and clear the option Invisible in photo realistic rendering. Now you can close the property window.

Next we scatter some hundreds of hair copies on top of Harry's head. We do not have to add every single hair, but only a number, which is enough to define the hairy area, hair growth direction and other such variations. The rest will be generated by interpolating the sample hairs.

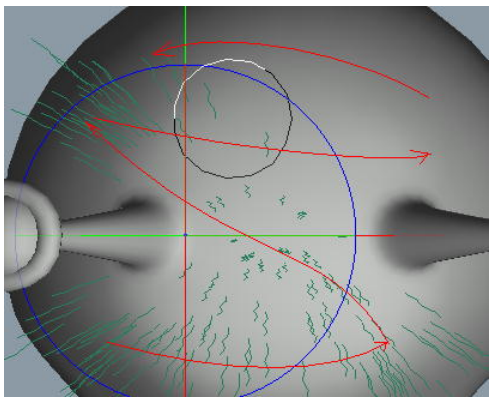
Make sure that the curve object - the sample hair - is still selected. Then take a top view over the head. Go to the Lattice mapping tab and activate the Paste tool. Select the Airbrush option, set Count = 1, Project=Surface and Rotate=Normal.



Move the mouse pointer over the view window. The brush size is shown as a circle. Adjust the diameter to approximately one quarter of the diameter of Harry's head (with the scroll wheel of the mouse or from the tool bar). Too large radius will scatter copies too distantly, making interpolation hard. Too small radius just means lot of pasting work to cover the head top.

Press LMB down to start pasting and move the mouse around at a constant speed to add hair. For an optimal result, try to fill the surface by pasting hair bands back and forth across the head. Make sure that these hair bands overlap enough; otherwise you may see visible gaps between them.

While pasting, you can release the LMB, change the view direction and continue again. If possible, continue new pasting from where you ended the previous brush strokes. You can also adjust the brush radius any time. Try to avoid brushing hair to the horns and other undesired places. However, misplaced copies do not matter much because we can relatively easily delete them later.



When a proper area of the head is covered by hair samples, click Accept to finish the tool. If you are not happy with the result, apply Undo and hit 'a' hotkey to start pasting again with the same options.

When hair pasting is done, the select window lists a large number of new objects. Curve duplicates are automatically selected and you can see their count in the top area of the select window. For a simple character like Harry, 500-1000 hair samples can be enough. A high number of hairs will unnecessarily slow down further editing and especially animation actions. Therefore, it is wise to aim at a minimal sample count.

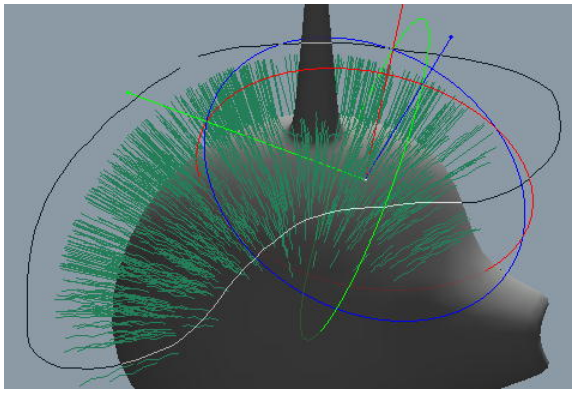
While the new hair copies are still active, select Drop to a level from the popup menu of the select window. Delete the original hair sample. Drag and drop the level containing the hairs from the root level into the level called 'harry' - hairs belong to the hierarchy of the character.

Open the hierarchy level that contains the hair curves on the select window (just click the little box at the left side of the level icon). After opening the level hierarchy, you can select individual hair objects again.

You can trim a symmetric hairline, and the same time, remove unwanted hairs as follows. Select any hair on the select window and hit Ctrl+a (or apply the pull down menu Edit/Select All). Then take a side view, zoom and pan so that you see the hairy area.

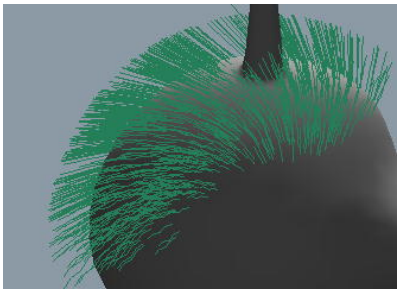
Press Ctrl key down to activate AND selection. Then press also '4' key down to activate lasso selection. Both keys held down, draw a loop that encloses the hairs you want to keep. Only the hairs inside the lasso remain selected.

Select Edit/Invert selection from the pull down menu (or use Ctrl+i) and hit the Del key. Undesired hairs are removed. If there are still some unwanted hairs left, for example around the horn and skull joining area, just zoom in heavily and click the hairs one by one and delete them by pressing the Del key.



Trimming the hairy area with the lasso selector

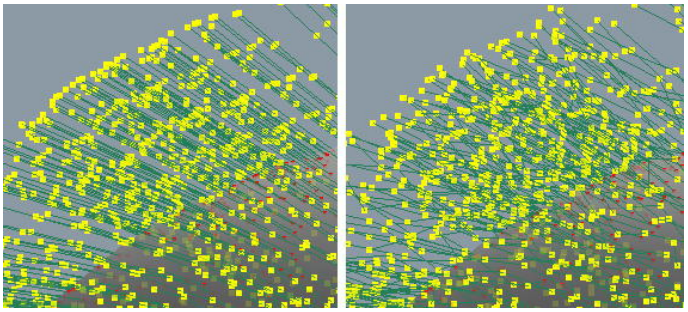
Let's experiment how to adjust hair direction and shape. Now all hairs are pointing perpendicularly away from the head. Select again all hairs like we just did above (use Ctrl+a). Take a side view, activate the rotate tool ('r' hotkey), set the About Pivot option from the toolbar, click in the middle of the head and then somewhere at the outer edge of hair. Move the mouse and observe how each hair rotates around its root. Rotate hairs so that they turn backwards and Harry gets a bit more aerodynamic haircut.



Each sample hair still has an identical shape, which is unnatural. To randomize the curves, make sure that all hair objects are still selected. Hit spacebar to put them into the point edit mode. Open the property window, go to the Spec tab, activate the Selected point field, type in 0 and hit Enter key. This selects the root point of every hair. Actually we do not want to dislocate the root point from the skin - it is the only point that must not move. So, apply the pull down menu Edit/Invert selection or press Ctrl+i. Now we have selected a suitable point set for geometry randomization.

Randomization can be applied using the Noise transform tool. Zoom in heavily to see part of hairs in detail. Go to the Transformation tab of the toolbar and activate the Noise tool. Click once in the view window; position does not matter because noise fills the whole space.

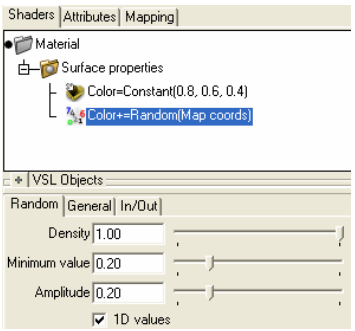
The second click position, however, is important. If you click close to the first point, the noise field will be dense and every single point moves quite randomly. If you leave more distance between the clicks - say a length of a hair or more - noise deforms hairs into recognizable turbulence patterns. After the second click, moving the mouse away from the click position increases deformation. The third click finishes the tool. Experiment freely with various distances - just hit Esc to cancel the tool and try again, or use Undo afterwards.



Randomizing hair points with the Noise transformation

So far, we have not defined any color for hair. We will now assign a unique color to each hair using a material. Hit spacebar to turn off the edit mode and select the parent level of the hairs. Go to the second (materials) tab of the select window. Select New/VSL material from the popup menu. A new material is added at the top of the list. Open the property window. Rename the new material as 'Haircolor' using the Name field. Select the 'Color' wizard and hit the Add button. A new Color field appears to the property window. Click the color square and pick a base color for the hair from the palette, for example a light brown (R=0.8 G=0.6 B=0.4). Then check the Advanced option. Click Surface properties in the shader tree to select it, open the popup menu and select the item New/Random. Set Density value of the created Random VSL object to 1.0 (to randomize everywhere), Base value to 0 and Amplitude to 0.2. Check also the 1D option if you just wish to change the darkness of hairs. 3D randomization will make hairs slightly colorful. Finally, in the General tab, change Operation from '=' to '+'. In other words, the material will make each hair a random amount lighter.

The material is now ready and you can close the property window. Make sure that in the geometry tab of the select window, the level containing the hairs is selected, and in the materials tab of the select window, the new Haircolor material is still selected. Pick Map/Parallel from the select window's popup menu and draw a parallel map box into the view window with 2 clicks. Map size and position do not matter, because the material simply picks a random value in each point.





Note

By using a Noise object instead of a Random object, you can assign hair colors that change less rapidly. With a suitable material, you can define color stripes or other patterns.

Switch back to the leftmost tab of the select window. Select the first hair object and then Edit/Select All to activate every hair object and the newly inserted parallel mapping, which is the last object in the level.



Note

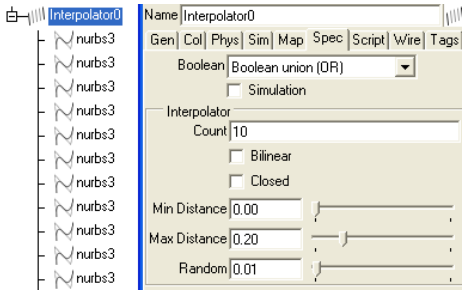
If Select All for some reason does not highlight all hair curve names, then you have left the point edit mode on, and the first curve has all points selected. If this happens, scroll down to the end of object name list and Ctrl-click the last name to range select all objects.

Go to the Materials tab of the toolbar. Activate the Map2Obj tool. Set Target=Object, Source=Color and Destination=Color. These settings will copy material defined color into object colors. Then hit Accept. Now you can find, select and delete the parallel map object from the select window. We do not need it any more because colors are now stored to hair objects.

Time to save the project! Then find a suitable view angle to hair and hit the zero hotkey '0' to render the view window. The hair looks OK but is far too sparse. We will now add the already mentioned interpolation system, which produces more copies. Once again, select all hair curves. Go to the Creation tab of the toolbar and click the Interpolator tool. A new level object called Interpolator is added to the hair level.

With the new interpolator object selected, open the property window and go to the Spec tab. The default interpolation count 10 is quite OK - it will multiply hair count by 10. Set Min distance to a small value such as 0.001. That ensures there will not be 10 overlapping hairs within one-millimeter distance. Set max distance to a value, which is about 10%-20% of the diameter of the head. A value 0.1 prevents interpolation across distances higher than 0.1 meters. The idea is to interpolate curves that are 'relatively' close to each other. Interpolating a curve on the back of the head to another one on the forehead is useless because interpolated copies would be placed inside the skull. Another example: you would not want to interpolate cat's tail hair towards a hair in its ear because interpolated copies would float in the air.

We can also add some geometry randomization here: set the random value for example to 0.01. This makes each interpolated copy unique. Do not set too high random value because hair can get dislocated from the skin. Do you remember when we positioned the pivot point of the first hair? The random value should not exceed the distance of the pivot and the first point of the curve.



Now interpolator is ready. Render a test image ('0' key) and adjust its parameters if necessary. Bilinear interpolation often gives a better result, especially if the number of used samples is low.

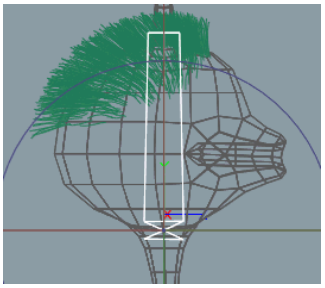
Save the project with a new file name, such as 'hairy_harry', to backup your work.

Skeleton

The next task is to bind Harry to a skeleton. The example project for starting this chapter is 'tutorprojects/harry/body_hair_skin.r3d'.

Take a side view. Set view projection to Parallel (hotkey '*' toggles the projection), because it is easier to position the skeleton without the perspective distortion. Zoom the view so that the body fills it. Placing skeletons inside shaded surfaces is often easier in wireframe mode. If your view window is still in Shaded OpenGL mode, select View Properties from the popup menu, go to Draw tab, clear the Shaded option and close the View Property Window.

Switch the tool bar to the Modifier tab. Activate the Skeleton tool. We will draw the backbone first. Because the body shape is so simple, one bone will be enough to control the body. So, LMB click first in the hip, move the mouse upwards. Swing the mouse sideways (approximately 45 degrees to both directions) to define a typical motion range of the backbone. At the start of the backbone, there is a triangular joint constraint, which shows how much the bone can rotate around its root. The triangle should point upwards. If not, move the pointer straight up from the hip and Ctrl-Click to set the constraint direction. Click second time at the top of the head to define the end point of the backbone. Then click Accept or hit Enter to finish the tool. Rename the created object as backbone.


Backbone added

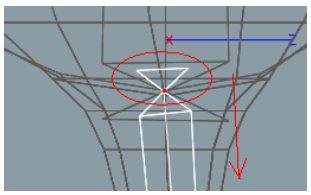
Next we will draw a skeleton for legs. Motions of bones are constrained by a set of angles. You can define the most important angle constraints intuitively already when drawing the skeleton. Therefore, it is best to draw the skeleton in a plane where it bends most. Clearly, a side view is the best view for defining how the knee bends. Take the side view and zoom the view to show the legs of the character.

Activate the skeleton tool and click first in the hip. Leg will connect there to the backbone. Then move the mouse down towards the knee. Try to avoid accidental mouse movements upwards from the hip because mouse motions now define the main angle constraint for the joint. Harry is a fat character and his legs cannot bend that high.

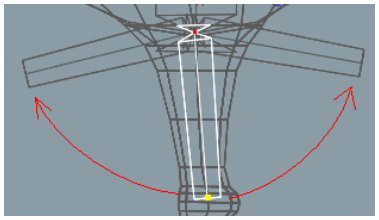
We can set now the orientation for the hip constraints of the leg. The constraint space should point towards the center of the motion range of the thighbone. For a leg, that direction is straight down. This is quite important: if the constraint space points upwards, key framing leg's motion from back to front would rotate the leg up through the body.

You can see the direction of the first constraint from the triangle at the start of the skeleton. If you moved the mouse down from the hip, the triangle can already point to the right direction. If not, press Ctrl down and LMB click in the middle of the knee. Ctrl-click does not add new joint but sets the constraint orientation.

Move the mouse forward and up until the thighbone is rotated to its extreme front position. Then rotate the bone back to the knee position and then backwards to its extreme back position. This way you can 'teach' to the bone how much it can move.


Ctrl click at the knee sets root constraint direction downwards

Click in the middle of the knee to add a new joint. Then carefully move the mouse down to the ankle point. Define the angle limits again by moving the mouse backwards and up, until the shinbone is rotated to its extreme position. Do not rotate the bone forward, because the knee does not bend that way. Click in the ankle to finish the shinbone.



Defining angle limits for a bone

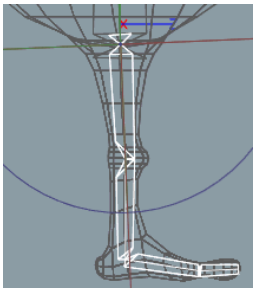


Note

If you accidentally define a wrong joint position, just select Go Back from the popup menu or hit Backspace. You do not have to start from the beginning. Note also that angle limits can be adjusted later easily. It is enough to draw the skeleton approximately the right way. You can fix and fine tune everything later.

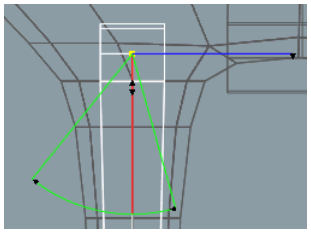
After the ankle joint, add one joint to the position where toes bend. Put the last joint to the tip of the toes. Click Accept to finish the leg skeleton. Rename the skeleton object as leg1.

Switch to the front view. Make sure that the backbone is in the middle of the body. Move it, if necessary. Select the leg skeleton. Use the blue object space handle to move the skeleton to the middle of the left leg.



The skeleton for legs finished

Hit the spacebar key to set the skeleton in edit mode. Click the little red knob at the hip joint to select the first joint of the leg. You will see some handles, which control constraint angles. Drag the little black triangle at the end of the green line to the left, until the angle sector appears suitable for thighbone's motion range to that direction. Then drag the triangle at the opposite end of the sector to the right.



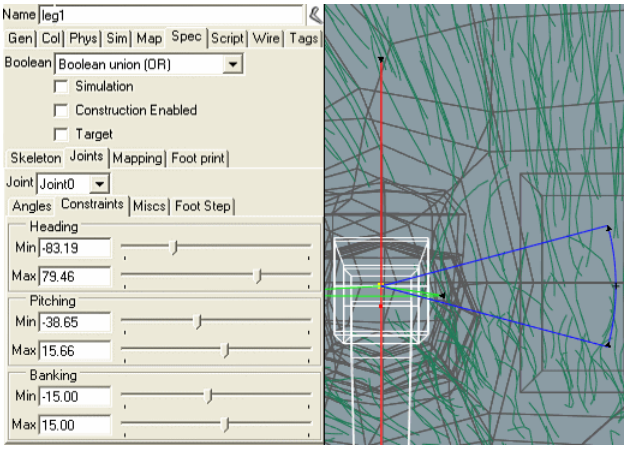
Adjusting angle constraints for leg motion sideways



Note

The cursor changes into a pen shape when the mouse pointer is close enough for dragging a constraint handle.

There is a third motion sector to define left: rotation of the thighbone around its axis. Take a top view. You can see the sector as a blue line. It can be adjusted just like the green sector above, but let's use numeric controls for a change. Open the property window, go to the Spec tab, and Joints/Constraints sub tab. Decrease the Banking/Min angle to, say, -15 degrees, and Max angle to 15 degrees. Note how the sector changes respectively on the view window. Close the property window.



Adjusting banking constraints from the property window

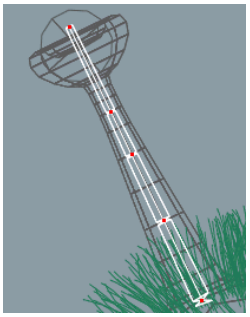


Note

If hairs clutter the view so that seeing the skeleton is hard, just select the hairs level and make it WF-invisible, then select the leg skeleton again. You can restore WF-invisible later when skeleton editing is done, or even leave the hairs permanently invisible in real time rendering.

Now the first leg skeleton is ready. Hit spacebar to turn the edit mode off. Switch back to the front view. Press Ctrl down and move a duplicate of the leg skeleton using the blue move handle to the middle of the other leg. Rename the copy as 'leg2'. Turn the edit mode on and adjust the green motion sector of the thighbone (it should be mirrored from the other leg).

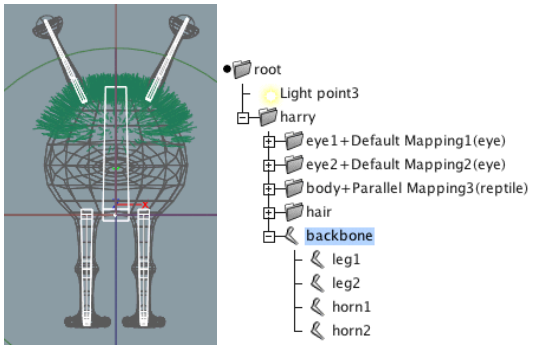
Legs are now ready. We need to add two skeletons more for the horns. We can assume that horns are quite flexible and bend to all directions symmetrically. So, we can draw the skeleton from the front view. Pan and zoom the view to see the left horn properly. Start the skeleton tool and draw the skeleton inside the horn the same way as you defined the leg skeleton. Remember to set the root constraint after the first point with a Ctrl-click, and swing the mouse slightly sideways to set symmetric angle constraints for each joint. For flexible bending, add as many skeleton joints as there are cross sections in the SDS control cage. When the bone and geometry density match, the result behaves elastically. You can end the skeleton anywhere after the bendable area. It does not have to reach longer than controlled geometry points. Rename the new skeleton as horn1.



The skeleton for the left horn

We need to allow some bending of horns in front-back direction as well. Set the horn1 skeleton to the edit mode. Click the first joint (the one close to the head) to select it. Open the property window and go to the Spec/Joints/Constraints tab. Enter the Pitching range Min=-30, Max=30 degrees. You can finetune the Heading limits to the same range if you like. Banking range can remain zero because horn shape does not need it. Activate the joints one by one and set same kind of angle constraints for them all. Then you can close the property window.

Duplicate the horn1 skeleton, move and rotate it inside the second horn geometry. Rename the copy as horn2. Take a side view and make sure that the horn skeletons are inside the respective geometry.



Multi select the leg and horn skeletons and drag them into the backbone hierarchy. Multi select all the sub skeletons, and hit the Attach button on the toolbar to connect the sub skeletons to the backbone. The Attach option means that pulling any bone with Inversed Kinematic tool will eventually drag the full skeleton along (unless so called anchors have been set). Select the backbone and click the Store button on the toolbar. This records the current state of the skeleton (including the sub skeletons) to be the native pose of the system. You can any time return to the stored pose by clicking the Native button. Finally, drag and drop the backbone object into the hierarchy of the Harry character.

The next step is to bind the points of the model to the skeleton. The project template for practicing the binding tools is 'tutorprojects/harry/skeleton.r3d'.

Select the backbone and hit the Bind Nrst (Bind Nearest) button in the toolbar. This operation, which binds all objects in the same hierarchy level, is a simple way to get started, although the result most likely needs some corrections. Note that the name of the backbone is now in bold typeface in the selection window, indicating that Bind Nrst is executed.

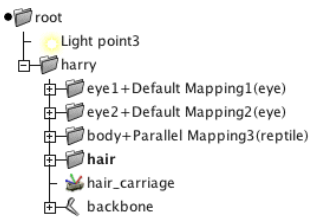
Make sure backbone is still selected and hit the spacebar to put the skeleton system into the edit mode. To test the result of the Bind Nearest action, drag the skeletons for example from the end of a toe bone to change the pose. Most likely you will see some strange deformations in the geometry of the character, because of some wrong skeleton bindings. Hit the Native pose button in the toolbar to reset the skeleton.

Apparently eyes and horns were mapped correctly - the respective skeletons are clearly the closest. Hair should be bound only to the backbone, but part of it got bound to horns. The simplest way to fix this is to unbind the hair completely and then make a new binding. So, multi select the hair level and the skeleton system's root object (=backbone) and hit the Unbind button in the tool control bar.

Hair no longer follows the skeleton. We can also clean up the model at this point. Open the hair level, then the interpolator object under it and select the first hair curve. The bold typeface on the select window indicates that the hair curves still have animation data attached (the root choreographs, under which the skeleton binding choreographs were placed). Make sure that the first curve is not in point edit mode and hit Ctrl-a to multi select all hair curves. Then select Delete Choreographs from the select window's popup menu. Note how the typeface of curves changes from bold to normal in the select window, to indicate that the skeleton binding has been removed.

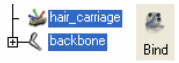
The skeleton we created will not be used to deform the hair - it is far too inaccurate for that purpose. Thereore, we can use a simpler and quicker way to attach the hair to the backbone. Instead of mapping the thousands of points one by one to the backbone, we will use a special Carriage constructor object to attach the hair object as a single target.

Close the hierarchy of the hair level and leave the level selected. Go to the Modifier tab of the toolbar and activate the Carriage tool. Click in the middle of the backbone, move the mouse and click a second time. The two clicks define a coordinate system, which will control the hair. The orientation of the coordinate system does not matter, because all its points will be attached to the backbone. When the tool is finished, a new item 'Carriage' appears to the select window. Drag and drop it to the hierarchy of the harry level, and rename it as 'hair_carriage' to describe its purpose better.



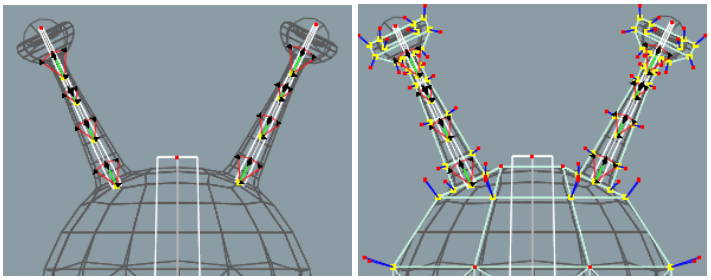
Put the new carriage object into the edit mode (hit spacebar) and select all its points by dragging a box around the coordinate system geometry. Next select the backbone object. It should still be in the edit mode. In the view window, click the bone inside Harry's head (= the real backbone) to select it. Remember: the selection is indicated by a highlighted bone axis and the constraint handles.

Then shift click the carriage object on the select window, to multi select both objects. Click the Bind button of the skeleton tool group on the tool control bar (note: use the Skeleton Bind, not the Carriage Bind tool). The geometry of the Carriage object became bound to the backbone.



The next goal is to fix the incorrect bindings of the body mesh. Multi select the hair level and the carriage object and hit 'i' hotkey to make the dense distracting wireframe invisible. Select the body level and hit spacebar to turn the edit mode on. If you do not see the point handles, open the level, select the actual SDS body under it and change its edit mode to Edit/Points from the tool control bar. Then close the level containing the body and leave it selected, edit state on. Make sure that all points are unselected - hit Ctrl + t to unselect all points, if necessary.

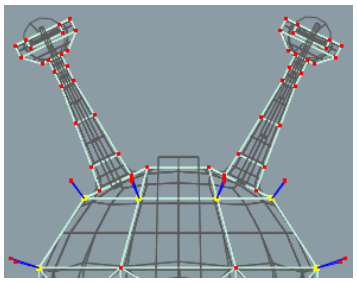
Select the backbone skeleton object. Drag select all bones on the left horn. Then press Shift key down and add all bones of the right horn into the selection by a new drag selection box. Shift click the Body level on the select window to multi-select it, too. Then hit the Select button on the tool control bar. All points mapped to the horns became selected.



Applying the Select tool to find out which points were mapped to the horn skeletons

Click the body level on the select window to remove the skeleton system from the selection. Then deselect all points of the horns of the body, for example by Alt-dragging two selection boxes around them. The remaining selected points are exactly the points, which were incorrectly mapped to horn bones instead of the backbone.

Press Shift key down and add click the Backbone object on the select window to multi select it with the body. The skeleton still has the horn bones selected after the previous steps of this tutorial. Click the Unbind tool to remove the incorrect binding.



Incorrectly mapped points extracted

Carefully change the object selection to the skeleton only by clicking it on the select window. You should not lose the valuable point selection of the body mesh by an accidental click or drag on the view! When the skeleton system alone is selected, go to the view window and click the backbone inside the body to select it. Then Shift click the body level on the select window to multi select it. Finally, hit the Bind button on the tool control bar to make a new, right kind of binding.

There are some more incorrect bindings at the leg-backbone joint area, but you can fix them exactly the same way as instructed above. Just select all leg bones, apply the Select tool to find out which points are mapped to them, exclude the correctly mapped points, unbind the remaining points from the leg bones and bind them to the backbone.

Save the project as 'harry_rigged.r3d'.

Animating the Character

In this section, we will add facial expressions and skeleton movements to the project.

The example project for practicing animation steps: 'tutorprojects/harry/rigged.r3d'.

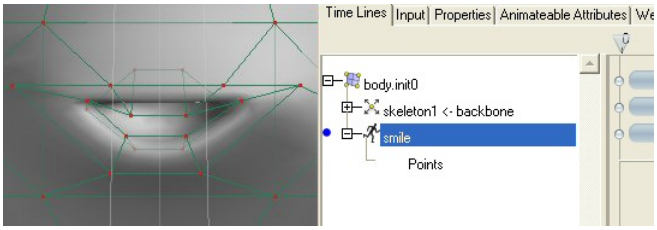
To see Harry's expressions better, turn shaded OpenGL on. Then, select the SDS object, which represents the body and set it to the point edit mode. If the animation window is not opened yet at the bottom of the screen, open it from the View pull down menu or by hitting the hotkey F7. The number of frames, displayed on the animation window, can be for example 100. We can change it later if necessary.

Open the choreography window (Animation/Choreographs from the pull down menu or F6 hotkey). Go to the Time Lines tab. There is a list of choreographs on the left side of the window. Open the popup menu of the choreography list gadget and make sure 'Only Selected Objects' is enabled. Otherwise the hundreds of skeleton mapped hair choreographs may show up although they will not be edited.

Now the choreography list shows one item: 'Body:Init'. It is the root choreography of the body. You can open the contents of it by clicking the little square at the left side of its name. It includes one item 'Skeleton1<-Backbone'. This is the choreography, which binds the body to the skeleton.

We will define a couple of expressions using a simple pose approach. Select the Body:Init choreography by clicking it and then New/Pose from the popup menu. A new item appears to the end of the choreography list. If you open its sub structure, you can see that it contains one animated attribute, Points. The attribute was automatically included in the pose because the previous skeleton chorography modifies geometry points.

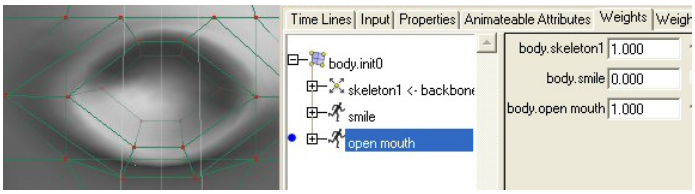
Points are just what we want to pose, so we do not have to make any changes in the attribute list. However, Select the pose item and apply Rename from the popup menu to rename it as 'smile'. In the view window, move the points of the mouth to form a smile expression.



The first pose: Harry is smiling

Go to the Weights tab of the choreography window. The weight of the new pose is at the full level 1.0. Move the body.Smile slider down and up, and observe how the expression changes respectively. Leave the slider to zero position. This makes it easier to define more expressions. Poses are combined by adding the weighted point movements together, and it is best to define each pose starting from a neutral expression.

Select New/Pose again (you can have the existing pose or the body:init0 choreography selected when doing this). Rename the new pose as 'Open Mouth'. Then edit the mouth area to reflect this new expression. Try the weight slider of the new pose to open and close the mouth. Leave the weight to zero.



The second pose: the mouth wide open

You can add as many expressions as you like this way. Let's see how we can use the poses in animation. But before that, there is one important change we must make: the order of choreographs is wrong. Namely, the skeleton choreography should be after the face expression poses in the choreography list.

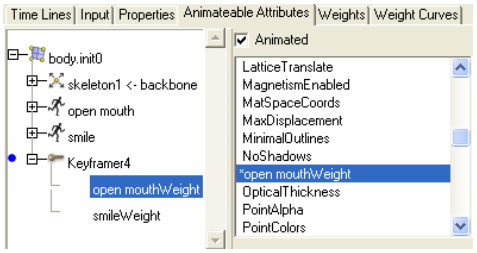
For example, the smile pose moves mouth sides upwards (up direction being defined in the object space of the body). If the skeleton had bent Harry's face to look down, movement upwards would move the points of the mouth inside the characters head, which is apparently wrong. To change the order, drag and drop the skeleton choreography after the two poses.



The skeleton deformation comes after the poses

Make sure that the body.init0 choreography is selected and pick New/Keyframer from the choreography list's popup menu. Click the new Keyframer item to activate it, and go to the Animateable Attributes tab. Find the attribute SmileWeight, select it and set Animated on. Do the same for the Open Mouth Weight.

The keyframer also includes Points as an animated attribute (it was included because the previous choreographs had it, too) - you may clear the animated state of that attribute. To do it, select Points from the attribute list and uncheck the Animated field. The keyframer will now only control the selected weights.



The pose weights prepared for key framing

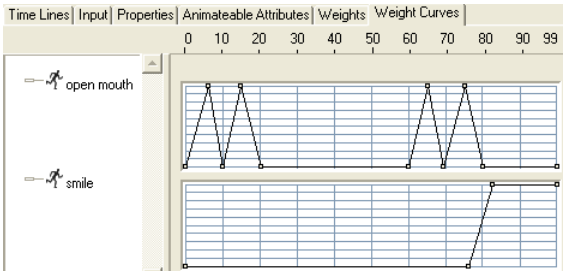
Switch to the Weight Curves tab. Select Create predefined curve/45 degree curve from the popup menu of the curve gadget at the right side of the smile item. Change the curve type to Poly line - it is continuous enough for controlling expressions, and very easy to set up. Then edit the curve so that Harry starts smiling at the end of the animation. To achieve this, click the curve about at the frame 75 and move the inserted point down to zero level. Then click the curve about 10 frames later and move the new point up to the full level one.



Note

The Weight Curves tab shows the curves, which control choreographs. The program does not draw any numerical scales to keep space requirements minimal. This is important, because often several curves must be visible simultaneously for animation synchronization. The vertical range of the curve graphs is by default 0..1, which is suitable for weight control. You can examine and edit a single weight curve more accurately in the Properties tab, or adjust the graph to include some negative weight range, too. A smile with a weight -1 is an unhappy expression.

Apply the same two menus, Create predefined Curve/45 degree curve and Type/Poly line, to initialize a curve, which controls the Open Mouth choreography. This time edit a curve, which opens and closes the mouth a couple of times in the beginning, then again during the third quarter of the time line. This Curve makes Harry talk. Make sure that the weight goes to zero when smiling starts. Otherwise the expression - a smile with a fully opened mouth - might look strange.

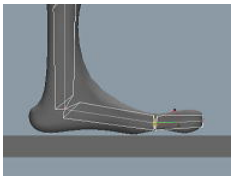


Weight curves, which make Harry talk and smile

Now you can play the animation and check that everything works as expected. The playback speed may be somewhat slow, mainly because of the large number of hair objects connected to the skeleton. You can also open a smaller view window from the Windows pull down menu and apply Animation/Preview from its popup menu to create a quickly playable .avi preview.

Next we add some skeleton controlled motions to the character. For example, let's make Harry jump. Rewind the animation to the beginning. To make animation easier, take a top view and use toolbar's Analytic/Cube tool to create a floor object. Take a side view and move the floor under the feet. Select the skeleton and make sure the skeleton is in its native pose - apply the Native button of the toolbar if necessary.

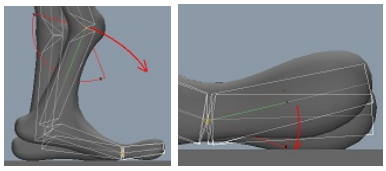
The first phase of a jump is that Harry squats down to get some speed for his jump. This probably takes half a second. So, RMB click on the animation window about at the frame 12. The time slider jumps to that position. Put the skeleton into the edit mode. From the side view, zoom into the feet. Drag-select both toe bones. Set the anchor selector in the tool control bar from Free to Anchor. A little anchor symbol appears to the locked joints. Be careful to lock the joints where the toes start from the feet, not the tips of the toes.



Toes anchored to floor

Turn animation recording on. Move the mouse over the knee position and drag the point handle at the knee forward and down. Dragging activates the inversed kinematics tool, which computes a pose by rotating the connected bones. Do the same for the other knee. Note that you can hit the Esc key to cancel joint dragging at any time.

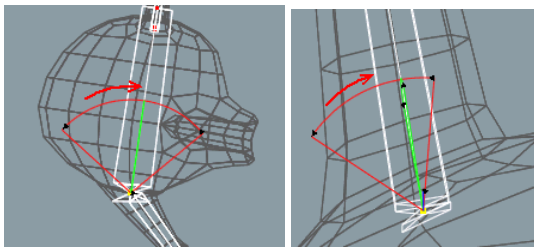
Drag the knees until the legs are properly bent. Then check the toes. It is possible that a toe angle constraint has rotated the toes of a foot when you bent the other leg. If that has happened, click the toe bone and drag from the red arc (forward kinematics handle) to rotate the toes so that they touch the floor.



Drag the knees with IK and fix the toes with FK

Click the backbone to activate it. Drag the red forward kinematics arc to rotate Harry's body to lean forward, so that his position becomes more balanced. Turn Shaded OpenGL off (from the View property window's Draw tab) if it is hard to see the handles inside the shaded skin. Then take a front view and zoom into the root of the horns.

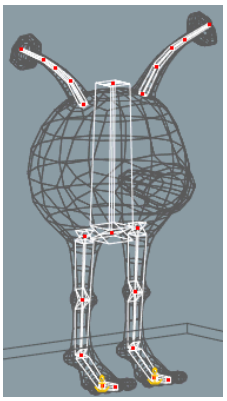
Select the lowest bone of the left horn and rotate it upwards using the FK arc. Be careful and zoom in enough - the bone is quite small, and there are many handles in that area. If you accidentally drag too close of the black triangles, you will start the angle constraint modification instead of FK. If that happens, hit the Esc key. Do a similar rotation for the right horn. The idea is just to make the horns match the motions of the body. These kinds of little details are important in character animation.



Rotate the body forward and the horns straight up from the FK handles

The next keyframe shows Harry leaving the ground. The jump starts with a quick acceleration, so 5 frames is quite enough. Move the time slider to the frame 17. Take a side view. Drag the knees back and up, to straighten the legs. The heels should still stay off the floor. You can also drag the joint at the hip to adjust the body position, and the FK handle of the backbone to make the body point straight up. Because Harry is accelerating the speed upwards, the flexible horns should bend downwards. So, take a front view and drag the horns from their end points to bend downwards.

The next position could define the highest point of the jump. But because the motion up and down is almost symmetric, we will now define the the key position when Harry has just fallen back to the floor, and his feet start resisting the impact of the floor. The highest point of the jump will be defined afterwards.



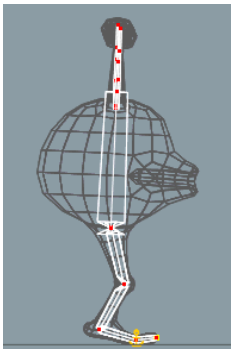
Harry is on his way up

So, move the time slider 10 frames forward, to the frame 27. Now Harry is falling and the horns should point again upwards; apply IK to the horn ends to pose them. From the side view, move the heels slightly downwards.

Now move the time slider back in time to the frame 22. Here Harry has reached the maximum height and just starts falling. Click the backbone to activate it. Press the y key down to lock the movement to the Y axis and drag the spine bone to the up most point of the jump. At this point, the horns have returned to a relaxed position. So, apply IK from the front view to achieve that.

We already defined the key frame (frame 27), where Harry hits the floor. The next key frame shows a squat position, because of the impact of hitting the floor. The squat position can become deepest in the frame 30. Move the time slider to that frame. Apply IK to the knees to squat him, check the toes and rotate the spine bone to lean forward. Don't forget the horns - they bend again down from the impact.

Move to the frame 33. This frame ends the jump. We can easily restore the original position by hitting the native pose button in the tool control bar. Then turn off the animation recording. You can also select the toe bones and release the anchors. The jump cycle is now ready. Rewind and play the animation to see the result.

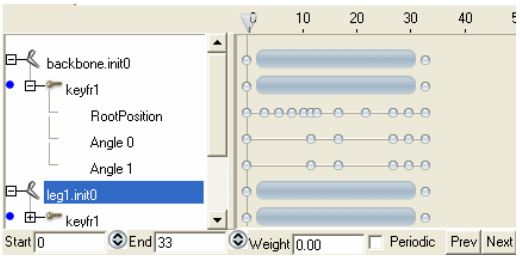


Harry squats when he hits the floor

If you followed the steps above, the motion probably looks quite good. However, some details need adjusting. Most likely the toes penetrate the floor when Harry straightens his legs. Let's fix these errors. Make sure the skeleton system is selected and open the choreography window. Go to the Time Lines tab.

The first thing that might need fixing is the length of time lines. We key framed the jump motion to happen during the frames 0 - 33. The program initialized the time lines to cover the whole animation. It will be easier to position and adjust the timing of the jump if the time lines reflect the key frame range. We will not add any other keyframe motions in this tutorial, and therefore we can set the root level time lines.

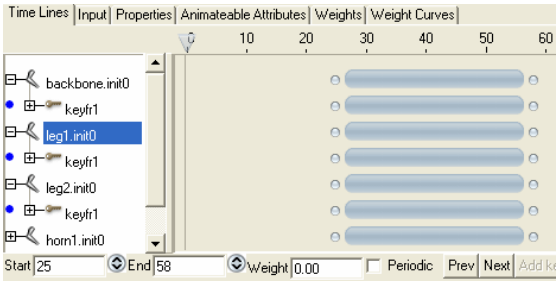
Click first the backbone.init0 choreography to select it. Open its sub time lines, including the keyfr1 time line, to see the animated attributes and their key positions. Press the Shift key down click the end knob of each .init choreography, one by one, to multi select it. Do not select any sub time lines! When all .init-related end knobs are selected, press both Shift and Alt keys down and drag the first selected knob to the frame 33. Alt-modifier scales the time line without moving underlying key frames.



Root level time lines normalized to match the key frame range

The keyframe choreography for the backbone includes some items we will not need. The native pose tool assigned also bone lengths to the skeletons. However, we did not actually change the bone length. A choreography, which assigns a constant value, is pretty harmless, but we can clean up the animation data and improve performance by removing unnecessary items. Click the Length 0 attribute name and hit the Del key to remove it. Remove also Length 1. Open also other keyframe choreographs and delete the length attributes from them.

Next we will change the time when the jump happens. Harry is talking in the beginning of the animation - let's make him jump after that, starting from the frame 25. Click in the middle of the time line of the first choreography (backbone.init). Press the Shift key down and select the rest of .init time lines by clicking in the middle of them. Shift key held down, LMB drag the first selected time line to the right, until the start knob is about at the frame 25.



Time lines moved to a later moment

Next we solve the problem of foot penetration into the floor. Move the time to the frame 25, where the jump starts. You can step the animation time back and forth frame by frame either by LMB clicking the time line, using the mouse wheel over teh time line or with the + and - hotkeys. Move time forward, until you find the frame where the feet are in the lowest position. In the example animation, it is the frame 31. Select the choreography attribute backbone.init0/keyfr1/RootPosition and go to the Properties tab. The position curves are defined in the object space of the skeleton. The red curve shows the motion in the vertical direction. Click the red curve in the position, which

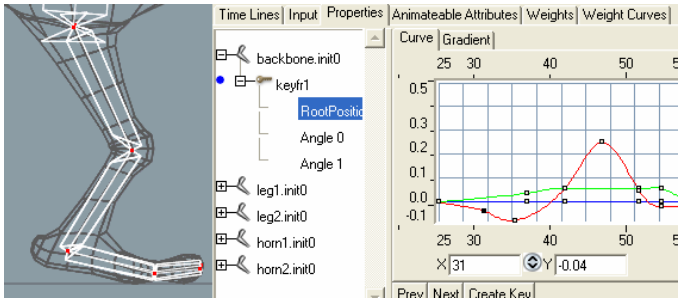
corresponds the frame 31. A new point appears to the curve. With this new key frame, we can fine tune the motion. Drag the point upwards, until the feet no longer penetrate the floor. The view window is updated instantly to reflect the editing of the animation curve.



Note

By checking the Show Tools option at the bottom of the animation curve editor, you can find some useful tools which help in editing the curves.

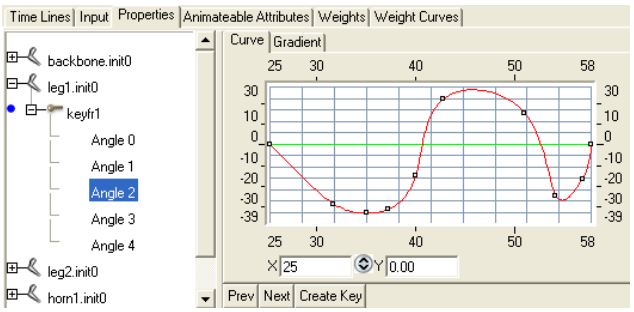
Note that the final height position of the feet is defined as a combination of several bone angles and the base position, which we just edited. Setting y=0 for the new curve knot does not mean that feet touch the floor. It means that Harry's hip is at the same height as in the first frame. If knees are bent or ankles are pointing straight down, feet will not match the ground.



Foot penetration fixed in the frame 31

When you move the time forward, you will notice that some frames later the feet penetrate the floor again. You can add a new point to the red curve and move it upwards to compensate. However, if you solve the penetration by only editing the vertical position of the skeleton, the result may look unnatural.

A better solution is to check all the joint angle curves and edit them as well. For example, leg1.init0/keyfr1/Angle 2 is the angle of the ankle. The ankles do not straighten up in the beginning of the jump. They give the final push for the jump just before the feet get off the ground. Editing the red curve to reflect this fact will create a better-looking motion and reduce the penetration problem.



Ankle rotations improved

Getting the motion right will take a lot of curve editing for all choreographs. It is also important to understand what the curves define, what is the relation of curvature to acceleration etc. Just continue editing the curves and experimenting until you learn to control the motions via the curves.

When you are satisfied with the jump motion, save the project.

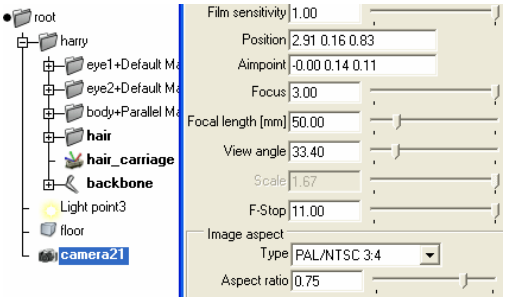
Rendering the Animation

The final stage of the production is rendering the animation to a suitable format.

The example project for practicing this part of the tutorial is 'tutorprojects/harry/animated.r3d'

The animation will be rendered directly to an .avi file. Because the images are not stored from a view window, we must define the camera view for each frame by adding a camera object to the scene. First we define the camera setup for the first frame. Navigate the view window so that you see the animated character for example from the side. Open the view property window, go to the Camera tab and turn perspective projection on. Make sure that the camera settings are OK. For example, you can set the focal length now to 50 mm. If the character does not fit into the 50 mm view, just increase the Distance value in the Orientation section of this view property sheet. When the camera setup appears suitable (don't worry too much at this point, you can change every detail later), close the view property window. Go to the Creation tab of the toolbar, activate the Camera tool and select Accept from the view's popup menu. This creates a camera object and records the camera settings of the view window into it.

Double click the camera object in the select window to open the property window. Go to the spec tab. Change the image aspect ratio to a suitable value. In this tutorial, we will use the image aspect of the PAL and NTSC standards. You can select it (height/width = 0.75) from the predefined ratios of the Image Aspect/Type gadget.



Remember to define the aspect ratio

You can adjust other camera settings, too, using the property window. For example, if the image was too dark, you can either make the flashlight color brighter, or increase the film sensitivity value. To preview the result, just drag and drop the camera object from the select window into the view window and render the view. Close the property window.

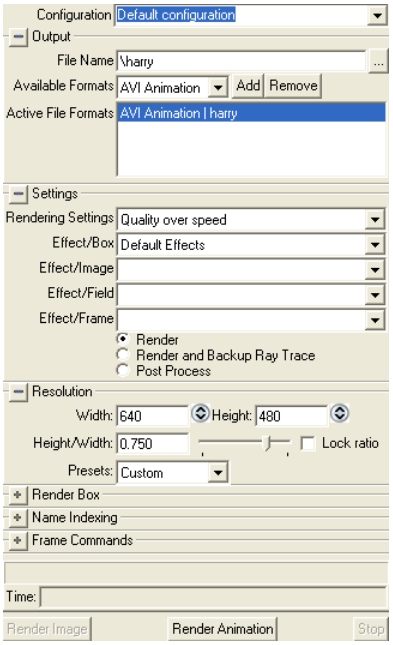
We can animate the camera easily using a couple of key frames. Move the time slider to the end of the animation. Then rotate the view so that you see Harry from the front. Decrease the distance to the character (4th view control from the top in the view control bar) and center the view to show Harry's face. The animation will end to a close-up of a smiling face. Turn animation recording on (use the red button on the animation window). Then select Camera/View to Object from the view popup. This tool records the view camera to the camera object. Because animation recording is on, this added a new key frame for the camera. Turn the animation recording off.

Note: View to Object operation does not assign the image aspect value, because view windows, which are attached to the main window, rarely have a useful image aspect ratio. Therefore, the key frame creation method above did not change the selected aspect ratio (0.75 for PAL) of the camera. If you zoom out slightly, you will see the rectangular wireframe of the camera on the view window. Most likely it does not match the proportions of the view window.

Play the animation to check the camera motion. If the view window does not follow the camera motion, make sure that View property window/Camera tab/Track animation play is enabled.

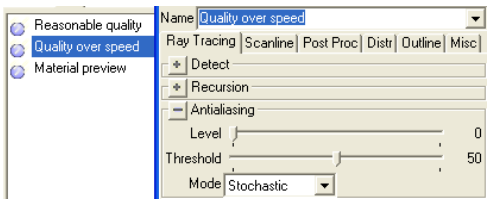
Most likely two key frames are not enough. Stop the playback in the middle of the animation. Navigate the view window so that the main actor is again properly in the picture. Then turn the recording on, apply again the View to object tool and turn recording off. Now the camera motion is ready. Rewind the animation back to beginning.

Select File/Render from the pull down menu. Enter a suitable file name and hit Enter. The default configuration has Windows BMP in the Active Formats list, which defines the outputted file formats. Use the popup menu of the list to remove it. Then select avi from the Available Formats list and hit the Add button. Now the output target is an avi file.



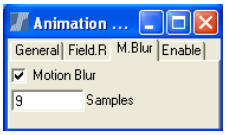
File render window

In the rendering settings field, there is an item Quality over speed selected. It is almost suitable for our project. However, thin hair objects require very careful antialiasing. The best way to antialiase such objects is using sufficient stochastic antialiasing. The important point is that stochastic antialiasing can be obtained as a free side effect of motion blur. On the select window, go to the render settings tab and select the item Quality over speed. Open the property window and go to the leftmost Ray Tracing tab. Set Antialiasing/Level to zero and Mode to Stochastic.



Antialiasing settings for use with motion blur

Close the property window. Select Windows/Animation Settings from the pull down menu. Turn Motion Blur on and set the number of samples to for example 9. Close the Animation Settings window.



Motion blur enabled

Go back to the file render window. We did not define any resolution yet. The official PAL and NTSC resolutions are included in the Presets box. Both PAL and NTSC use non-square pixels, and playing animations rendered to those standards on a computer screen shows the images stretched, because all usual screen resolutions have a pixel aspect ratio, which is quite close to one.

So, let's use a lower, computer-oriented resolution. Enter for example Width=640, Height=480. Note that the aspect field shows a ratio 480/640 = 0.75, which is exactly the aspect ratio we defined for the camera object. This means that the pixels will be exactly square-like (pixel aspect = 1.0).

Save the finished project. Then start rendering by clicking the Render Animation button. Note, that you can safely cancel the rendering, save the project to a temp file, reload it later and continue rendering. When the rendering is done, play the created .avi file with your favorite media player.





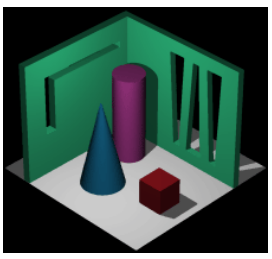
Chapter 2. Modeling

Using Curve, Circle and Rectangle Tools

Realsoft 3D supports three kinds of geometric objects:

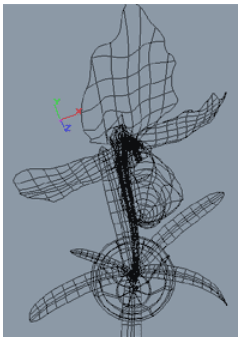
1. Analytic
2. NURBS
3. SDS

Analytic objects are memory efficient, fast to render and they perform fast Boolean operations. Analytic means that the surface and other geometric properties of an object are defined accurately using a mathematical equation. For example, an analytic sphere is defined by two attributes: center and radius. The problem with analytic objects is that due to their minimal definition, their geometry is very rigid and doesn't allow you to represent arbitrary freeform shapes.



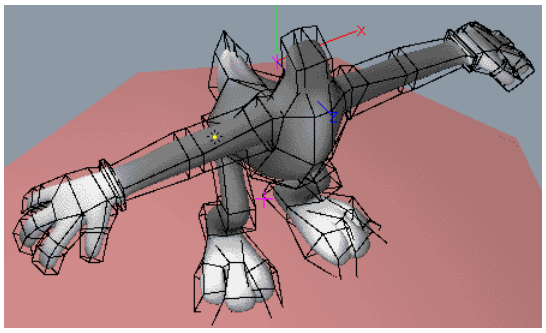
Various Analytical objects - Polyhedron with boolean holes, cylinder, cube and cone

NURBS (Non-Uniform Rational B-Spline) has become a standard curve and surface representation method in computer graphics. NURBS objects are slightly slower in Boolean operations but they can be used for representing freeform shapes easily, because a NURBS surface is defined by a set of freely editable points. One powerful feature of NURBS surfaces is the ability to cut away part of the surface with trim curves. A NURBS surface always has a regular, 'rectangular' topology: It consists of a grid of connected rectangular patches. Branching surfaces can be modeled by joining several surface patches together, and cutting with trim curves creates holes.



NURBS object with trim curves defining edges

SDS (=Subdivision) surfaces allow arbitrary polygonal and smooth freeform shapes. The major advantage of SDS surfaces is their free topology: A single SDS object can include branches and holes. Therefore, they are very suitable for modeling organic shapes, human figures, etc.



SDS object - created from an SDS cube

The toolbar or Realsoft 3D includes a specialized tab for each of these three object types. You can find basic geometrics shapes from each tab. You can also use so called unified toolset to create geometric objects. The unified toolset can create all geometry types described above, and even more: the four general purpose tools have a large number of sub options. By combining the sub options, you can access hundreds of different tools.

The default toolbar does not include the unified tools, but you can add the tools to your program interface using the Customization/Available Objects pull down menu.



Unified creation tools

You can activate these tools also from the Tools/Unified pull down menu. The menu contains four basic creation tools: Curve, Sphere, Circle and Rectangle. With these four tools you can create all basic geometric shapes with a freeform, spherical, circular and rectangular shape.

Using the Rectangle tool

The unified Rectangle tool allows you to create various objects with a rectangular shape:

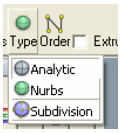
- Rectangular NURBS paths
- Analytic rectangular surfaces i.e. rectangles
- Extruded rectangles i.e. cubes
- Pyramids
- Pyramids with a cut top
- Rectangular freeform meshes
- Rectangular trim curves
- Paint rectangular shapes to bitmap textures
- Rectangular material mapping objects (parallel map, cube map, mesh map)

For example, to create an analytic rectangle:

1. Activate the unified Rectangle tool (Pull Down Menu/Tools/Unified/Rectangle)



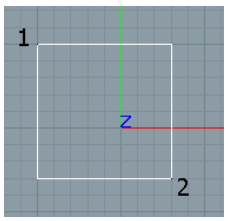
Unified Rectangle tool activated. The tool control bar shows you the unified Rectangle tool specific options



Select Analytic

2. Click the Type field and Select Analytic.

3. Now enter two points defining two opposite corners of the rectangle through the view window.



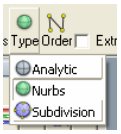
A rectangle defined by entering two opposite corner points

An analytic rectangle is very memory efficient and a fast object to render. It can be used for representing planar surfaces such as walls, floors, etc. However, it cannot be bent nor twisted. Analytic rectangles can only represent flat planar surfaces.

If you need to create a rectangle which can be deformed (for example, which can contain bumps in its geometry), you should create a rectangular NURBS mesh. This can be done as follows:

1. Activate the unified Rectangle tool.

2. Set the Type field to Nurbs.



Set the Type field to Nurbs

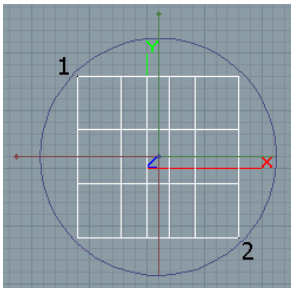
Now the tool control bar shows you options specific to NURBS meshes. For example, there are two resolution fields, which define the number of control points for the rectangular mesh.

3. Set U to 6 and V to 4 to create a Nurbs mesh with 6x4 control points.



Resolution set to 6x4

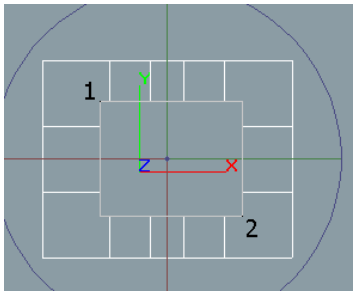
4. Enter two points through the view window to create a NURBS rectangle.



A Nurbs mesh consisting of 6x4 control points

The trim curve feature makes it possible to remove a part of a Nurbs surface. For example, to drill a rectangular hole to the mesh:

5. Make sure the Nurbs mesh is selected and activate the unified Rectangle tool. Set Class to Trim Curve and click two points over the Nurbs mesh to define a rectangular trim curve.



NURBS mesh with a rectangular hole

To create an analytic cube:

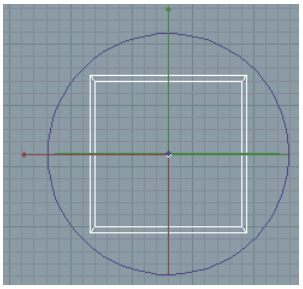
1. Activate the unified Rectangle tool, set the Type field to Analytic and set the Extrude option.



Options for creating an analytic cube

The Filled option is set by default. It means that the cube will be a solid one with 6 faces. If Filled is cleared, the cube will become hollow and without the front and back covers.

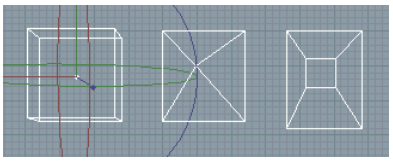
2. Define two opposite corner points through the view window to define the base rectangle. As soon as you enter the second point, the unified Rectangle tool starts to outline an apex point and the object looks like a pyramid. Don't define the third point. Just select Accept from the view's popup menu or from the tool control bar. A cube is created.



Analytic cube

If you want to create a pyramid instead of cube, enter three points and then press Accept. If you want to create a pyramid woth a flat top, enter four points.

In other words, the type of the object depends on the number of points entered. If only two points are entered, the result is a cube. Three points define a base rectangle and an apex (i.e. pyramid). Four points define a base rectangle and a top rectangle i.e. cut pyramid.



Three different objects achieved by using the Extrude option

Using the Circle tool

The unified Circle tool allows you to create:

- arcs
- disks
- cylinders
- cones
- cones with a cut peak
- hyperboloids
- disk and cylinder material mapping objects
- circular shapes to textures

To create an analytic cylinder:

1. Activate the unified Circle tool.

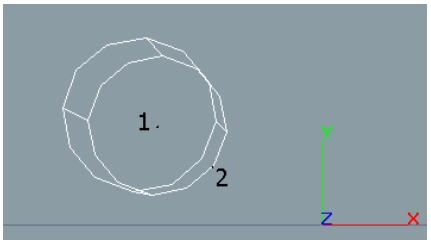


Options of analytic cylinder

2. Set Type to Analytic just like you did with the unified Rectangle tool and select Extruded from the Extrude gadget.

3. Click two points through the View window and select Accept. A cylinder is created.

If you accept the tool after three points, the result is a cone. Entering four points (center and radius for the bottom and top disks) creates a cut cone.



Shaping a cylinder

Note: the Define field allows you to control the creation method. For example, a circle can be defined by entering three points on it, by defining one center and one circle point or by defining a rectangle, which represents the bounding box of the circle.

To create a circular Nurbs curve:

1. Activate the unified Circle tool.

2. Reset the Filled option (we don't want to create a surface).

3. Set the number of points defining the curve to the Points field, for example 8.



Options for a circular Nurbs curve

4. Enter two points through the View window to define the center and the radius for the curve.

Using the Curve tool

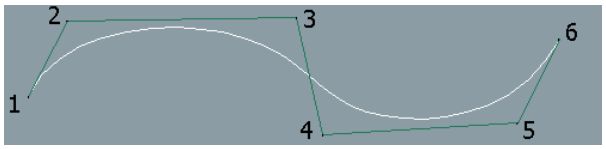
The unified Curve tool can create:

- freeform polygonal and smooth curves
- polygons, polyhedrons and polymids
- extruded curves
- meshes with a freeform boundary
- trim curves with an arbitrary shape
- freeform shapes to textures
- freeform material mapping objects

To create a smooth NURBS curve:

1. Activate the unified Curve tool

2. Enter a number of points through the view window. At your last point, select Accept from the toolbar or from the popup menu of the view window.



Drawing a smooth Nurbs curve

To create a polygonal curve:

1. Activate the unified Curve tool.



2. Set Order to Polygonal. Then define a desired number of points through the view window and click Accept.

Polygonal curves have sharp edges - quadratic and cubic curves use control points which are interpolated through to create a smooth curve. You can control the sharpness of these curves by adjusting the weights of each control point.



Polygonal curve

To create an analytic polyhedron (extruded polygon):

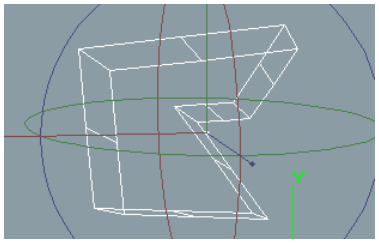
1. Activate the unified Curve tool



Options for a polyhedron

2. Set Type to Analytic and set Extrude. Define a desired number of points through the view window and click Accept.

3. The program starts visualizing the apex of a pyramid-like shape. A new mouse click would create such a pyramid, but choose Accept instead. In other words, Accept *twice* to create a polyhedron - one Accept + a new click creates a pyramid shaped polyhedron.



Polyhedron created

Using the Pen option to create surfaces

The unified curve tool supports a Pen option, which allows you to create Nurbs surfaces interactively. Use the pen option this way:

1. Draw a Nurbs curve as instructed earlier in this tutorial

2. Make sure the curve is selected and activate the unified Curve tool.

3. Select a suitable pen type from the pen gadget of the tool control bar.

4. Draw the new Nurbs curve. The program shows you the resulting surface as you enter the points of the new curve.

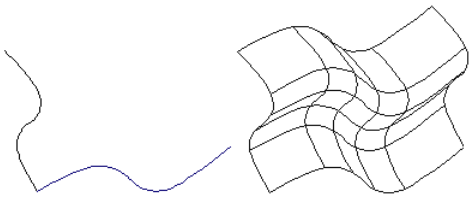
The following pen options are available:

None. If this is set, pen is though to be an infinitely small point and a curve is created.



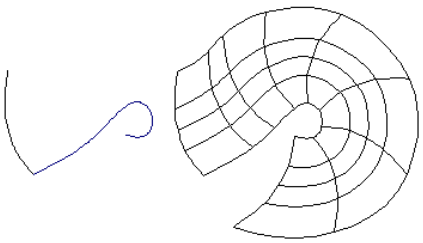
Plain curve

Rail. If this option is set, the selected pen curve will be swept along the curve drawn. The pen object is not rotated while sweeping.



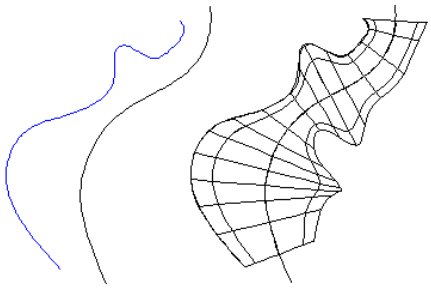
Rail pen

Sweep. This option is the same as the previous one except that the pen object is rotated according to the curvature of the curve drawn. For example, by using a circle as the pen object, one can create a circular tube.



Sweep pen

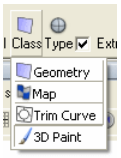
Rotate. This method uses the pen object as an axis and rotates the new curve around it.



Rotate pen

Using unified tools for mapping materials and for 3D painting

The unified tools can also create material mapping objects through the Class option. For example, we can use the unified Rectangle tool to map the wood material to a sphere using the Parallel mapping method.



Class option of the unified Curve tool

To do this:

1. Create a test object, for example a sphere.

2. Activate the unified Rectangle tool and set Class to Parallel Map. Now the control bar contains a list of available materials. Select wood and define two opposite corner points through the view window.

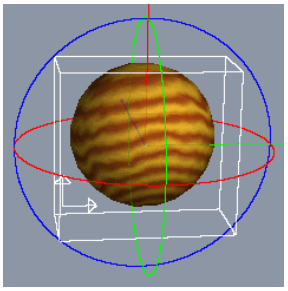


The Parallel map option of the unified Rectangle tool

A parallel mapping object is created. You can render the view window to make sure the created material mapping object works.

Note that there are also many special purpose tools for creating mapping objects. You can find them from the Materials tab of the main toolbar, for example. If you need to map many materials, you might find the specialized tools more productive.

Similarly, you can find the 3D painting tools through the unified tool interface, although the Materials section of the toolbar contains more specialized painting tools.



Wood mapped sphere

Using Object Handles

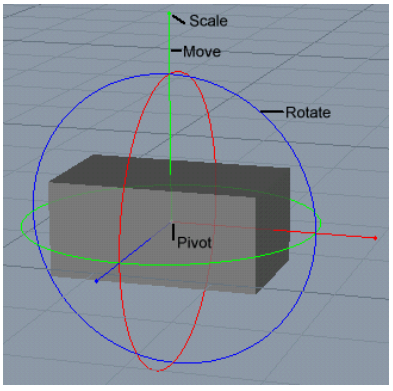
Object handles allow you to manipulate object properties directly - pretty much the same way people tend to manipulate objects in real life. For example, you do not have to push some 'move' button somewhere on your desk to enter 'move paper' mode or tool. You just grab the paper and move it.

Object handles in Realsoft 3D work the same way. They provide direct control over object properties like position, size and orientation.

Transformation handles

All geometric objects have nine transformation handles, which allow you to control position, size and orientation. When selecting multiple objects the handles of the first selected object are shown.

When you 'drag' (press the left mouse button while moving the mouse) one of the transformation handles, the corresponding transformation tool is activated. The transformation follows the mouse movement during the drag operation.



Transformation handles

Translation handles

To move a rectangle, follow these steps:

Create a small rectangle and select it. You can now see the three perpendicular lines, which are the handles to move the rectangle.

Move the mouse over the 'x' axis handle (the red axis), press the left mouse button and hold it. Note: when the mouse is positioned over the handle, the cursor changes to reflect the function of the handle. Also the status bar informs you about the purpose of the handle.

The rectangle is moved along its 'x' axis as you drag the mouse. When a desired new position is reached, release the mouse button to accept the tool. Hitting the 'Esc' key during the dragging cancels the operation.

Rotation handles

To rotate the rectangle just drag one of its circular rotation handles.

Scale handles

To scale the rectangle drag the end points of the move handles (the perpendicular lines). The scale handles are marked with small diamonds.

Note: the color of a handle corresponds to directions of the object space as follows:

- X - Red
- Y - Green
- Z - Blue

Activating handles by clicking them

You can also activate a transformation tool by clicking the corresponding object handle. In this case the clicked handle acts as a 'constraint' for the tool. All the tool options are shown and you will be able to use numeric interface, dragging and other snapping techniques.

Let us see what is possible in this mode but first a quick excursion to a useful technique called 'point snapping':

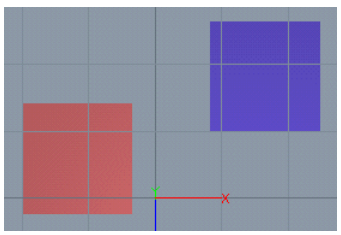
To 'point snap' specific points of an object, move the mouse above and to the left of the point and press and hold the left mouse button. While holding the left button down move the mouse downwards and to the right (drag the mouse) until the point is inside the small rectangle that is drawn. Now release the mouse button. This handy operation is called 'point snapping' and can be used to snap the mouse position exactly to a desired point position.

Now we will perfectly align two rectangles using the "drag selection" technique:

Create two rectangles say 'red' and 'blue'.

Now we want to move the red rectangle so that the upper horizontal edge of the red rectangle is aligned to the lower horizontal edge of the blue rectangle.

Select the red rectangle and click the 'y' translation handle (the green axis). The 'y' handle becomes thicker to indicate that it is selected and the Move tool is activated. The rectangle now moves on its 'y' axis as you left click and move the mouse.

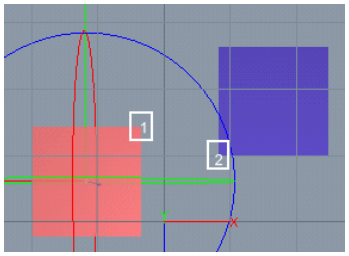


Rectangles to be aligned

Drag select the top right corner of the red rectangle using point snapping [1]. Then drag one of the bottom corners of the blue rectangle to define the target position [2]. The two edges of the rectangles are now precisely aligned!

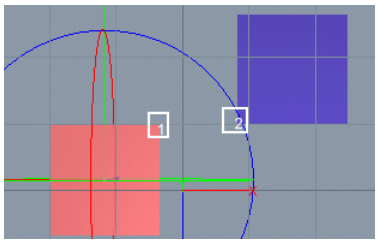
Now scale the red rectangle so that its top right corner matches the bottom left corner of the blue rectangle.

Click the end point of the 'x' translation handle (the red axis). Note: when you move the mouse over the end point handle, the cursor changes. This, in addition to the help text in the status bar, indicates that the scale operation will be activated if you click the mouse.



Snap the upper right corner of the red rectangle to the left bottom corner of the blue rectangle

Select the top right corner of the red rectangle using point snapping. Then drag the left bottom corner of the blue rectangle the same way. The right edge of the red rectangle is now perfectly aligned with the left edge of the blue rectangle!



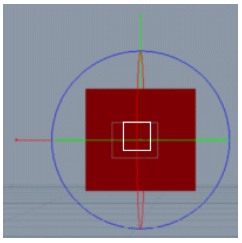
Use point snapping to define two points for exact result

The pivot point

When rotating and scaling objects, it is often necessary to define an appropriate center for the transformation.

When activating the Scale or Rotate tool from the Toolbar this center point is defined by the first mouse click. When using object handles the origin of the object space is used as the center of transformation. Many program call this 'center of transformation' point the *Pivot Point.*

To change the pivot point you can either drag the pivot point or click it. Dragging is perhaps faster to use but by clicking the pivot point you can use point snapping and other snapping methods for more precise results. Moving the pivot point means that you move the *object space origin*.

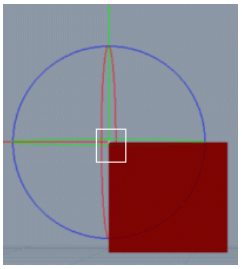

Drag over the pivot point

Now move the pivot point of the 'red' rectangle exactly to the left top corner:

Move the mouse over the pivot point handle and left click to activate the Move tool with 'Handles' option. Hitting the 'Insert' key also activates this "move pivot" mode.

Use point snapping to pick up the pivot point so that you can position it exactly at the desired point. Now the object handles move while you move the mouse.

Then drop the pivot by point snapping the top left corner of the rectangle. This accepts the Move tool and the pivot point is now located exactly at the top left corner of the rectangle.


Drag over the left top corner to drop the pivot point

Using modifiers with handles

The 'Shift' key is perhaps one of the most often needed modifiers. The purpose of this modifier depends on the tool.

Uniform scaling: If you scale an object, the Shift key acts as a '*uniform*' modifier. By dragging any of the three scale handles while holding down the Shift key, you can scale them uniformly in all the three dimensions. This is a common practice in many other applications as well, so it should be easy to remember.

For example, to make the red rectangle bigger:

Select the red rectangle and press down the 'Shift' key.

Drag a scale handle, while still holding down the Shift key.

The rectangle maintains its proportions in all three dimensions during the scale operation.

Rotating with angle grid When rotating an object, the Shift key activates the *angle grid.* The default angle grid is 45 degrees. This enables a rotation of, say, 90 degrees quickly and accurately.

For example, to rotate the red rectangle 90 degrees:

Hold down the Shift key

Move the mouse cursor over the blue circular rotation handle and drag (move the mouse while holding down the left mouse button) the mouse around the pivot point while still holding down the Shift key. You will only be able to rotate the rectangle in 45-degree steps and therefore 90-degree rotation can be easily achieved.

Duplicating and instancing: Many tools in the program use 'Ctrl' key as a 'copy' modifier. The 'Alt' key can be used for instancing objects. This is also true for object handles. Holding down the Ctrl key while dragging any of the handles first duplicates the selected objects and then applies the handle operation to the duplicated object. The original object is not affected.

Let's create four red rectangles.

Select the red rectangle (if not already selected) and press down the 'Ctrl' key.

Drag the 'x' move handle (the red axis handle) while pressing the 'Ctrl' key to duplicate the rectangle and move the new rectangle sideways. Repeat this operation three times to get three new rectangles.

There are many other useful modifiers which can be used for making object handles even more powerful. However, they are beyond the scope of this tutorial.

Using the numeric interface

It is often necessary to apply exact transformations to objects. Let us assume you want to rotate the red rectangle by exactly 23.4 degrees.

Make sure the numeric window is opened. If not, select the pull-down menu 'View/Numeric Window' to open it. You can now see how coordinates run on the numeric window as you move the mouse on the view window.

Select the red rectangle and double click one of its rotation handles. You can either double click the rotation handle or just click it two times. If you now move the mouse, you can see how the rotation angle field in the numeric interface shows you the current rotation angle.

Enter 'Numeric' mode by hitting the 'n' key. This activates the 'Angle' field in the numeric window. Alternatively, you can activate the 'Angle' gadget by clicking it with the mouse. Type the desired rotation angle of 23.4 degrees and hit the 'Enter' key to accept the rotation.

The selected red rectangle is rotated by exactly 23.4 degrees.

Also the Move and Scale tools support the numeric interface. However, when activated through handles, only one-dimensional scale and transformation factor needs to be defined. For example, to move the red rectangle 0.1 meters along its 'y' axis:

Click the green 'y' axis of the rectangle twice. The numeric window now shows you the displacement along the 'y' axis.

Hit the 'n' key and enter '0.1' in the Move1D field. Then press the 'Enter' key. The object is moved 0.1 meters along the axis.

Rotating and scaling about arbitrary points

Object handles allow you to scale and rotate objects easily provided that you want to apply the modification about the pivot point. As soon as you want to apply transformation about an arbitrary point, you have to move the pivot point. Moving pivot point frequently just to get an object rotated sounds like a lot of work and can soon become a 'major pain'. Fortunately, there is a faster way.

To rotate a rectangle about its left top corner:

Select the rectangle.

Click the desired rotation handle as usual. This activates the rotation tool about the pivot point. However, the tool activated by the rotation handle is the standard Rotate tool i.e. the very same tool you can activate through the toolbar or by hitting the 'r' shortcut key.

All transformation tools support 'Go Back' function, which allows you to re-enter any of the input points if needed. Therefore, it is possible to re-enter the rotation center. Select 'Go Back' from the view's pop-up menu or hit the 'Backspace' shortcut key.

You can now click or drag anywhere to define a new center of rotation. The transformation is still locked to the original, suitable rotation plane.

Transforming multiple objects

You can multi select any number of objects. In this case, only the first selected object provides the transformation handles so that the numerous handles don't clutter the entire view window. The handle of the first selected object transforms all the selected objects.

By controlling the selection order you can define a desired transformation space for modifying all the selected objects. For example, you can even add special 'reference' objects whose only purpose is to provide appropriate transformation spaces for the actual objects.

If you frequently have to transform objects in three special directions, just create three dummy objects whose object spaces match those directions. Then multi select the desired orientation object first and then the actual objects you wish to transform. The first selected object provides you conveniently oriented transformation handles.

Note: you can also change the object selection *after* you have activated a tool by clicking a handle. This is a very powerful feature, too. Let us demonstrate this with a simple example:

The goal is to rotate the red rectangle using the handles of the blue rectangle. You can, of course, move the pivot point of the red rectangle over the blue rectangle and then use the rotation handles to rotate the object. But this is not the same if the blue rectangle has a different object space orientation. However, there is an easy way to accomplish that: Just select the blue rectangle and click its rotation handle. This activates the Rotate tool and uses the pivot point of the blue rectangle. Then, using the select window, select the red rectangle. Then continue the rotation as usual. You have used a rotation handle of the 'blue' rectangle to transform the 'red' rectangle!

Accessing object specific handles

All geometric objects share a number of common properties, such as position, orientation and size. Correspondingly, they all support a set of common handles described above.

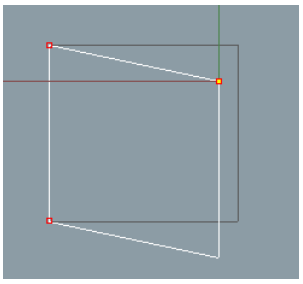
Additionally, geometric objects can define handles, which are specific to the object class in question. These object specific handles are shown when you enter the 'Edit' mode. Such special handles can activate arbitrary object specific tools.

For example:

Select the red rectangle.

Select 'Edit' from the view's popup menu or click the 'Edit' gadget in the tool control bar. The transformation handles disappear and three small point handles (little squares) appear.

Move the mouse over the handle in the top right corner. Press the left mouse button and drag the handle to a new position.



Editing a rectangle

Leave the edit mode by deactivating the 'edit' option from the popup menu or the tool control bar.

The object specific handles of a rectangle are the 3 points defining the geometry of the rectangle. You can use them to 'point edit' the rectangle. You can naturally also apply the usual tools like move, rotate, etc. and techniques like point snapping to these handles.

You can find more information about object specific handles from the tutorials describing geometric objects.

Working with Multiple Projects

Any number of projects can be created. Each project contains an independent set of materials, geometric primitives, post particle and post image effects. It should be noted that some data sections are shared between projects: render settings, grids, file render configurations are such shared objects.

You can cut or copy objects in one project and paste them into another project.

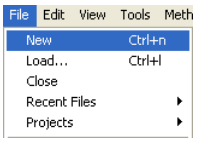
Animation time is also defined by a project: it is possible to continue modeling a project while playing the animation of another project.

There is one special purpose project called *material preview project*. By modifying this project, you can define how material preview renders preview icons for the select window's material tab.

Opening a new project

To create a new project, select the pull down menu: File/New.

This creates a new project and makes the new project the *current working project*. All newly created objects are inserted into the current project. Also, all windows (such as the Select window and the View window) reflect the contents of the current working project.

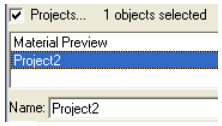


Create a new project

Switching between projects

For example, you can create objects in a temporary project and when ready, move the new object into the actual project by using the cut/paste tools.

The current project



Project list opened

To make a project the current working project:

1. Click the Projects check box in the select window. The project list is shown.

2. Click the desired project to make it the current working project.

You can rename projects through the Name field.

Deleting projects

The current project can be deleted from the project list by selecting the pull down menu File/Close. All objects, such as materials, geometric primitives, etc. are deleted. Note: the files are not deleted from your harddrive. The previous project in the project list becomes the current working project.

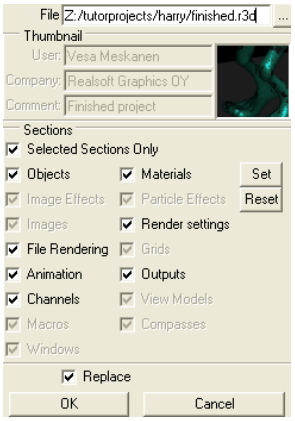
Loading projects from a file

The file load dialog, opened from File/Load pull down menu, is 'non modal' just like all the other dialogs in Realsoft 3D.

Load function automatically detects the format of the file to be loaded. If none of the installed file format classes recognize the given format 'Unknown file format' error message is shown.

The *Replace* check box defines whether the loaded objects should replace the existing objects or not. If not checked, new objects are inserted into the current project. If checked, existing objects are deleted from the current project before new objects are loaded in.

After a file has been selected, new format specific options appear to the window. The native Realsoft 3D file format provides options for loading individual data sections from a file. If the *Sections* box is checked, a suitable combination of data section selectors can be checked for loading only particular data from the file. For example, if you want to load only materials, check the Materials check box and reset all the other section check boxes.



The file load window in Sections mode, where you can select type of the data for loading

Saving projects to a file

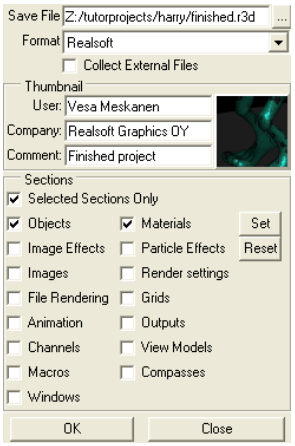
The menu File/Save As opens a file dialog. This dialog, just like the File/Load dialog, is 'non modal'. It does not block use of other windows and hence it can be left open when necessary.

The *File Format* gadget defines the file format used for saving the project. The gadget shows the installed file formats, which support save operation.

The *Thumbnail* contains a small screenshot. To create a thumbnail picture, just click in the picture frame. Name of the *user* and *company* are filled automatically, but can be altered if needed. The *comment* field can be used to add some project specific information.

Sections defines, which sections in the current project should be saved. See the File/Load description above. If this field is set to Selected sections, the window is expanded and a number of check boxes are shown for section selection. Save all tries to save all data sections. Note that some formats may not support saving of all sections. You cannot save Realsoft 3D materials to a DXF file.

If *Collect External Files* is checked, all the bitmap texture files used by the scene are collected into one single textures folder, which is placed to the same folder where you save the project. This helps greatly when you need to move a project to another computer system. Finding all the textures (possibly scattered to many folders) would be laborious to do manually.



The native Realsoft 3D file format selected. The lower part of the window allows you to specify file format specific options.

The pull down menu tool File/Save saves the current project using its name as a file name and the native file format. All data sections are saved.

The pull down menu File/Save as Startup saves the entire current project to the file, which is automatically loaded in program startup. Use this function after you have customized the working environment to suit your personal needs. The backup copy of the original startup file can be found from the 'environments' folder.

Rendering projects

The current working project can be rendered to a file using the File/Render window.

This File Rendering window allows you to select file format, rendering settings, post effects, image resolution, frame command and file name for the image to be rendered. Both still images and animations can be rendered. You should insert a camera object to the project before rendering it to a file.

See the manual chapter Rendering for more information about this topic.

Snapping to Points Using Dragging

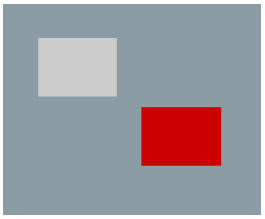
Many tools in the program require 3D geometry points as inputs. Typically, these geometry points are entered using the mouse, which is not a very accurate input device.

Snapping allows you to snap the entered mouse points to existing geometry points, achieving exact results during various modeling tasks.

In this tutorial we go through some basic snapping techniques, such as using the drag box and the lasso selector, as well as snapping to nearest point. You will find this functionality very useful in many modeling tasks.

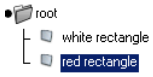
Dragging supported by the View window is a very powerful modeling aid. It allows you to snap to all existing points, pivot points, mid points and so on, all in one simple operation.

Consider the following situation: You have created two rectangles (red and white) and you have to align them so that the left top corner of the white rectangle matches exactly the right bottom corner of the red rectangle.



Initial position: two displaced rectangles

Do this:



1. Create two analytic rectangles as shown in the image.

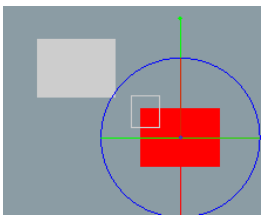
2. Select the red rectangle ...



... and activate the Move tool by hitting 'm' key or from the Transformation tab of the toolbar.

3. Move the mouse near the left top corner of the red rectangle. Press down the left mouse button and move the mouse. Drag box is shown. Move the mouse so that the left top corner is inside the drag box. Then release the mouse.

The corner of the rectangle was snapped into the mouse pointer. Now, move the mouse and the left top corner of the red rectangle follows the mouse precisely.

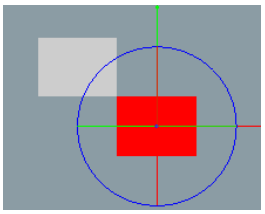


Drag over the left top corner of the red rectangle to snap to the corner

4. Now, drag over the right bottom corner of the white rectangle to snap the target point of the move tool there.

You can also drag over multiple points. In this case, all points inside the drag box are averaged. This allows you to snap to the middle point of the dragged points.

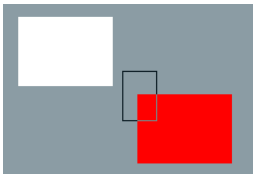
For example, let's move the left top edge of the red rectangle to the middle of the right vertical edge of the white rectangle.



Left top corner of the red rectangle snapped precisely to the right bottom corner of the white rectangle

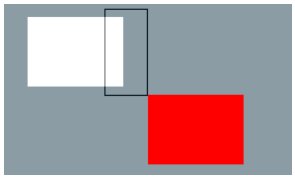
1. Select Edit/Undo from the pulldown menu to restore the original situation. Select the red rectangle and activate the Move tool.

2. Drag over the top left corner of the red rectangle. Move is activated and the dragged corner follows precisely the mouse.



Drag over the top left corner

3. Drag over the right vertical edge of the white rectangle so that its right edge is completely inside the drag box, and release the mouse.



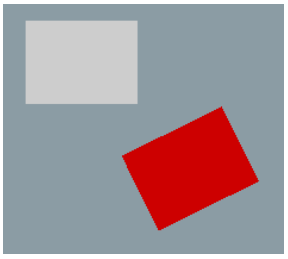
Drag over the vertical edge..



to snap to its middle point

Correspondingly, you can snap to the center of an object by dragging over the entire object. This averaging feature built in to the dragging makes the dragging very powerful feature because it allows you to achieve several important snapping methods through one easy to use interface. Dragging works with all transformation tools. Whenever you have to click the mouse in the view window, you can also snap to existing points using dragging.

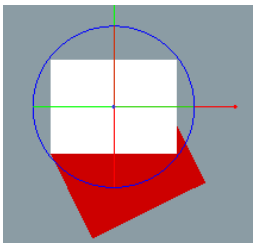
Let's see how you can take advantage of the dragging feature with the Rotate tool. The starting situation is like in the adjacent image. The task is to move and rotate the white rectangle so that its bottom edge matches exactly the top edge of the red rectangle. Thanks to dragging, it takes just a couple of mouse clicks to do this.



Initial positions

1. Select the white rectangle and activate the Move tool.

2. Now, using dragging, move the left bottom edge of the white rectangle to the left top edge of the red rectangle just like we did in the previous examples.



Corners snapped together

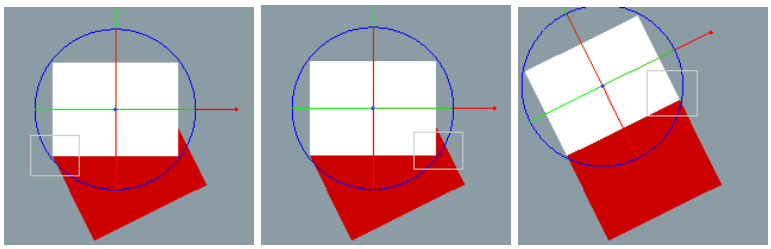
Next we should rotate the white rectangle about its left bottom corner to get the edges aligned.

1. Make sure the white rectangle is still selected and activate the Transformation/Rotate tool.

2. Drag over the left bottom edge of the white rectangle. This snaps the rotation center to the corner.

3. Enter the second point for the Rotate tool by dragging over the right bottom corner of the white rectangle. If you now move the mouse, the bottom edge points precisely towards the mouse.

4. Enter the last point for the Rotate tool by dragging over the right top corner of the red rectangle. Edges are now precisely aligned!



Drag three points for the Rotate tool: center, from point and to point



Note

Dragging averages *all* points inside the dragged area, including overlapping ones. For example, dragging over the white rectangle in the image above does not snap to the center of it, because the top edge of the red rectangle becomes included as well. The average of two points of the white top edge, two points of the white bottom edge and two points of the red top edge is not the middle point.

As you can see, dragging is an extremely simple but extremely powerful feature. With just a couple of mouse clicks you can move, rotate and scale about any point.

Lasso selector

In addition to the rectangular drag box, you can use a lasso selector to pick up geometry points. You can activate the lasso selector by holding down the lasso selector modifier. By default, this modifier is bound to the key '4'. Press down this key and drag on the view window to see the lasso selector.

3D snapping

Most people still use a two dimensional monitor and a mouse. This means that you can only see the position of a point using two coordinates x and y.

By default, dragging eliminates the third z (depth) coordinate so that when you, for example, move objects using dragging, their depth is not affected - the distance from view's camera to the two rectangles above did not change in the dragging operations.

By holding down a special *depth modifier* you can override the default behavior and dragged points are entered as true 3D points. The default key for depth modifier is ',' (comma).

Snapping to the nearest point

In addition to dragging, you can snap to single geometry points using snap to nearest modifier. Current mouse position is snapped to the nearest snap point when you hold down the key '5'. The nearest point snapping works always in 3D (the depth changes as well). Note, that snapping starts as soon as you press down the left mouse button. This means that you can drag the mouse around and see which snap point becomes selected - release the mouse button when a suitable one is found.

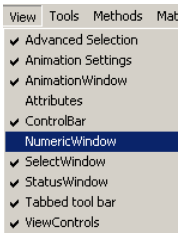
Snapping to the nearest curve

By holding down the key '6', you can project the current mouse position on the nearest curve point (any point, not only edge/control point).

Using the Numeric Interface for Accurate Modeling

In addition to the mouse, the input data for creating and transforming objects can also be defined using a numeric interface.

For example, to create a curve by entering the control points numerically:



Open the Numeric Window

1. Open the numeric window by selecting the pull-down menu View/NumericWindow.

2. Activate the Nurbs Curve tool from the 'NURBS' tab. The numeric window now shows you controls specific to the NURBS Curve tool. There are two fields: 'Position' and 'Relative'. The former allows you to define coordinates in 'world' space. The latter allows you to define points relative to the previously entered point (or to the 'hot point', which is the origin of the input plane, when no points has been entered yet). 'Relative' coordinates can also be accessed in polar (angle coordinate) form.

3. Hit the 'n' key. This activates the Position field. You can also click the field in the numeric window to activate it.



The numeric window with the 'Position' field activated

4. Enter three coordinates by separating them with a white space and hit Enter. This creates the first point for the curve.



Enter the first point and hit Enter

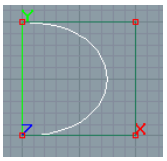
5. Enter three more points. After each entered point click the Enter key. Note: in some countries comma ',' is used as decimal point in which case you can replace the '.' in the following examples with comma.

```
0.1 0 0
```

```
0.1 0.1
```

```
0 0.1 0
```

Now four points has been added to the curve, and it has an arc-like shape.



Shape after 4 points

6. Let us try the relative measuring next: hit the Tab key. The Relative field becomes activated. Type the coordinates 0 -0.2 0 and hit Enter. A new point is added -0.2 meters below the first one. The relative measuring field remains active and you can continue entering coordinates this way, or hit Tab until the focus gets back to the Position field.

7. Now click the Accept button in the tool control bar. This terminates the curve tool and the curve is created.

Expression evaluation

All numeric fields support standard expression evaluation.

For example, to create a sphere with a radius sin(0.3)

1. Activate the sphere tool.

2. Click the left mouse button in the View window to define the center of the sphere.

3. Enter numeric mode by hitting the key 'n'. This automatically activates the Radius field in the numeric window. Type

```
sin(0.1)
```

and hit Enter.

This creates a sphere whose radius is 0.099833.

Note: Also vector fields support expression evaluation. For example, you can define the center of a sphere by entering three expressions separated by a space:

```
0.1+0.1 2/3 0.224*0.555
```

Measuring units and accuracy

The program uses internally the standard SI units. For example, positions and sizes of geometric objects are defined as meters. All numeric fields of the user interface convert internal SI units to desired current units.

You can select current units from the Preferences window.

If you want to work with millimeters:

1. Open the Preferences window from the pull down menu File/Preferences.
2. Go to the Metrics tab.
3. Set the Distance field to 'mm'.
4. Close the window.

Now all numeric fields that represent distances (positions, lengths, etc.) are shown in millimeter units.

Note: you can change units any time. It has no effect to the scene you are modeling. The user interface only displays properties of objects using the new units. The Metrics tab of the options window contains also a slider for defining the number of decimals. If the measuring unit is meter and number of decimals is three, all dimensions are shown with millimeter accuracy (0.001 m = 1 mm).

Grids

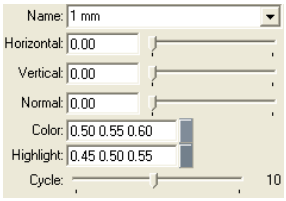
In addition to dragging and the numeric interface, grids help you to enter accurate coordinates during the modeling process.

Any number of grids can be defined in the select window's grid tab.

To create a new grid object:

1. Go to the Select window's Grids tab.

2. Press the right mouse button and select New from the select window's pop-up menu.



Editing grid properties

3. Select Properties from the select windows pop-up menu to edit the grid properties. The following options are available:

- Name: The name of the grid object
- Horizontal, Vertical, Normal: sets the grid dimensions, which define the distance between snap positions. Usually, all three fields are set to the same value. If you wish to enter small values like 0,001, make sure the value of the options/window/metrics tab is set to match.
- Color: The base color used for grid drawing.
- Highlight: A second color used for drawing the grid lines, whose frequency is determined using the Cycle slider.
- Cycle: Defines the density of grid lines that are drawn with the highlight color.

- Dynamic: Automatic adjustement of snapping interval.
- Dynamic Color: Automatic adjustement of grid color for dynamic grids. The shorter the snap interval, the lighter grid is drawn.

You can activate the desired grid by dragging and dropping it into the view window. This sets the current grid for the view window and activates both Snap to grid and Show grid options. You can also use view property window's Input tab for controlling view's grid snap and draw options.

Note: do not mix the grid with the ground plane visualization in perspective views. The ground plane has nothing to do with snapping and is provided as an orientation guide only. It is automatically turned off when grid drawing is enabled.

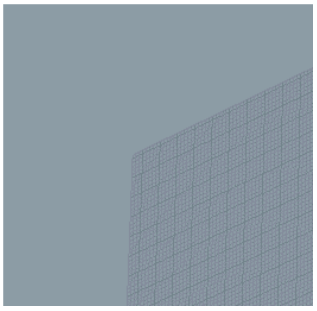
The view control bar allows you to switch Snap to grid and Show grid options on/off quickly any time.

Show Grid and Snap to Grid icons

The grid drawing fills the entire view window. If you zoom out the view window so that the grid appears denser and denser, the grid system stops drawing the base colored lines at some point, because the too densely drawn lines would not serve any useful purpose. Drawing only the highlighted lines keeps grid visualization useful. If you continue zooming out, also highlighted lines get too dense and their drawing will be stopped.

The same density control is used when perspective projection is enabled and view camera is not locked to the grid plane. Only the part of the infinite grid plane, which is not too dense, is visualized.

The automatic density control does not affect grid snapping, only grid drawing.



Too dense part of the grid is not drawn in perspective mode

Boolean Operations

Boolean operations offer a very powerful way of creating and modifying objects.

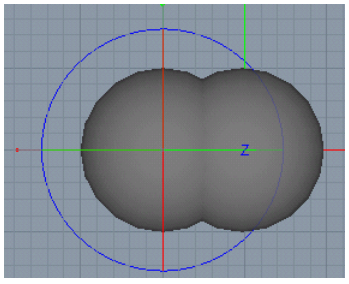
Consider a situation where an engineer cuts an object made of pine with a tool that has just been painted. Since the paint on the tool is still wet, it sticks on the object and the cut surface is painted with the wet paint. The engineer notices this and replaces the tool with another clean tool. The new cut surface is clean showing the beautiful texture of the pine. The Boolean operations of Realsoft 3D work in a similar way.

A simple example of a object created with a Boolean operation is a lens. A lens can be thought of as the intersection of two partially intersecting spheres.

Tutorial level: Beginner

Example project: 'tutorprojects/modeling/boolean/splitlens'

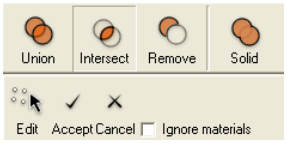
1. Create an analytic sphere using the Analytic Sphere tool.



Two intersecting spheres

2. Duplicate the sphere by holding down the Ctrl key while moving it so that the original and duplicated spheres are partially intersecting (you can also use Edit/Duplicate pull-down menu to duplicate an object).

3. Multi select both spheres. As usual, this can be done by clicking the both spheres in the select window while holding down the Shift key.

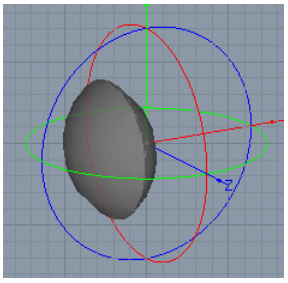


Intersect tool activated

4. Switch to the Boolean tab of the toolbar.

5. Activate the second tool, Intersect.

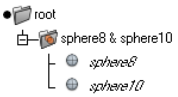
6. Then click Accept.



Intersection of two spheres

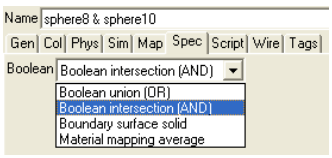
The resulting object will be the part of the work piece that is inside the tool. It is a lens. If you change the material of the lens to glass, it can be used as a magnifying glass!

In the select window, you can see that the two objects have been replaced with a new level named 'sphereX & sphereY'.



Structure of a Boolean operation between two spheres

This reveals the nature of Boolean operations in Realsoft 3D: a Boolean operation is just a *level* object consisting of objects to be operated. The objects used in these operations can be complex ones with many levels of hierarchy, and they can contain objects that result from previous Boolean operations.



Activating intersection for a level object

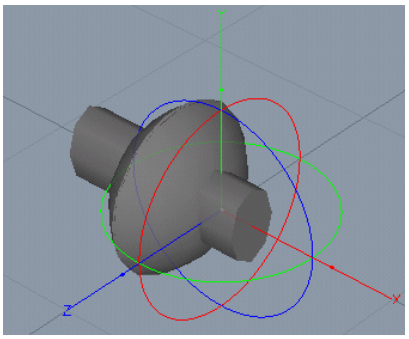
This means that instead of using the procedure described above, you can also make Boolean operations manually. When you create a new level, its type is set to union (OR). If you want to do intersection instead, open the property window and change the Type field of the Spec tab to Boolean Intersection.

Note that the original Boolean operated objects (sphere1 and sphere2) can be modified after the operation just like any regular objects. You can even activate the Simulation for the Boolean level and let gravity and other simulation effects to animate the sub objects.

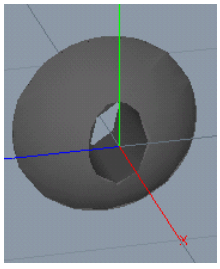
Let's continue experimenting with the lens object by cutting an oval hole through the lens.

7. Rename the created intersection level as Lens.

8. Create an oval analytic cylinder passing through the lens object.



A cylinder passing through the lens object



A lens with an oval hole

9. Select the lens and the cylinder object (in this order) and activate the Boolean operation tool again. Select the third Boolean operation option Remove and click Accept.

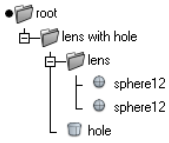
A new object 'Lens & !cylinder' is created. Rename it as 'lens with a hole'.

In Realsoft 3D it is possible to execute the Boolean operations so that the tool affects not only the volume but also the surface of the object that it is operating on. This is a very useful feature. For example, if you cut a notch in a pine board using a shiny reflecting (material attributes) tool and use Remove operation with Paint option, you get a shiny reflecting notch on the board.

Likewise, if you use a red tool you get a red notch. And if you use a chimney to cut a notch, you get a piece of wood with a notch of brick coating. On the other hand, if you use any of the non-painting operations, you get a clean notch

of pine on your pine board (the wood texture of the pine board 'x-rays' right through the object so that you get a continuity of the texture). In this case the surface of the tool does not affect the object that is operated on.

To test this feature:

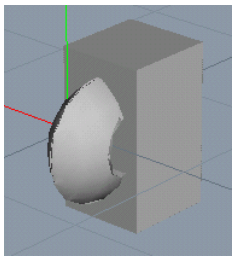


A hierarchical Boolean operation

10. Open the 'Lens with a hole' object by clicking its '+' handle in the select window. You can see that the level consists of two sub objects: the original complete lens and the cylinder representing the oval hole.

11. Select the oval hole object and open the property window. Go to the Col tab and change the color of the cylinder to red (1 0 0).

Click Render and you will see a white lens with a red hole.



A cube enclosing half of the lens

You can disable this Paint effect by clearing the Paint in Boolean Operations option in the cylinders property window's 'Gen' tab. You can obtain the same results by setting the Ignore materials option of the Boolean tool, instead of changing the attributes afterwards.

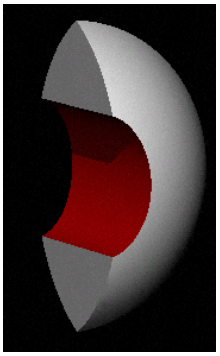
Finally, we can make a section diagram of the lens by removing half of the lens using a cube object.

12. Create an analytic cube so that half of the lens is inside the cube.

13. Select the 'lens with a hole' and the cube. Then click the Boolean tool in the toolbar, set Intersection option and click Accept.

If you have OpenGL switched on, the inaccurate polygonal representation might not look that beautiful. However, if you ray trace the view, you can see a perfect half of a lens with an oval red hole through it.

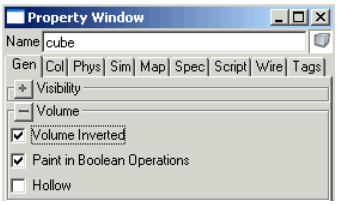
This is still a very simple object for Realsoft 3D. You can continue Boolean operations resulting in more complex shapes. They can be saved to disk and used in constructing other objects. As you complete useful new objects and tools, using the program becomes easier and faster.



A split lens with an oval hole through it

Inverting volumes

We created half of lens as an *Intersection* of the lens object and a cube. All parts of the lens outside the cube were removed.



Inverting volume

If we invert the *Volume* property of the cube, the operation will cut the part of the lens inside the cube. As a matter of fact, this is the way the Remove Boolean operation works. *Remove a cylinder from a lens* is the same operation as *Intersect a Lens with an inverted cylinder*.

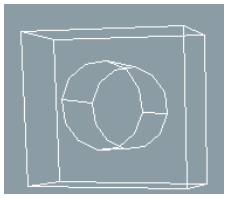
To invert the volume of the cube, which cuts half of the lens:

14. Select the Cube and open the property window. Go to the Gen tab and set the Volume Inverted checkbox.

The previously removed half of the lens becomes now the result of the operation.

Wireframes of Booleans

Real-time visualization of a Boolean operation does not always show properly the true shape of the object. The reason for this is that the wireframe for a Boolean operation level is computed from wire frames of sub objects, and performing the operation often increases the quality requirements. Wire frame and shaded quality can be adjusted using the quality settings of the property window's Wire tab.

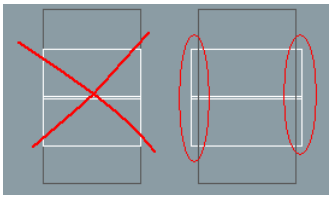


Parts for a Boolean operation

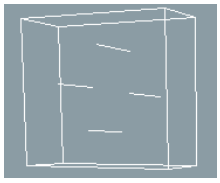
The automatic computation of Boolean wireframe is not always desirable. Turning Boolean wireframe computation off can increase the screen refresh rate and sometimes even give a better visual look. The next example demonstrates this.

1. Create an analytic cube and an analytic cylinder through it.

2. Now check an important thing: take a side view and make sure that the cylinder is long enough. The ends of the cylinder must stick out from the cube. If the objects are equally long, drilling with the cylinder will not succeed. Stretch the cylinder, using the blue rotation handle, if necessary to achieve this.



Verify that the drill is long enough

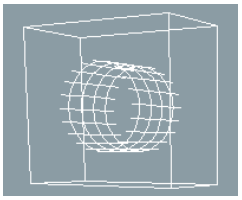


Insufficient Boolean wireframe

3. Select the cube and then (with the Shift key held down) the cylinder. Activate the Boolean tool and select the third operation type Remove. Click Accept.

All the parts of the cylinder wireframe outside the cube are cut away (because they do not belong to the resulting volume). This leaves very little wireframe left. Let's first adjust it by using the wireframe quality controls.

4. Open the hierarchy of the level 'cube & !cylinder' created by the Boolean operation from the select window. Select the cylinder. Open the property window and go to the Wire tab. Change the Wire density slider from the default value 1 to the value 5. Then select the parent level. Click AND wireframe off button first on and then off to cause a refresh of the Boolean wireframe.



Cylinder wireframe density increased

Now the denser wireframe clearly shows that a hole has been drilled through the cube. However, the original wireframes of the cube and the cylinder would give an elegant and beautiful representation of the operation (see the illustration of step1 above). To achieve this:

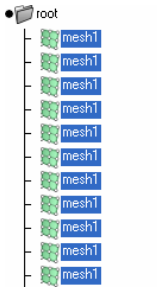
5. Select the cylinder again and drop the Wire density value back to one. Then select its parent level and check the AND wireframe off box. Now the object disappears from the view window. This happens, because the Boolean tool made sub objects invisible, so that they would not obscure the computed Boolean wireframe. Switch to the Gen tab of the property window. Select both the cube and the cylinder, and clear the Invisible in real time rendering box. The original wireframes appear again, which is what we wanted.

Summary:

- A Boolean operation level automatically computes a new wireframe/shaded representation from its child objects
- The quality of the Boolean visualization can be controlled using the Wire attributes
- Boolean wireframes can be switched off if necessary

Boundary representation solids

In the previous examples we used analytic sphere, cylinder and cube objects in Boolean operations. All analytic objects are very fast in Boolean operations because their interior volume is defined efficiently by a mathematical formula.



Two nurbs meshes and a set of weld objects defining a closed volume selected

However, it is also possible to use one or more surfaces, such as NURBS or subdivision meshes, to construct new solid objects. This kind of freeform solids are defined using so called Boundary Representation Theory. Solids defined this way can be used in Boolean operations the same way as analytic objects.

To create a new solid object from a set of NURBS meshes:

1. Create and select any number of surfaces defining a closed volume.



Note

Closed volume means that it should not be possible to get inside the object without passing through a surface.

2. Activate the Boolean tool. Select the last operation Solid and click Accept.



Operation set to Solid

This creates a new solid primitive from the selected surfaces. Now you can use this new level object similar to other solid objects.

For example, to drill a hole through the new boundary solid object:

3. Create an analytic cylinder through the object.

4. Select the level object and the cylinder (in this order) and activate the Boolean tool with Remove option. Then click Accept and render the view to see the accurate result.

Editing Boolean operations

When you execute a Boolean operation, the hierarchical structure of the object is modified to reflect the resulting object. If you move the lens object of the previous example, you notice that the hole faithfully follows the lens.

On the other hand, you can select the hole and make it bigger. The result of the Boolean operation changes immediately to reflect this change. You can also create animations where the hole moves, or where the progression of the hole being drilled is animated. The hierarchical implementation of Boolean operations makes this very easy in Realsoft 3D.

Advanced Boolean Operations

Realsoft 3D has a very powerful Boolean feature set that allows the user to create subtractions and unions of objects in many ways. Not limited to simply cutting a hole through an object, Realsoft 3D can cut entire scenes in half and have complex level structures join together to create very detailed objects.

Most commonly, Boolean operations are used to subtract one object from another. For example, a drill cutting a hole into a desk. But the power of Booleans allows us to use more than just one object. By dropping a whole series of objects to a level, we can make multiple cuts into one object:

1. Create an analytical cube.

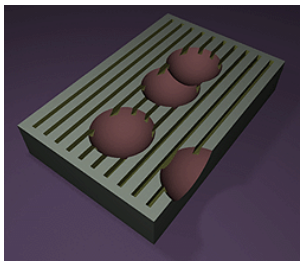
2. Duplicate the cube (Ctrl+d) and size it so it is a very thin slit, smaller than the original using the handles. Position it so that it is slightly inside the original cube.

3. Drop this thin cube into its own level (Right click "drop to a Level") and rename the level: "Cuts"

4. Select the thin cube in the level. Make the "cuts" level the current level (Right click "Make Current")

5. Turn on Macros/Record (Macro pull down menu).

6. Duplicate the thin cube and move it up slightly. (Do not use the handles to duplicate - use either the pull down menu, right click, or the keyboard shortcut Ctrl+d to duplicate)



7. Turn off Macros/Record. Press the Ctrl+g key to execute the macro once. Repeat until you have the desired number of thin cuts (say 10).

8. Make the Root level the current level (Right click "Make Current"). Select the first cube, and the level "Cuts".

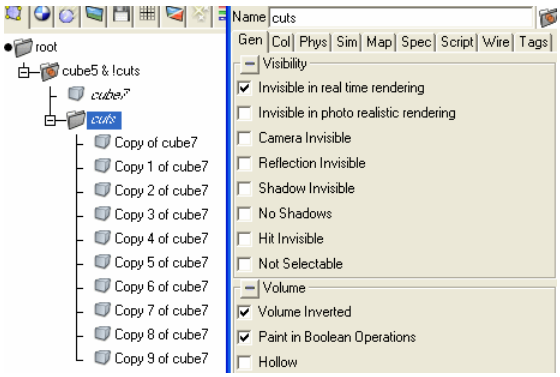
9. Select the Boolean tab of the toolbar.

10. Select the 'Remove' icon. Leave all other values to default and accept.

The Boolean operation now cuts the thin cubes from the original cube. This is dependent on the selection order - always select the object you want to cut FROM, then the object to be cut OUT.

Note that we can now add object freely to this "Cut" level and they all cut out from the original cube. Typically, if only 2 objects are used in a Boolean operation, the Cut object is set the attributes: invisible in wire, and volume inverted.

By dropping these objects to a level before applying the Boolean operation, the level inherits these attributes and all objects underneath inherit the level's properties. Hence we can also change the level's color to change all the cut colors. This ability to add objects to the level without having to set the Inverted volume attribute allows us to take this example further:



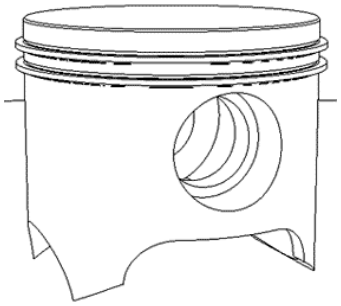
Adding materials to a cut object level allows us to create all sorts of interesting effects - For example, having a brick outer wall, then cutting holes in the wall with wood mapped cubes would yield a wooden window sill in the walls.

It is important to note that Booleans can include other Boolean objects. To create a piston ring, one creates a cylinder, cuts a hole in the middle, then uses that object to cut out from another cylinder, then duplicating the first cut ring to create a piston with a ring cut and piston ring inside. Such nesting of Booleans is limitless - cutting and shaping an object until the final form is realized.

Primarily analytical objects are used as they are closed shapes and the volume is easily inverted. The polyhedron shape is of particular use - it is an 'analytical freeform' shape. By using polyhedrons creatively, one can create almost any shape.



Attached is a relatively simple scene making good use of Boolean operators. Starting from a simple cylinder we "chiseled" out the form of a piston. Thinking in terms of subtraction is the key - How you want to "Cut" the object to obtain the shape you desire. Often this means cutting the cut tool - as with this scene. Examine how chamfers were cut from the shape with cones, then cut again from the piston cylinder. Note that this object was created solely from analytical objects - predominantly cylinders.



A useful technique for modeling complex forms such as this is to duplicate many of the objects from the original cylinder. This creates objects in the correct transform space and often only requires simple scaling or rotating.

When attempting to visualize the shapes inside a Boolean - creating the cut shapes OUT of the Boolean level is useful, placing them in the correct location, then dragging the icon into the "Cut" level.

Boolean representations can often be quite confusing and the Wire and Open GL representations quite inaccurate. Remember that these are merely previews of the object - renderings show the final form. Working in wireframe is by far the fastest method with Booleans - as the Open GL view is constantly being updated, computation times can sometimes be too slow for complex Booleans. Working in wireframe is best with complex scenes.

Under the object properties/wire tab of a Boolean level is a tag "AND wireframe off". This can be set to turn off the automatic Boolean wireframe representation if needed.

Problem solving Booleans

Occasionally, after much cutting and pasting between levels - errors can appear. These errors can either be strange artifacting - which is quite possibly the result of two shapes exactly on top of each other. Or Booleans looking completely wrong. If this is the case, the easiest method to check your objects is to drag half of the objects out of the Boolean level and render (Often the wireframe representation is enough). If the Boolean looks OK then slowly add various shapes back under the Boolean level until you can pinpoint which shape is causing the problems (drag the icons out of the Boolean level and turn off render.) The problems are most likely due to having 'Invert Volume' on. Check the problem objects for unusual attributes under the 'Object properties/Gen' tab.

One must be very careful with the placement of cutting objects. Sometimes duplicates may appear unnoticed by the user and strange artifacting may result as two identical objects may be cutting another. The finite accuracy of computers allows us to place objects in exactly the same positions, but when cutting two objects, it is imperative that the cut object is slightly away from the edge or corner to allow at least some cut or no cut at all. Otherwise strange results might appear. This is also applicable when mapping scope or finite materials.

When mapping Boolean objects, it is often a good idea to drop the Boolean level to another level, then apply the material.

Lattice Mapping

The lattice mapping system allows you to attach an object to another object called *lattice*. When the lattice object is modified, also the attached object becomes modified accordingly.

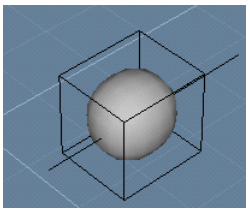
Any object can act as a lattice object. For example, you can attach a number of logos to a NURBS curve and deform the curve to control the positions of the logo objects.

The idea is that a complex geometric object, such as body of a character, can be controlled easily using a simpler object, such as a skeleton.

Tutorial level: Medium

1. Create a SDS sphere.

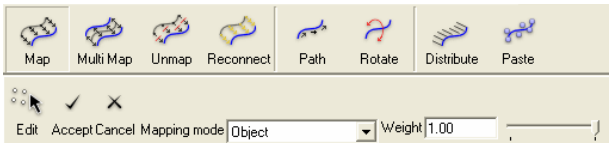
2. Create a horizontal straight NURBS curve consisting of four control points and passing through the sphere.



A NURBS curve passing through a SDS sphere

3. Multi select both objects, the SDS sphere first, the curve second. Make sure that you select the objects in this order because the Mapping tool maps the first selected object to the last selected object.

4. Click the Map tool in the Lattice Mapping tab of the toolbar. You can now see a couple of mapping specific options in the control bar. These default options are just fine, so Accept the tool.



Apply the Map tool to the selected sphere and curve objects

We have now mapped the sphere to the curve. Select the curve and move it. The sphere is moved with the curve. If you rotate the curve, also the sphere gets rotated. How about if you want the curve to affect only the position of the sphere, so that the curve does not affect the direction of the sphere in any way? To do this:

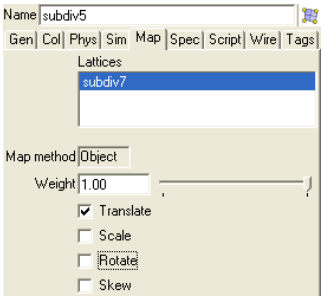
1. Select the SDS sphere.

2. Open the property window and go to the Map tab.

3. Reset the Rotate check box.

If you now rotate the NURBS curve, the sphere is only moved. In other words, the curve controls only the position of the sphere.

If you want to use the curve to control the orientation of the sphere, not the position, just reset the Translate check box and set the Rotate check box.



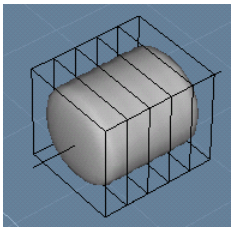
Rotate check box reset. The curve affects only the position of the sphere.

Pointwise mapping

In the previous example, we mapped a sphere to a curve using object mapping option. Therefore, the lattice curve controlled the four properties of the object space of the sphere: position, size, orientation and skew.

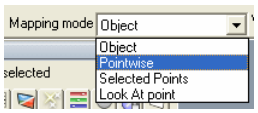
To allow lattices to fully deform objects, point mapping should be used instead of object mapping.

To map an object to a curve using pointwise mapping:



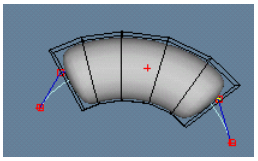
A NURBS curve passing through a SDS cylinder

1. Create a freeform object (such as a SDS cylinder, which containse enough points to allow interesting deformations). Create also a NURBS curve passing through the mesh just like we did in the previous example.



Use Pointwise mode

2. Select both the cylinder and the curve in that order. Click the Map tool. Now you can again see the mapping options in the Control Bar. Set the Mapping mode field to Pointwise and accept the tool.



NURBS curve bent. The mapped SDS cylinder is bent accordingly

3. Now select the curve and enter the Edit mode. Point edit the NURBS curve. The mapped mesh is deformed with the curve.

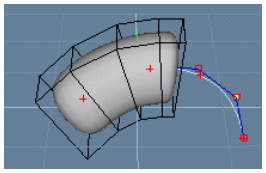
Modifying mapped objects

If the position of the mapped object is not correct, you can simply move it into the new position. For example:

1. Select the SDS cylinder we created in the previous example and turn off the Edit mode if it is still on.

2. Activate the Move tool.

3. Set Target=LatticeCoords from the tool control bar.



SDS object moved along the curve

4. Click a point on the lattice curve. When you move the mouse, you can see how the SDS cylinder moves along the curve. When a desired position is found, click the mouse again.



Note

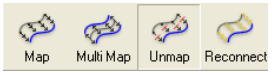
When moving mapped objects, the first point entered for the Move tool is very critical. Make sure you click over the curve so that the move tool has a good reference point to start moving the target along the curve. If the first point for the move tool is heavily displaced from the curve, the moving route in the curved lattice space is not intuitive, making the operation hard to control.

Detaching mapped objects from lattices

You can unmap an object from a lattice object by using the Unmap tool. For example, to detach the SDS cylinder from the curve.

1. Select the SDS cylinder and the curve objects (in this order).

2. Activate the Unmap tool and click the Accept button in the tool control bar.



Unmap tool activated

If you now modify the curve, the SDS cylinder is not affected in any way.

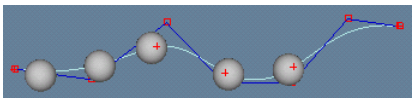
Mapping multiple objects to a lattice

You can map any number of objects to a single lattice object.

There are two ways to accomplish this. First, you can multi select all the objects and map them to the desired lattice object at once. Just make sure the lattice object is the last selected object when accepting the Map tool.

Or, you can select and map objects to a lattice object one by one. You can repeat the Map tool any number of times for the same lattice object.

Let's map a number of spheres to a curve.



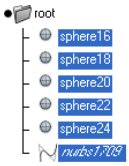
A NURBS curve passing through five spheres

1. Create five small analytic spheres and a NURBS curve passing through the spheres.

2. Hold down the Shift (multi selection) key and multi select all objects by clicking their names on the select window one by one: spheres first and the curve last.

3. Activate and accept the Map tool with the default options.

Modify the NURBS curve. The spheres are modified with it.



Select the NURBS curve last by Shift-clicking the objects from top to bottom

Swapping lattice objects

The Reconnect tool detaches objects from their current lattices and attaches them to a new lattice object.

For example, a sphere moving along a NURBS curve can be reconnected to a skeleton to get the sphere to move along the skeleton.

Note that only the lattice object is switched. All other mapping information remains untouched.

This tool can be very useful when you need to map complex objects to curved lattices. For example, you might want to map a NURBS mesh to move along a cylinder surface. Instead of trying to bend the mesh to match the shape of the cylinder, simply map the mesh to a planar surface and animate it there. When the animation is ready, reconnect the animated mesh to its actual cylinder lattice.

Let's see how this works.

1. Create a NURBS mesh (rectangle) consisting of 5 x 5 points and map it to a bigger analytic rectangle using pointwise mapping.

2. Activate animation recording and move the mesh in the initial position in the beginning of the animation.

3. Move the time slider to the last frame and move the mesh to another position. You can also single point edit the mesh for animated deformations.

4. Reset animation recording and play the animation to see how the NURBS mesh moves in the space defined by the analytic rectangle.

5. Now, create an analytic cylinder (actually a NURBS cylinder or sphere or any other geometric object with a curved surface will do).

6. Select the animated NURBS mesh and the new cylinder, activate the Lattice Mapping/Reconnect tool and click Accept.

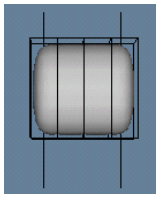
Play the animation to see how the mesh moves along the surface of the cylinder.

Multi mapping

The previous examples used only one lattice object. However, a single object can be mapped to any number of lattice objects.

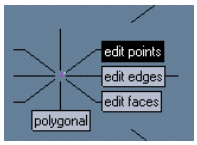
For example, you can use multiple NURBS curves to control different parts of a single SDS mesh.

Let's see how this work. Let's create a SDS cylinder whose left side is mapped to one curve and the right side is mapped to another curve.



A SDS cylinder and two vertical NURBS curves

1. Create a horizontal SDS cylinder (use for example 4*8 points) and two vertical NURBS curves at both ends of the SDS cylinder.

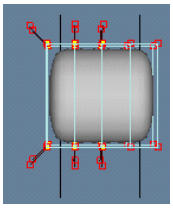


Enter point editing mode

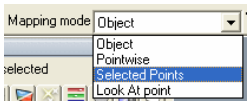
2. Select the SDS cylinder and enter point editing mode (select Edit Points from the compass).

3. Select all the left side points of the SDS cylinder using dragging.

4. Multi select the SDS cylinder (with its left end points selected) and the left side NURBS curve in this order.



Select points in the left half

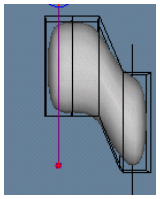


Apply the Map tool with the Selected points mode

If you look at the View window now, you can't see any points selected in the SDS cylinder. This is normal. The points are still selected, although not highlighted because of multi-selection situation.

5. Activate the Map tool and set the Mapping mode option to Selected points. Then Accept the tool.

Try moving the left side NURBS curve now. The left end of the SDS cylinder is moved with the curve.



The leftmost curve moved upwards

6. Now repeat the steps 3...5 for the right side points of the SDS cylinder. In other words, select the right side points of the SDS cylinder, then multi select the right side NURBS curve and apply the Map tool again with the Selected points option.

Weighted mapping

The Weight parameter controls how strongly lattice objects affect the mapped points or objects. This is often needed in multi mapping.

For example, you can map a point on two lattices so that the first lattice affects the point with 30% strength and the other one affects the point with 70% strength. Or you can define weights so that the shorter the distance to a skeleton, the stronger the skeleton affects the points/objects.



Mapping with 50% strength

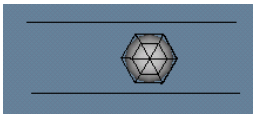
Let's multi map a SDS sphere to two NURBS curves and experiment with weights:

1. Create a SDS sphere and two identical NURBS curves passing through the sphere (duplicate the curve to get two identical curves).

2. Select the sphere and the first curve just like we did in the previous examples and apply the Map tool using the Object mapping mode.

3. Map the sphere on the second curve.

You have now mapped the sphere on two curves using equal full weights.



A sphere multi mapped on two NURBS curves with 1.0 weights

4. Now move the curves. The sphere is always located exactly between the NURBS curves, because both curves affect the sphere with an equal weight.

To make the first curve affect the sphere only 10%:

5. Select the sphere, open the property window and go to the Map tab.

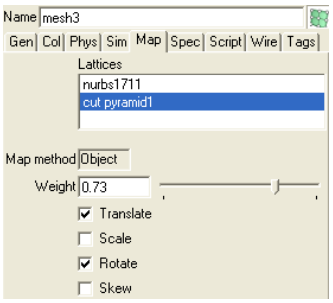
6. The Lattice field shows you the name of the first curve. Set Weight to 0.1. The sphere is moved towards the second curve.

7. Now, pick the name of the second curve from the property window's lattices list. Then set also the weight of the second lattice curve to 0.1. The sphere is moved back to the center of the curves because both curves affect again the mapped object with equal (0.1) weights!

Weights are like strings, which attempt to pull the mapped points to the position defined by the lattice objects. The higher the weight, the stronger the string. Therefore, if you have only one lattice object, weights are irrelevant, because even the weakest string will pull the mapped object or point. There is no other string resisting it.

Using the property window to edit mapping parameters

The Map tab in the Property window controls mapping parameters.



Lattice mapping properties

- *Lattices:* The list of lattice objects controlling the selected object. Other gadgets of the tab display to properties related to the lattice defined by this gadget.
- *Map method:* Displays the type of the lattice mapping (object or pointwise)
- *Weight:* The control weight of the lattice. In pointwise mapping, each point can have an individual weight, and the gadget shows and changes the weight of the selected points (set the mapped object to edit mode if you want to change point weights).
- *Translate, Scale, Rotate, Skew:* These gadgets define which components of the target object's space are affected by Object type lattice mapping.

Using the vertex painting tool to control weights

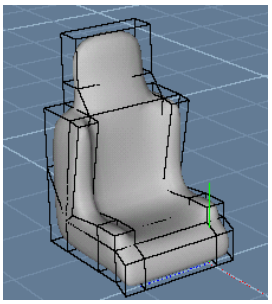
You can use the Vertex Painting tool (this multi-purpose tool is located in the Materials tab) to control weights of pointwise lattice mapping. Select the mapped object and then activate the tool. The tool control bar shows a selector Property to be painted. Pick Weight from it, select a suitable brush size, define the weight and paint over the desired points.

Recording and Executing Macros

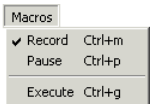
When you use the program in the macro recording mode, all actions are recorded into the *current macro*. Current macro can then be executed, for example, to speed up a certain modeling task. You can also rename the current macro to create *a named macro* and bind the named macro to keys, buttons or menus.

This tutorial shows how macros can be recorded and executed. The GUI chapter of the manual explains how to bind macros to the components of the user interface.

In this tutorial, we create one chair and record a macro, which duplicates and moves the chair. Then we execute the macro a couple of times to get a row of chairs.



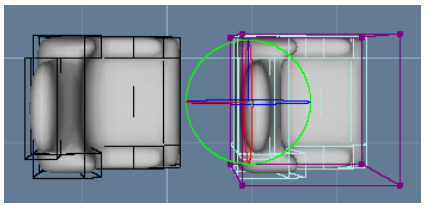
A chair for a train



Recording on

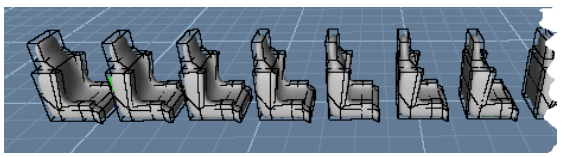
1. Create a chair and make sure it is selected.

2. Switch to the top view and activate macro recording by selecting the Macros/Record pull down menu. This operation clears the previous current macro and enables macro recording.



Duplicate and move the chair

3. Select the Edit/Duplicate pull down menu and use the Move tool to move the duplicated chair to get a second chair.



Duplicate+Move macro executed a couple of times

4. Now reset macro recording by selecting Macro/Record again. Then select Macros/Execute a desired number of times to create the rest of the chairs.

Rotate a view

When macro recording is activated, by default all applied actions are recorded to macros. For example, you can record a macro, which rotates the view window.

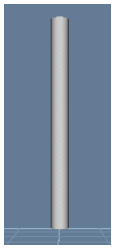
1. Set macro recording on

2. Rotate the view window by holding down the Alt key while dragging with the right mouse button.

3. Clear macro recording. Now hit the key combination 'Ctrl g' to execute the current macro (or select the pull down menu Macros/Execute). The view window is rotated.

Spiral stairs

You can record a macro, which first moves an object, then rotates the view and finally moves the selected object again. Such a macro contains two kinds of actions: those which are applied to the view window and those which are applied to the selected objects.



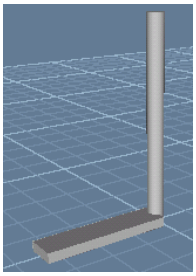
A center pillar for spiral stairs

Sometimes you don't want view specific events to be recorded into macros.

Macro filters prevent actions applied to a certain target, such as the view window, being recorder to a macro.

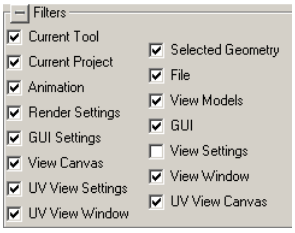
This tutorial demonstrates how to create spiral stairs using macros. The idea is that each step of the stairs can be created by duplicating, rotating and moving the previous step. These three functions can be recorded to a macro. The stairs are then built by executing the macro a desired number of times.

1. Create a cylinder representing the axis of stairs.



The first step created

2. Create one analytic cube representing the first step.

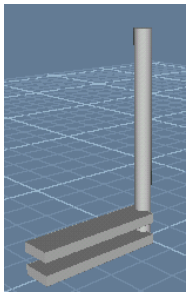


Disable macro components, which affect view's options

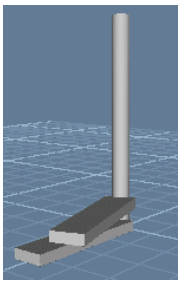
3. Go to the Select window's Macro tab (the third last one) and open the Filters frame. By default, all macro targets are enabled. Reset the View Settings check box to disable events, which change view orientation.

4. Activate Macro Recording. Make sure the step object is selected and apply the Edit/Duplicate pull down menu. This records the Duplicate operation to the current macro.

5. Switch to the side view. Because macro recording for view window is disabled, view rotation is not recorded into the current macro. Move the duplicated step upwards using the Move tool.



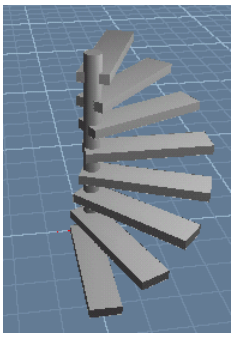
Duplicated step translated upwards



Second step ready

6. Switch to the top view and rotate the duplicated step using the Rotate tool.

We have now created the second step by duplicating, moving and rotating the first one.



Macro executed six times

7. Reset macro recording and make sure the second step is still selected. Select the Macros/Execute pull down menu to create the third step for the stairs. Repeat 'Macro/Execute' a desired number of times to get all steps created.

The stairs are now ready.

Entering points from a macro

In this example, we create a simple macro, which just enters a number of points through the view window. Before you try this example, make sure the View Settings macro target is activated in the select window's Macro/Filters controls.

1. Set macro recording on

2. Click the left mouse button several times in the view window. Nothing happens because you haven't activated any creation tool. However, these mouse clicks are still recorded to the current macro.

3. Disable macro recording. You have now recorded a macro, which simply feeds a number of points to the current tool, whatever it is.

4. Click the NURBS/Curve tool and hit the key 'Ctrl g'. A smooth NURBS curve appears to the view window. Click Accept to finish the tool.

5. Activate the NURBS/Curve tool again. This time set the order to Polygonal. Then execute the current macro and click Accept. A polygonal NURBS curve is created through the recorded points.

Activating tools from a macro

You can record a macro, which only activates a tool, but lets you interactively define the necessary points for the tool. The purpose of such a macro is to save the work of selecting the suitable tool options.

1. Set macro recording on.

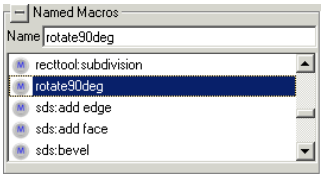
2. Click the desired tool and set suitable tool options through the control bar.

3. Disable macro recording and cancel the tool. Hit the keys 'Ctrl g' to activate the tool with the specified options.

Creating named macros

The latest recorded macro is named as 'current' by default. Whenever you record a new macro, the old current macro is destroyed.

If you want to keep a macro more permanently, you have to rename it. This can be done as follows:



Named macros

1. Record a macro as described in the previous examples.

2. Go to the select window's macro tab and open the Named Macros frame.

3. Activate the Name field, enter a new appropriate name and hit Enter.

Executing a named macro

You can execute any named macro by dragging & dropping it into a view window.

If you want to repeat a macro multiple times, just select it from the named macro list, enter a suitable count to the Repetitions field and click the Apply button.



Repeat a macro 5 times

Particles

Particles are objects with a very simple geometric description. Due to their simplicity, they are memory efficient and a large number of them can be created.

Phenomena such as water drops, fur, fog and fire, can be simulated using particles. Particles can be rendered in a variety of ways: as small dust particles, lens flares or foggy spheres, to name a few examples.

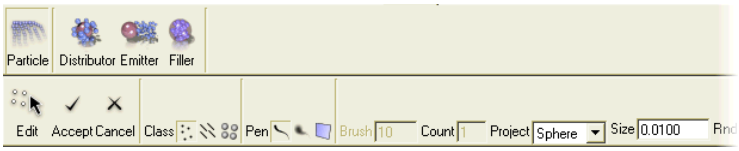
Despite their simplicity, the rendering engine of Realsoft 3D can make particles look very interesting. For example, particles can be textured just like other geometric objects.

The Particle tool allows you to create particles interactively.

To create a flock of particles:

1. Activate the particle tool.

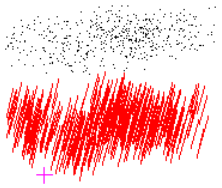
When you activate the tool, the Control Bar shows you a number of particle specific options.



Particle tool activated. The Control Bar shows you creation specific options.

2. Specify desired options from the Control Bar. To get started, just use the default values. Then create particles interactively through the view window by holding down the left mouse button while moving the mouse.

When done, accept the tool by clicking the right mouse button and selecting Accept from the opened pop-up menu.



1D and 2D particles

You can also airbrush particles on the surface of the selected objects. To do this, set the Surface option of the particle tool. Orientation of 2D and 3D particles will then be determined by the orientation of the underlying surface. This option allows you to create easily fur and hair for objects.

Below is a description of particle tool specific options.



Particle tool options

Class

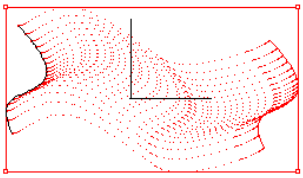
The type of the created particles. Currently three kinds of particles can be created:

- **1D**. 1D particle is defined by a single point. Rendering engine can render 1D particles as points, circles, stars etc. 1D particles look the same from all directions.
- **2D**. A 2D particle consists of two points. So, it has properties such as direction and length. Real world objects like water drops, which have a symmetry axis, can be rendered using 2D particles. Also, polarity can be simulated (particles can have positive and negative ends).
- **3D**. The geometry of 3D particles is represented as a coordinate system. In other words, 3D particles can be scaled, rotated and skewed. They can be used for rendering objects, which look different from all the three main directions. Please note that the current scan line shaders do not render 3D particles visible.

Pen

Currently there are three pens available:

- **Pencil**. This creates one particle for each mouse movement.
- **Brush**. The number of particles created for each mouse movement can be defined through the Count field.
- **Object**. This creates particles along a selected NURBS curve. The number of particles generated for each mouse move event can be defined using the Count control.



A NURBS curve used as a pen for creating 1D particles

Brush

This field controls the radius of the airbrush particle tool. The size is measured in pixels.

Count

Defines the number of particles created per each mouse movement when using the brush and selected object pens.

Project

This option controls the depth coordinate of particles added by the brush pen. Three choices are available:

- Disc: Particles are created into the input plane. The result is a flat particle cloud.
- Sphere: Each mouse movement adds a new spherical cloud of particles around the mouse position defined by the input plane. The diameter of the particle cloud is defined by the Brush gadget.
- Surface: Particles are added to the surface of the selected objects. The surface also defines the orientation of the particles.

Size Defines the size of created particle items.

Rnd Size

Randomizes the size of created particle items. 0=constant size, 1=fully randomized size.

Rnd Dir

Randomizes the direction of created particle items. 0=fixed orientation, 1=quite random orientation dominated by original direction, 10=totally random orientation.

Modifying particles

Particles can be modified just like any regular objects, such as spheres and NURBS curves.

For example, you can modify the created particle flock by using the move tool with Collision Deformation option set:

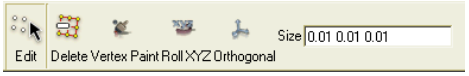
1. Select the particle object and activate the move tool with the Collision deformation option.



2. Move the particle object against the sphere.



When you select a particle object, the control bar automatically shows you particle specific tools.



Particle tools

You can also enter the edit mode to single point edit the particles.

You can use any of the available tools, such as move, rotate or bend, to modify the selected particles. You can duplicate points simply by dragging them while holding down the copy modifier key 'Ctrl'.



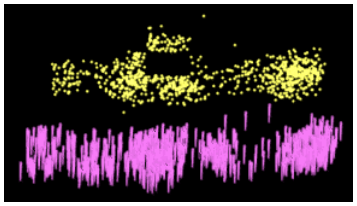
Some of the particles selected in edit mode

Properties of particles

All common object properties, such as name, color or mass, can be modified through the property window.

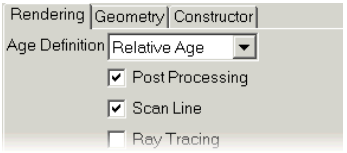
The color of a particle object can be defined in the usual way using the property window's Col tab.

The default scanline shading of particles does not look especially pretty.



1D and 2D particles rendered

Particle specific options can be controlled through the Property Window's Spec tab. It includes three sub tabs, which are described below.



Particle specific properties

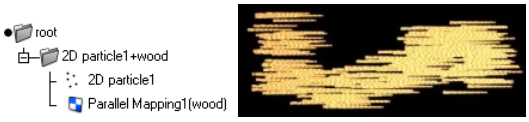
Rendering tab

Particles can be rendered using *scanline*, *post processing* and *ray tracing* techniques. These are not exclusive: you can render a scanline particle and add a post effect on top of it. For more information about how to define visual appearance of particles, see the Post processing chapter.

Scan line rendering

The color, optical and other surface properties of scanline rendered particles can be defined in the same way as those of any regular objects - by using the material system.

For example, you can texture map scanline particles using the wood material of the default material library.



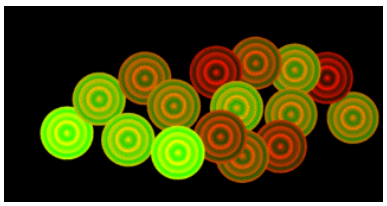
Texture mapped 2D particles. Wooden particles consist of two sub objects: 2D particle and wood.

To texture particles:

1. Create a set of particles. Select the particle object.

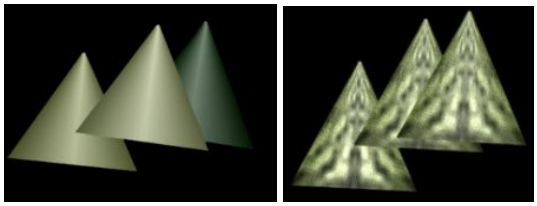
2. Drag&drop a material from the Materials tab of the Select window into the view window. This creates the object hierarchy shown in the example image.

Particles have a well defined surface parameter space, and therefore you can use UV mapping to map materials to particles. The UV space of 1D particles is two dimensional. The 'u' coordinate is defined in the radius direction. It is zero in the center of a 1D particle and runs to one with the radius. The 'v' coordinate increases with the order number of individual particles: the first created particle has the v coordinate value 0, and the last particle has v equal to one.



UV mapped particles. Color changes by the 'u' coordinate.

The UV space of 2D particles is three dimensional. 'U' runs along the length of a particle. 'V' runs perpendicular to the direction of the particle (from the axis towards the edge). The 'W' coordinate runs with the order number of the particle. Using the UV coordinates, you can define materials, which for example, make 2D particles look like 3D cylinders, cones, or hairs. And of course, 2D particles can also be bump mapped, clip mapped, they can be transparent or reflect light.



2D particles whose color fades by v coordinate making them look like 3D cones, and bump mapped 2D particles

Scan line rendered 1D and 2D particles are visible in ray tracing. For example, other objects can cast shadows on particles.

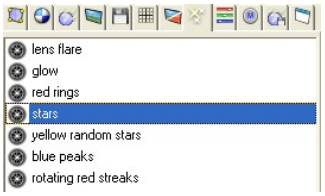
The only drawback with scan line rendered particles is that they are quite memory consuming. However, a high render box count in rendering decreases memory consumption. Summary: defining material properties for scanline particles does not differ much from defining material properties for any regular geometric object such as a sphere or a NURBS mesh.

Post Processing

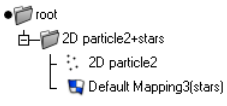
In addition to scan line rendering, also post processing can be used for rendering particles. To do this:

1. Select the desired particle object.

2. Drag&drop a post particle effect from the select window into the view window.



Drag&drop the desired particle effect into the view window to map it to the selected particle objects

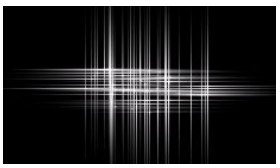


Star filtered particles

As expected, this creates the object hierarchy shown in the adjacent image.

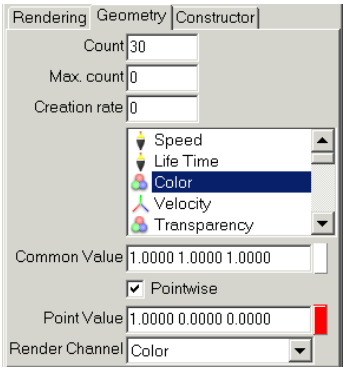
Rendering this shows a number of stars.

Note that several rendering methods (scanline & post processing) can be used simultaneously. For example, you can use the scan line particles to render hairs and then use post processing to make the hairs glow.



A star effect mapped to 1D particles and rendered

Geometry tab



Geometric particle properties

The Geometry tab controls particle specific properties such as size and length. It includes the following fields:

Count

Controls the number of particles. If you increase the count, position and other properties of added particles are computed by interpolating the existing particles.

Max Count

Limits the maximal number of particles. The value is used, when a particle constructor is enabled, making the count vary over time.

Creation Rate

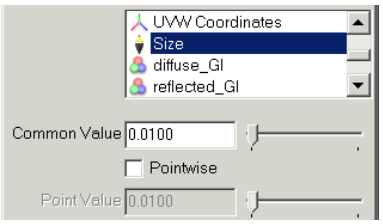
Defines how rapidly particle count increases, when a constructor is enabled. The value defines the total number of added particles during the whole animation. Note that the rate value can be animated, making the density of particle flow variable. A negative value stops particle recreation (particle count starts decreasing, when existing particles die).

Properties

This list view allows you to select and define particle specific properties, such as life time and speed.

The rest of the fields in this tab are used for defining initial values and other properties for the selected attribute.

Common Value



Uniform size defined for 1D particles

This gadget shows the common 'per object' value of the selected property.

For example, to modify the size of 1D particles:

1. Select the Size property from the Attribute list.
2. Enter a desired radius to the Common Value field.

Pointwise

Each property can have a uniform value or it can be defined per particle. Uniform means that the all particles in the object share the defined property.

By checking the Pointwise box, you can assign an individual value for each particle. For example, to modify the size of particles:



2D particles here and there

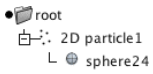
1. Select the Size property from the Attribute list.
2. Enter a suitable base value to the Common Value field. This value will initialize per point values.
3. Check the Pointwise option.
4. Set the particle object to edit state and select some particles from the view window.
5. Enter a new radius to the Point Value field.

Render Channel

This gadget selects a channel for holding the value of a pointwise property in rendering. For example, you can define that a dynamic float channel Age holds the value of the Life Time attribute of particles. Then you can make a simple VSL material, which reduces the color of particles by the Age channel.

Constructor tab

Constructor



2D particle object contains a sphere

This control allows you to control the shape of the particle flock using other geometric objects. Distribute over surface option distributes the particles over the surface of the sub objects. To use this feature:

1. Create some 2D particles and create a sphere inside the particle object.

2. Select the particle object and open the property window. Go to the Spec/Geometry tab and activate the Distribute over surface constructor.

3. Modify the sphere. Particles remain distributed over the sphere.


2D particles distributed over a sphere

Animating particles

Particles can be animated in the same way as any geometric objects.

Key framing and Morphing Particles

Particles can be key framed the same way as other geometric object. For example, to morph particles, simply turn on animation recording and single point edit the particles.

Path Animations

Particle objects also support path animations.

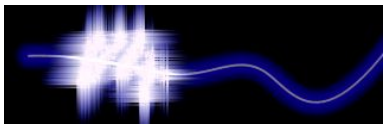
For example, you can use pointwise mapping to map particles to a path to achieve animated path deformations:

1. Create a set of particles with the Particle tool.

2. Create a NURBS curve.

3. Select the particle object and then the curve object and apply the pull down menu Animation/Path/Pointwise.

4. Select the particle object and drag&drop a desired post particle effect (such as the star effect) into the view window. You get a bunch of stars traveling along the path.


Particles following a path

Simulations

Particles are often animated using the simulation system. For example, you can use a number of 1D particles as meteorites acting under the influence of a gravity field caused by a planet.

To do this:

1. Create a number of 1D particles representing meteorites.

2. Create a sphere representing a planet.

3. Multi select the sphere and the particles and apply the tools Animation/Simulations/Gravity and Animation/Simulations/Collision Detection. Open the Property Window for the sphere, go to the simulation tab, and change Gravity and Collision Detection to Cause.

4. Select the particles and apply the tool Animation/Physical Properties/Warm butter.

5. Set the frame count to 500 and play the animation.



Note

Using the simulation system to control particles does not differ from using the simulation system to control any regular object. However, note that particles don't have volume, and therefore two particles cannot collide. Particles can collide with true 3D objects, such as spheres, without any problems.

Creating Leaves Using 3D Particles

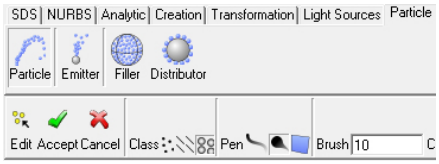
This tutorial shows how to simulate tree or plant leaves using textured 3D particles in Realsoft 3D V8. Thanks to Bernie den Hertog for the original ideas and proof reading.

Originally created by Robert den Broeder, August 25th 2004. Last update April 14th 2017

Beginner/intermediate level.

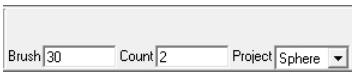
Creating the particles

Click on the Creation tab and select the Particle tool.



Particle tool selected, set to create 3D particles using the Brush pen.

Select the 3D particles and select the Brush as pen type. This allows you to literally spray particles around. Next set Brush (which defines the brush radius in pixels) to 30, Count to 2 and Project to 'Sphere'.



Spherical brush projection selected

These settings allow you to spray just a few 3D particles with every mouse click. This also causes the particles to be sprayed around using a spherical brush, meaning that the particles are not all in the same 2D plane, but spread in all 3 dimensions, only limited by the size of the brush.

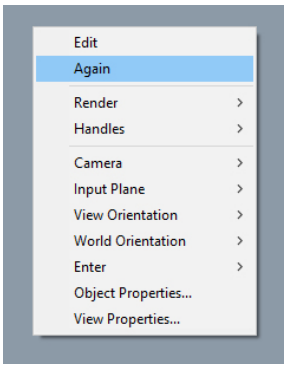
To see how this spherical brush works, just click and hold the right mouse button for about 10 seconds and then accept the tool. In your view window the particles will form a 3D sphere about the size of your brush.

A nice feature is the counter in the View window, which shows you exactly how many particles are created:



The counter shows how many particles are created

Press Ctrl + Z to undo last action. Then, in the view window, press the right mouse button and select 'Again' from the pop-up menu. This restarts the particle tool with the same settings as you had the last time you used it.



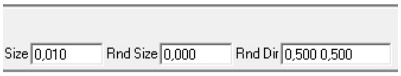
Select 'Again' to restart the tool with the same settings as last time

The particle tool has some additional controls for size and direction. Set 'Size' to 0.01, 'RndSize' (Random Size) to zero and RndDir (Random Direction) values to 0.5 and 0,5.

If RndSize is set to zero, all particles will have the same size. If RndSize is set to a higher value, more and more particles will have a different size.

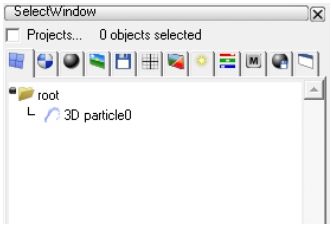
The same goes for RndDir: if set to zero, all particles will have the same orientation towards the camera; change that value and their orientation will change randomly. The randomization is controlled with 2 values. The first value defines how much the axis of each particle changes randomly. In case of 3D particles, the controlled axis is the z axis. The second value controls the random rotation around the axis.

The greater the values, the more randomized direction each particle gets. For the first component, value 1.0 already randomizes the direction 180 degrees, but the original direction still dominates the average of directions. For the second random component, value 1.0 gives a totally random rotation around the axis.



Settings for Size, Random Size and Random Direction

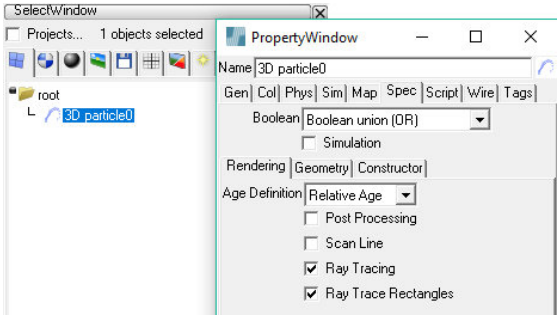
All settings are OK now. So, spray just a few particles around by clicking and releasing the left mouse button at various locations in the view window. If you click and hold the left mouse button, more particles are added to the scene. The counter exactly shows how many particles are created. When done, accept the tool. A 3D particle object is now added to the hierarchy in the SelectWindow.



From spheres to rectangles

If we now render the scene, all 3D particles will render as spheres. This is the default setting for 3D particles. Because we want to make the particles look like leaves, we need the shape to change from spheres into rectangles. This is simply done by changing the properties of the 3D particle object. Double click on the 3D particle object in the Select Window to open its properties. Go to the Spec tab of the Property Window. On the Rendering tab make sure that "Post Processing"and "Scan Line" are unchecked and then check "Ray Tracing" and then "Ray Trace Rectangles". This last option ensures that all particles are rendered as rectangles, which can be textured. In this case, we can

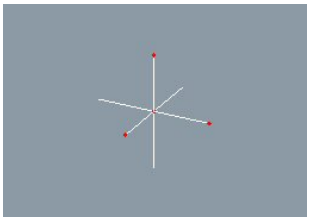
texture them now to look like leaves. And because we set the RndDir option as well, not all rectangles (leaves) will face the camera, which enhances the effect we need.



Ray Tracing and Ray Trace Rectangles options set on the Rendering tab of the Property Window

When the Ray Trace Rectangles option is set, all 3D particles are rendered as equally sized squares. This is of course due to the settings we used when creating them (RndSize = 0). It is also possible to edit the size and shape of each individual particle inside the 3D particle object after we created them. Attributes like size, shape, position and orientation of the individual particles can even be key framed for animation.

To change the size of a single particle, just select the particle object in the Select Window and press the Space bar on your keyboard to go in edit mode. The 3D particles have 3 points that can be edited and one central point that can be used to move the particle around:



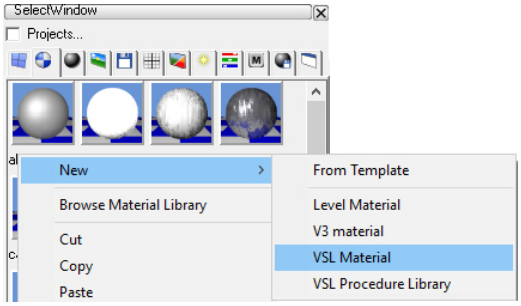
A single 3D particle in edit mode

Modifying the shape of a particle is simple: just drag the red dots on each axis until the particle reaches desired shape. To move a particle, just click or drag the red dot in the center of the particle. As usual, transformation handles allow you to move the particle to a new location in your view window. If you wish to move several individual particles, hold the Shift key and click at the center dot of each particle you wish to move. You can also use the Drag Select method to multi select particles. To modify all selected particles simultaneously, use the Drag Select method to multi select particles and then apply the Noise tool, which is located in the Transformation toolbar .

Creating the Leaf material

To make the particles look like leafs of a plant, we need to texture them. This requires a material, which we will create from scratch.

In the Select Window, click the Materials tab and create a new VSL material.



Create a new VSL Material in the Materials tab of the Select Window

It will be added at the top of the material list. You will see a black square representing the new material and it has a name like Material64. Double click on the new material to open the Property Window. Change the name, call it Leaf1. Check the Preview box. We need a leaf texture and an alpha map of the same leaf. So why not use these two images?



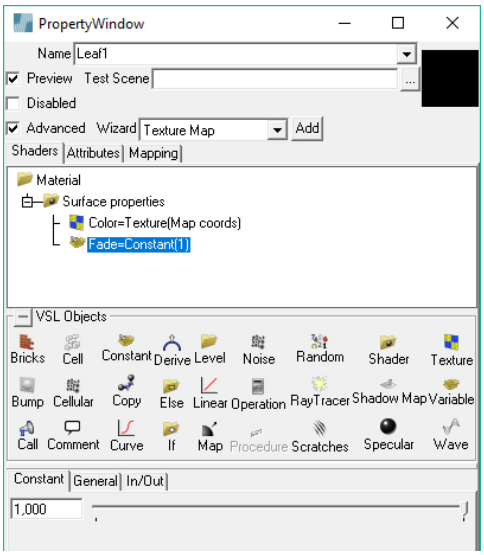
The color map (D_longleaf.gif) and the alpha map (A_longleaf.gif) of our leaves (Thanks Bernie den Hertog!)

Just save these two images to your favourite directory and use them in this tutorial.

Use the texture wizard and select the coloured leaf texture (D_Longleaf.gif). Leave the Tile X,Y and Grade X,Y boxes empty. Check the Advanced checkbox to get access to the VSL statements. Click the Color=Texture(Map coords) instruction. At the lower part of the Property Window go to the Texture tab and tick the Edge X and Edge Y boxes. This stops edge feathering from appearing.

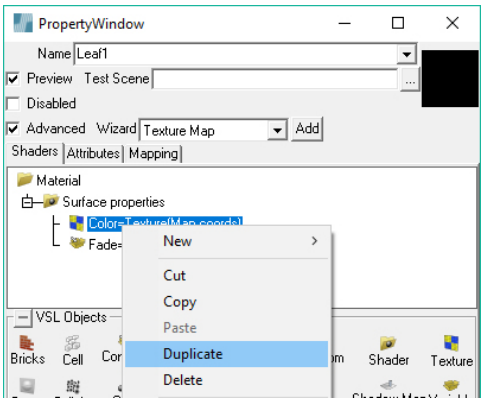
Select a Constant object and drag that into the Surface properties shader. In the lower section of the Property Window there are three tabs, related to the Constant object. We start with the third (most right hand) tab. The third tab is called the In/Out tab. Make sure that 'Output' is set to 'Surface : Fade'.

The second tab is called General. The Operation should be set to '=' (the default value) and do not check the other boxes. Finally, go to the first tab (Constant) and set the Constant value slider to 1. Notice that the preview window now turns completely black!



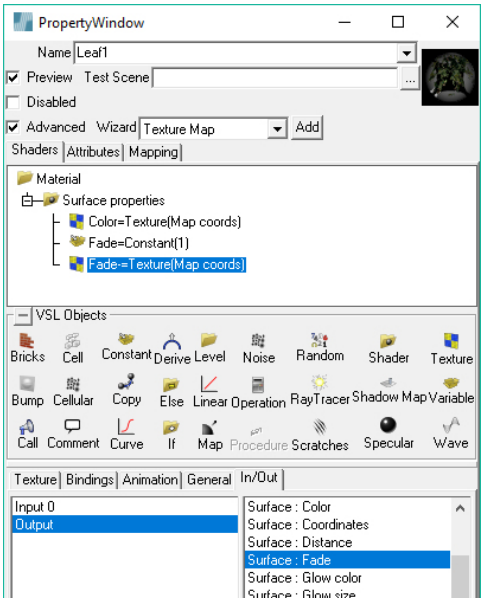
The Constant object added, output set to Fade and Constant value set to 1.

Select the Color=Texture(Map coords) instruction, press the right mouse button and select 'Duplicate' from the pop-up menu. The duplicate instruction will be placed at the bottom of the hierarchy.

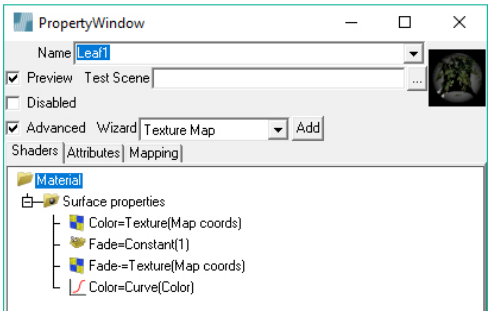


The Color=Texture(MapCoords) instruction will be duplicated

Select the duplicate instruction. In the lower section of the Property Window there are now five tabs. Go to the Texture tab and select the alpha map image (A_longleaf.gif). Next, click the General tab. Set Operation to ' - ' (the minus sign) .Finally, go to the In/Out tab. Click on 'Output' and set the Output to 'Surface : Fade', simply by selecting it from the list on the right side of the window. Note that the preview window is immediately updated.



Finally, drop a VSL curve object into the hierarchy, which gives us color control over the texture. At the In/Out tab of the curve object, set both 'Input 0' and 'Output' to 'Surface : Color'. The shader should now look like this:



The material is completed.

And then...

Map the Leaf1 texture to the 3D particles using a default mapping. Add a simple infinite rectangle to have a background and add a light source. Press render, and you should get something like this...



The first render: leaves-textured particles!

Now this scene is more interesting when not all leaves have the same colour. This is where the VSL curve object becomes handy. Perform the following steps:

1 Create just a few new 3D particles using similar settings as before and now also add some random size, like 0.3.

2. In the Material tab of the Select Window duplicate the Leaf1 material, call it Leaf2

3. Modify the color of the texture in Leaf2, using the VSL curve. This is done as follows. Make sure the material Preview box is checked. In the VSL Property Window, select the Color=Curve(Color) instruction. In the lower section of the window a graph appears, showing a red line. Click in the middle of the red line. This creates a knot point. Drag the knot point slightly diagonally upwards. Notice the color change in the preview box. When happy with the new color, close the Property Window.

4. Select the new particles in the Select Window and map Leaf2 on these particles using the default mapping.

5. Render again and it already looks more interesting!

Another example

In this part of the tutorial we put our knowledge to use and show some more of the sheer power of 3D particles!

First create an analytic cube, which will act as a brick wall. Then create a brick texture and map that to the cube (wall) using a parallel projection. Don't forget to enable the Roll option (Properties Spec tab) for the parallel mapping object. You get something like this:



Nice brick wall eh?

Then it's time to grow some plants on that wall. Again we will use our 3D particles with the same Leaf1 material applied to it. Only this time we will use a different projection when creating the particles. This allows us to let the

plants follow the structure of the wall. Activate the particle tool again, select 3D particles and the brush pen, just like before, only this time with the following settings:



Settings for the 3D particles. Notice the Surface projection and the size of the particles.

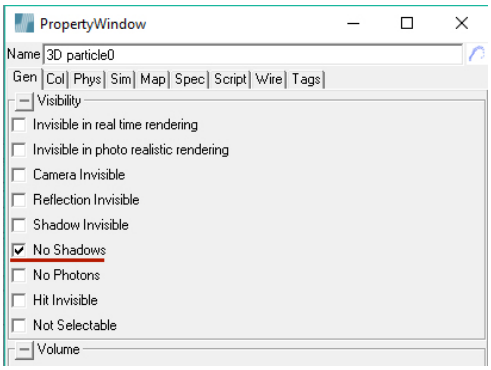
Make sure the cube (wall) is selected in the Select Window. Now spray just a few particles at a time right on the top left corner of the wall and make your way down. Notice that there will be no particles added to the scene when you do not spray on the wall. This is because we set the projection method to 'Surface' and we selected the wall as target.

Then open the properties of the 3D particle object, set the Ray Tracing and Ray Trace Rectangle options at the Spec tab. Apply a default mapping of the Leaf1 material. Then repeat the particle creation process with the same settings and then apply the Leaf2 material and render. I got something like this:



Wall with some leaves growing over it.

When using many textured particles, you may find yourself waiting for your render to finish. Setting the 'No Shadows' option in the Property Window of the 3D particle object could be a real time saver!



No Shadows option set, to increase render speed!

Camera

The rendering engine needs a camera object to define position of the camera, viewing angle and focal length, among other things.

A view window contains an internal camera, which can be controlled interactively with several view navigation functions available through the keyboard, view property window and the view control bar. In addition to this, the user can create any number of camera objects into the hierarchy of geometric objects. View windows can fetch their orientation and other properties from these camera objects.

This tutorial examines how various camera properties can be managed.

Creating a camera

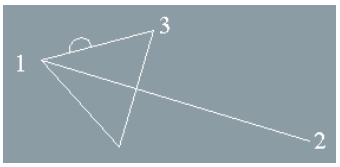
1. Activate the Camera tool from the Creation tab of the Toolbar.



The Camera tool selected

2. Define the camera geometry by entering three points through the view window. These three points define position, aim point and viewing angle for the camera.

You have now created a camera object. You can modify the camera just like any other geometric object using the modify tools (Move, Rotate etc.) and the property window.



Enter three points in a view window to define the position, aim point and viewing angle for the camera

View's camera

A view window uses an internal camera object to define its orientation. When you rotate, pan and zoom the view, you actually modify the position, direction and distance of this internal camera object.

Instead of defining geometry for a camera object under construction by clicking three points, you can also use geometry from a view's internal camera. Using this feature, you can easily save the current state of the view camera to a camera object and restore it whenever needed.

To create a camera object which matches the current orientation of a view window:

1. Activate the camera tool.

2. Accept the tool immediately without entering any points. The created camera matches exactly the orientation of the current view window. (Including orthographic views, i.e. non perspective views!)

Working with multiple cameras

You can add any number of cameras to your project using the two creation methods described above.

To fetch the viewing angle and other settings of a camera object into a view window, simply drag & drop the object from select window to the view window.

If you have included multiple cameras to a project and want to start file rendering or animation preview, you have to first define which camera is the Current camera. By animating the Current camera option, you can also jump from one camera to another during animation.

To make a camera the current camera:

1. Select the camera in the Select window

2. Click the Current icon in the tool control bar

The camera backdrop tool

You can define backdrop images using the Backdrop tool. A backdrop image is a plane defined in a camera's space.

Because a backdrop image is a true 3D surface, other objects can reflect it, cast shadows on it and so on. For example, you can create a magnifying glass, which really magnifies the backdrop image.

Any number of backdrop images can be created per camera. Only the backdrop images of the Current camera are active in file and view rendering.

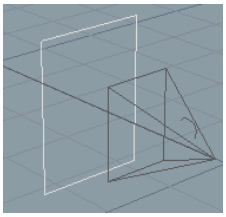
The Backdrop tool allows you to create most commonly needed backdrop effects easily.

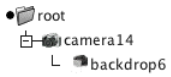
To define a constant background color:

1. Click on a camera in the Select window. This makes the Backdrop icon show in the control bar. Now click on the Backdrop icon and define the desired color to the Color field of the tool control bar



Backdrop tool activated, constant white color defined





A backdrop object shown in the view and the select windows

2. Select Accept on the desired view window.

This automatically creates a new camera object with a backdrop object. If a camera object was selected when applying the tool, the backdrop object is inserted into it (a new camera is not created).

To preview the backdrop in a view window, simply render the view. If the scene contains multiple cameras, make the camera Current first.

Backdrop images are represented as planes in view windows. You can move a backdrop object like any other geometric object. However, you can't shear or scale it. The backdrop always remains perpendicular to the focal axis of the camera.

Backdrop tool sets up a backdrop image, which always renders with the specified color. Light sources don't affect the shading of the backdrop image in any way. This result is achieved by setting the actual diffuse color of the backdrop to black and assigning the desired RGB value to the Illumination channel. To change the unshaded constant color of a backdrop:

1. Select the backdrop object.

2. Open the property window and go to the Col tab.

3. Select Illumination from the Attribute list and enter a new color to the value field below the attribute gadget.

Activating a lens flare effect for a camera

To activate a lens flare for a certain camera:

1. Select the camera.

2. Open the property window. Go to the Spec tab. The Lens Flare gadget shows the contents of the current library of post particle effects (you can add new alternatives using the Post Particle Effects tab of the select window). Select a suitable alternative.

To preview the lens flare, drag & drop the camera from the select window into a view window and click Render.



Note

You can render a lens flare in a view window without a camera object. The view property window contains a camera property gadget (for controlling view's internal camera) and you can select the effect directly using it.

Camera lens flare is a camera specific effect. Lens flares are automatically created by the light sources, which are directly visible from the camera. You can add other kinds of lens flares to your scenes using the post processing system of Realsoft 3D. You can, for example,create thousands of particles and render them as lens flares. This approach provides much more control than the 'automatic' camera lens flare feature. Multiple effects per light position can be defined and flares can be added only to certain light sources by placing a flare-mapped particle to the desired positions.

Depth of field

Depth of field is another camera specific effect. A camera object provides you with an intuitive interface to this effect.

To activate depth of field for a camera, just check the Depth of field check box in the property window's Spec tab. The aim point handle of the camera allows you to specify the distance at which scene renders sharp. By animating the position of the aim point, you can easily animate depth of field. The amount of depth of field depends on the focal length of the camera and F-stop value just like in real world cameras.



Note

You can also control the depth of field effect without a camera object. The module, which renders the effect, is included in the Default Effects configuration, which is available in the Post Image Effects tab of the select window. If you select the included depth of field effect and clear the Camera defined option using the property window, you can control its parameters independently of camera objects. See the Post Processing chapter for more information.

Animating cameras

Camera objects, just like any other geometric object, can be animated using various methods. For example, you can key frame cameras, use gravity to attract them, map them to a motion path and so on.

The key framer allows you to animate the position and orientation of the cameras easily. The program uses an advanced quaternion interpolation system to achieve smooth and intuitive camera rotations.

In many cases, either the camera rotates about its aim point or the aim point is rotated about the camera position. If you want to achieve the former effect, just drag the pivot point of the camera into the aim point. Then rotate the camera using the rotation handles to key frame it.

When using motion paths, banking needs some special attention. The path animation tool provides the most common banking control types. For example, you can select airplane type banking.

Lattice mapping system allows you to set up camera tracking animations easily. For example, you map the aim point of a camera to a moving object to make the camera follow the object automatically.

Connecting a view and a camera object

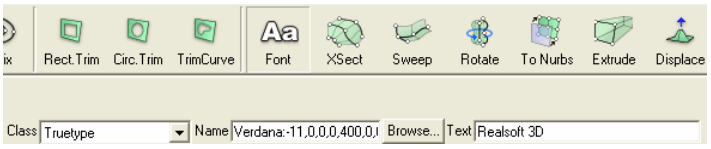
The view window supports a Tracking option. In other words, you can lock a view window to the current camera object. Whenever the camera object is moved or otherwise changed, the view window updates accordingly.

You can activate this feature from the view property window's Camera tab. If you set Tracking = Track animation play, the camera is updated only when animation time is changed.

If you set Tracking = Track always, the view window and the camera become completely synchronized. If you rotate the view window, the camera object is rotated accordingly. If the camera object changes, the view changes with it. Take care using this option when setting keyframes with animation recording ON, make sure to turn this feature OFF when you have finished recording. Otherwise you may ruin your carefully set camera keyframes by changing the view later.

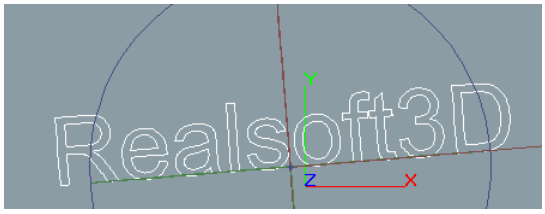
Fonts

The Font tool creates NURBS curves representing a given text string and font type. The generated curves can then be extruded to 3D or used as animation paths, etc.



The Font tool selected

1. Activate the Font tool. It's located in the NURBS tab of the control bar.

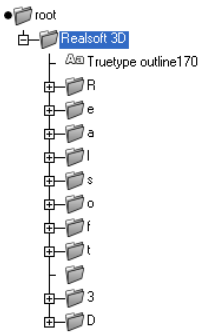


Enter two points through the view window to define a base line

2. Define a TrueType font and some text through the control bar.

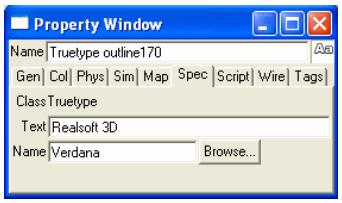
3. Enter two points through a view window to define a base line for the outline curves.

This creates the object hierarchy shown in the example image.



Font construction consists of a Font Manager object and font curves

You can modify the generated text any time by selecting the font manager object (such as 'truetype outline69') and editing its text property using the property window.



Changing Text of a font manager object

For example, to change the Realsoft 3D text to 'Yes':

1. Double click the 'Truetype outline69' object in the select window. The property window is opened, or simply activated if already open.

2. Go to the Spec tab and enter a new string to the Text field.

The font manager object updates the curves to reflect the new string.

To extrude the NURBS curves, simply select all levels below the font manager containing the single letter curves and select NURBS/Extrude from the main toolbar.

Rotator Object

The rotator object creates a number of new objects by copying, rotating and translating its sample objects. For example, you can create the balls of a ball bearing using the rotator object.

To create balls for a ball bearing:

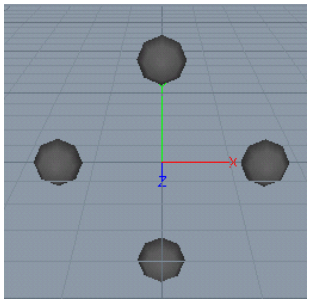
1. Create one analytic sphere

2. Select the sphere and activate the Rotator tool (it's in the Modifier tab). The tool control bar now shows you a number of rotator tool specific options.



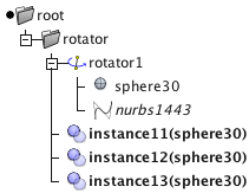
Rotator tool activated

3. Enter a point through the view window to define the center of rotation. This accepts the tool. You now have four spheres rotated about the defined center point.



Four spheres created by rotator tool

In the select window, you can now see a rotator object and three instances. If you click the '+' handle of the Rotator object, you will find the original sphere and a NURBS axis object inside it. The rotator object duplicates the sphere about the created axis object.



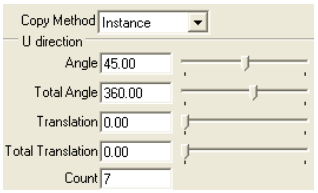
A sphere is instanced and rotated three times by a rotator object

Modifying rotator objects

You can change the number of object copies created by a rotator object (and other options as well) afterwards. For example, to change the number of spheres to 8:

4. Select the rotator object and open the property window.

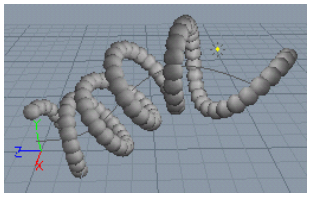
5. Go into the Spec tab and set the U Direction/Angle field to 45 degrees. Then set the Count field to 7. You now have eight spheres (the original sphere + seven instances).



Changing the U direction angle and number of copies

In addition to rotation, the rotator object can translate objects along the defined axis. Value 1 translates spheres by the full length of the axis. You can also replace the straight NURBS line with any curved object to get spheres translated and rotated along a curved path.

If you only define 'translation' and set rotation to zero, the rotator object can be used as an alignment tool.



Translation and rotation along a curve

The rotator accepts also a two-dimensional object, such as a rectangle, as an axis. In this case you can control the rotation/translation in two directions. For example, you can create arrays of objects using two perpendicular translations.

Animating a rotator object

Rotator objects can be animated the usual way: just set animation recording on and modify the rotation options, such as the number of copies, to key frame them. You can also animate the axis and sample objects freely.

Tube Object

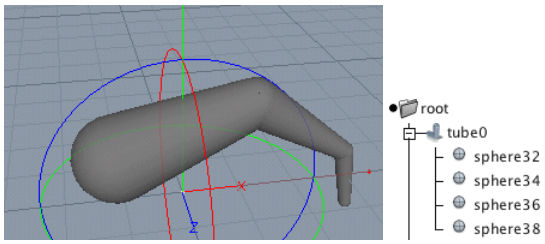
The tube object connects analytic spheres using cylinders and cones.

To create a tube object:

1. Select the New/Tube from the pop-up menu of the select window. This creates a level object.

2. Create the desired number of analytic spheres inside the tube object. The object hierarchy in the select window should look as in the picture below.

The spheres you dropped inside the tube level object are instantly connected with cylinders and/or cones.



A tube connects four spheres with a varying diameter

Editing tubes

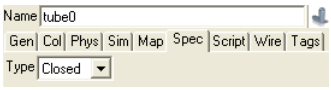
You can freely modify the position and size of spheres inside a tube object. The shape of the tube is updated instantly.

However, you should not stretch the spheres. The reason for this is that it is not possible to connect ellipsoids using cones and cylinders. If you need to create a tube with an arbitrary shape and cross-section, use NURBS or subdivision surface tools instead.

The property window can be used for selecting the desired tube type. For example, to create a closed tube:

1. Select the tube object

2. In the property window/spec tab, set Type to Closed.

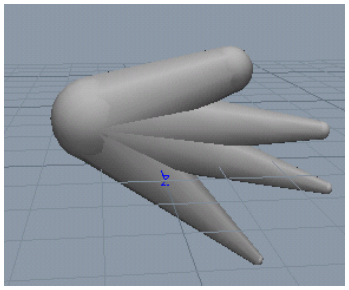


Selecting Closed tube

This connects the first and the last spheres so that the tube becomes a closed loop.

The Star type tube connects the first sphere with the other spheres creating a star shaped tube system.

Tutorial project: 'tutorprojects\modeling\tube\star'



The Star option connects five spheres

Animating tubes

You can animate the shape of the tube by animating its sub objects.

For example, you can create a fancy animation by using collision detection and gravity to control the sub objects as follows:

1. Create a tube object consisting of a set of spheres.

2. Select the spheres. Then select the menu Animation/Simulations/Gravity from the menu bar.

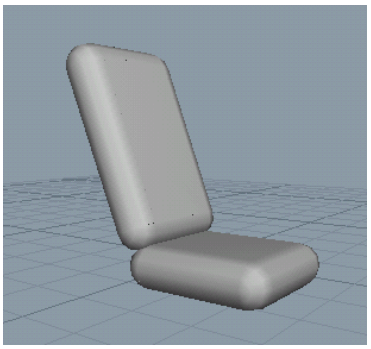
3. Select the menu Animation/Simulations/Collision Detection.

4. Play the animation.

Tutorial project: 'tutorprojects\modeling\tube\tube through colliding spheres'

Instances

Consider a situation where you need to model hundreds of chairs for a cinema theatre. The chairs all look the same, only their position and orientation vary.



A sample chair

Instances allow you to manage this kind of situations efficiently. An instanced object duplicates its source object. When the source object is modified, all instances referring to it will change accordingly. Instances can be transformed or deformed using any available modify function including single point editing. Instances can be instanced, instances of instances can be instanced, etc.

For example:

1. Model a chair of a movie theatre.

2. Make sure that the chair is selected. Then select the pull-down menu Edit/Instance.

Modifying Instance Objects

You can modify an instance object using all common tools, such as delete, duplicate, move, rotate and scale:

3. Select the instance of the chair and activate the Move tool.

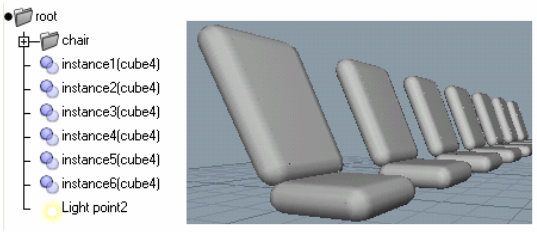
4. Move the instance to the right side of the original chair.

We can use the macro system to achieve more instances quickly:

5. Select Macros/Record from the pull-down menu.

6. Select the instance object. Select Edit/Duplicate. Then move the duplicated instance to the right.

7. Select Macros/Record again from the pull-down menu to switch macro recording off. Go to the Macros tab of the select window. Enter a suitable value to the Repetitions field and click Apply.



A row of chairs

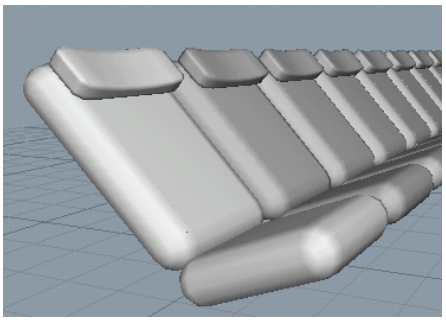
Modifying the Source Object of Instances

Let's improve the chairs. Because of instancing, only one chair needs improvements:

8. Add a neck support to the sub hierarchy of the original chair.

9. Select all sub objects of the original chair and rotate them.

All instances change to reflect these modifications.



More comfortable chairs

Properties of instances

- Instances can be freely animated.
- Levels with many sub objects can be instanced.
- Instances spare RAM memory used for storing object geometry and wire frame representations. Instances also consume less disk space than 'real' geometric objects when saved to a file.
- When the Render as Instance option is unset, the photorealistic render engine converts the instance to an actual geometry (for speed reasons). In this case, instances do not decrease rendering time memory usage. If Render as Instance is active, raytrace rendering becomes very memory efficient but usually a bit slower.
- Instance property gadget has a control which specifies how transforming the source object or its parent levels affects the instance. By default, the Ignore Matrix option is set, meaning that you can move, rotate and scale the source object independently of its instances. To modify both the source object and the instances, you have to make the modification to the sub hierachy of the source object (sub objects or geometry points in edit state). If the option is cleared, the instance system takes into account the modifications performed to parent levels. If you modify the parent level of an object, instances referring to that object become modified as well. In many cases this is the expected behavior. However, there is one side effect: if an instance and its source object are moved into a new hierarchy level, the transformation history defined by the parent levels may change. Therefore, the evaluation of modifications performed to the instance may also change. The following example clarifies this: Create a level, and put an object into it. Create an instance of the object to the same hierarchy level and move it 0.1 m to the right. Rotate the level 45 degrees. Both objects rotate as expected. The instance still contains the information 'source moved 0.1 m to the right', but moving happens now in the rotated parent space. Now create a new level and drag & drop both the instance and the source to it. The instance jumps to a new position, because the new parent level is not rotated.

Constraints

Assume that you have to create a sphere whose 'x' coordinate always stays right in the middle of the 'x' coordinates of the two other spheres. However, in 'y' and 'z' directions, you should be able to move the sphere without any restrictions.

Constraint objects allow you to set up this kind of relations between objects. To define the relation outlined above:

1. Create three analytic spheres and name them as 'sphere1' 'sphere2' and 'sphere3'

2. Select the sphere1 and activate the Constraint tool. You can access this tool from the menu (Tools/Modifier/Create Constraint). Click Accept.
3. In the select window, drag and drop sphere2 and sphere3 into the created 'constraintX' folder.

4. Select the constraint object, open the properties window and go to the Spec tab. Set Command Language to JavaScript and enter the following program:

```
var i = 0;
var mid_x = 0.0;

for(i = 0; i < 2; i++)
{
    child = Self.GETSUBBYORDNUM(i);
    if(!child)
        break;
    p = child.GetCenter();
```

```
    mid_x = mid_x + p.x;
}

if(i != 0) {
    mid_x = mid_x/i;
}

p=Target.GetCenter();
p.x=mid_x;
Target.SetCenter(p);
```

The constraint defines all attributes of the associated objects in its own attribute space.

Field Evaluator

The field evaluator object provides you with useful information of an object, such as true velocity, spin, acceleration and angular acceleration. The animation chapter presents how to use the field evaluator to create an object whose muscles are deformed by acceleration. This chapter will show you how to evaluate material properties in any point in space and use the evaluated property to control other objects.

Tutorial level: Medium

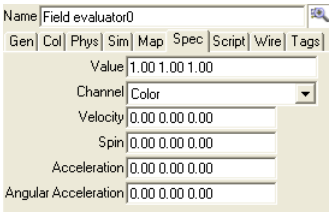
Evaluating color

To find out the color in a certain point in space:

1. Create a field evaluator object by selecting New/Field Evaluator from the pop-up menu of the select window.

2. Move the origin of the field evaluator object to a desired position.

3. Open the property window and in the Spec tab, set the Channel field to Color. You can read the color from the Value field.



The color in the specified point is 1.00, 1.00, 1.00

If you want to find out the transparency, just set Channel to Transparency.

This kind of tool is naturally useful for examining object properties. Furthermore, you can use the evaluated information to drive choreographs.

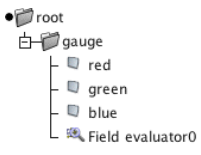
Gauge

Let's animate object lengths based on the color of a given point in space. Three rectangles represent a sort of gauge object. When we move the gauge to a desired position, the lengths of the rectangles immediately show us the color in that point.

There is no such built-in feature in the program. However, we can create it easily using the field evaluator object. The idea is simple: first we model three rectangles, red, green and blue. Then we make their length depend on the evaluated color.

1. Create a level object consisting of three rectangles (red, green and blue). Create also a field evaluator object.

2. Select the field evaluator object and in the property window's Spec tab set the Channel attribute to Color. This way we ask the field evaluator object to evaluate the color property.

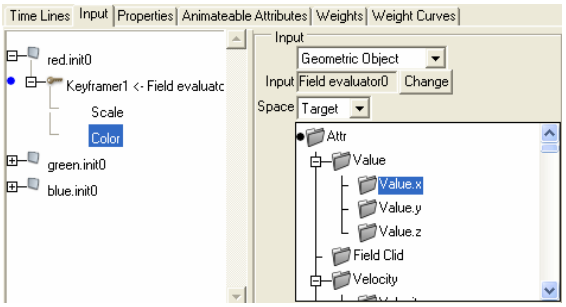


A gauge object consisting of three color indicators and a field evaluator object

3. Animate each rectangle so that they grow longer as the time proceeds. You can also rotate the rectangles so that they behave more like a traditional speed gauge. Or, perhaps you want to animate the color of the rectangles.

The only thing we need to do now is to use the field evaluator to control the choreographs of the rectangles. This can be achieved as follows:

4. Open the choreography window and select the Input tab. Select the first keyframe choreography, which modifies the red rectangle. aDrag the field evaluator object from the select window and drop it over the selected choreography. Then select the Value.x from the input attribute list. Repeat this step for the remaining two choreographs. Then switch to the Time Lines Tab. Select again each choreography and set its Start and End values to 0 and 1, because the RGB values typically range between 0 and 1. The gauge is now ready.

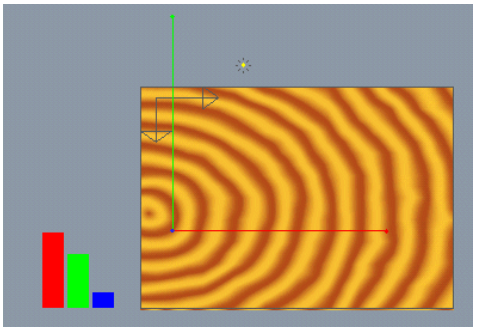


Use the field evaluator's Value components to drive the choreographs

To test it, just create a material mapping object (such as a parallel map) anywhere in space and move the object space origin of the field evaluator to the desired point. The lengths of the three rectangles change immediately to show you the color in that point.

Note: you can freely move the field evaluator in the object hierarchy. For example, to measure the color of a wooden sphere, just drag & drop the field evaluator object into the 'wooden sphere' hierarchy.

Tutorial project: 'tutorprojects\animation\fieldeval\gauge'



The origin of the field evaluator object moved over a textured object. The gauge object shows us the color

Metaball Object

Technically speaking, a metaball object is a special kind of level which interprets its sub objects as a density distribution and computes a connecting boundary surface to a given density iso value. In practice, a metaball 'melts' its sub object together. The melting property makes the metaball object especially suitable for representing liquids.

The first metaball tutorial shows you how to set up a metaball system using a special construction tool.

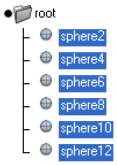
Tutorial level: Beginner

Example project: 'tutorprojects/modeling/metaball/snowman'

1. Model a snowman using some analytic spheres. As the name might indicate, a metaball object melts only spheres, so do not use other kinds of objects. However, you can freely stretch the spheres if necessary.



A snowman consisting of six analytic spheres

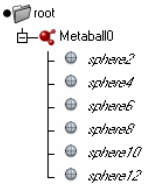


Select the spheres and activate the metaball tool

2. Select all spheres. Go to the Creation tab of the toolbar and activate the Metaball tool.



Metaball tool

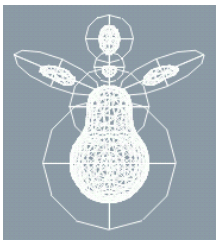


The hierarchy of a metaball system

A new object called 'Metaball' appears in the object hierarchy. Open its sub hierarchy; all the original spheres are placed there.

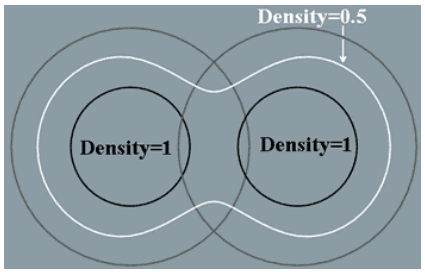
Controlling iso value

The default properties of the 'snowman' metaball are such that the boundary surface shrinks radically from the original spheres.



Shrunk snowman

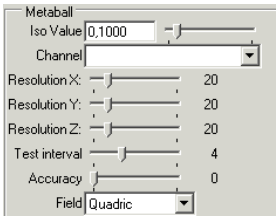
The surface generated by a metaball level follows a given density value in space. Each sphere inside a metaball level adds its contribution to the density field: the density is zero at the outer edge of the sphere, and reaches its maximum value of one at the middle of the sphere radius.



A cross-section of a metaball system consisting of two spheres

3. Select the metaball level. Open the property window and go to the Spec tab.

The Iso Value field defines the density value, which the metaball surface follows. The default Iso Value is 0.5. Change it to a smaller value, say 0.1, and you will see how the snowman rapidly gains weight.



The metaball specific properties

Editing metaballs

The sub objects of a metaball level define its construction history. You can modify the sub objects and add new spheres, and the shape is updated automatically.

4. Select the head sphere of the snowman.

5. Activate the Transformation/Scale tool and expand the sphere. The metaball surface follows the modification.

6. Select the Metaball level. Apply the Make Current menu from the select window's popup.

7. Create two new analytic spheres below the snowman (legs). The spheres are inserted to the current level (= metaball), and the shape changes accordingly.

Alternatively, you can create the spheres at the root level, edit them a bit and finally drag&drop them into the metaball level.

Converting metaballs to another geometry type

Sometimes it may be desirable to convert a shape modeled using metaballs to another form. For example, a metaball surface does not have well defined UV coordinates, which can make applying textures and materials difficult. Fortunately, it is easy to convert a metaball shape to subdivision surface form. The converted shape can be edited further and textured using the SDS tools.

Let's experiment how this happens using the snowman shape:

8. If Metaball is still the current level, select the root level and apply Make Current to it. Then select the metaball level.

9. Go to the SDS tab of the toolbar. Activate the To SDS tool. Make sure that Use Default Resolutions and Subdivision Object options are set. Then click Accept.

10. Note that when the To SDS tool is used, it converts all sub objects to SDS objects as well. We do not need any of these sub objects converted. Hence select all the newly created SDS objects except the last one and delete them. This leaves us with the metaball SDS only.

11. You can now delete the metaball object.

12. Select the created subdivision object and open the property window. Go to the Spec tab. Change Rendering/Type option to Smoothen to Nurbs, and set Quality to the minimum value 1. Close the property window.

13. The remaining step is polygon reduction by converting triangles to rectangles. To do this, select the function Methods/Subdivision Surface/Merge Triangles to Quadrangles.

A subdivision surface that consists of quads usually renders smoother than a triangular SDS mesh, because it is difficult to divide most surface forms to triangles using a symmetric pattern.

Metaball properties

The Spec properties of a metaball include the controls listed below.

Iso Value: The iso value of the density distribution. The generated surface follows this value.

Channel: Defines the channel, which modifies the density distribution. If set, materials that affect the metaball are evaluated to find out the channel value.

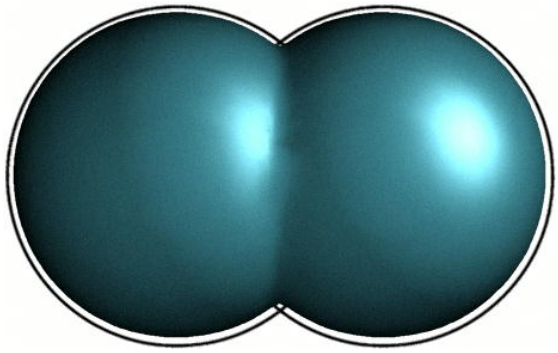
Expansion: Expands (or shrinks) the size of metaballs from their true geometric size.

Resolution X, Y, Z: These gadgets define the overall accuracy of density field evaluation for metaball surface computation. The higher the resolution, the more accurate and smooth the computed surface will be. On the other hand, increased accuracy means longer computation time and higher polygon count on the resulting surface.

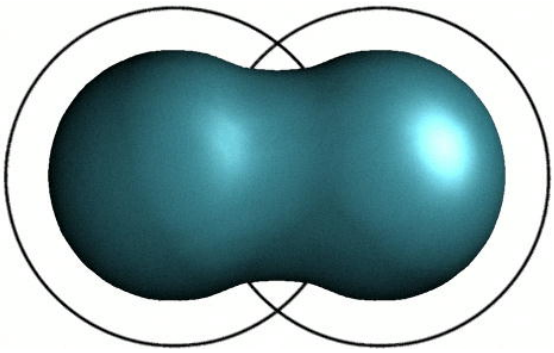
Test interval: An optimization control. If the metaball system does not include small details, or if the Resolution fields are so high that all details will be accurately sampled, the test interval value can be larger than one. Test interval defines how accurately the program examines points defined by the resolution values to find out places where the density really changes. If most of the space is empty, using this control can speed up metaball evaluation dramatically.

Accuracy: Defines how accurately a detected density iso value point is computed. The higher the value, the more exactly the surface follows the given iso value. The default accuracy level zero is sufficient for most situations.

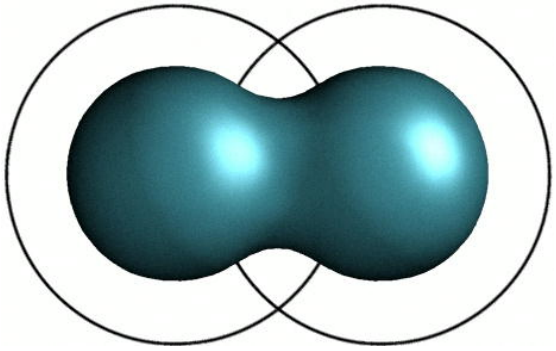
Field: The type of density distribution. Defines how smoothly the density inside a metaball component decreases from one to zero. The following four alternatives are available:



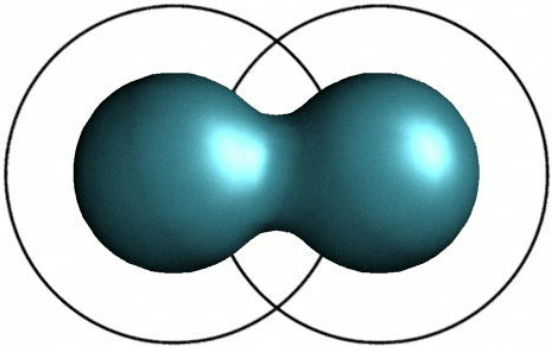
Volume: The density drops rapidly to zero at the boundary.



Linear: The density decreases at a constant rate.



Quadric: The density decreases smoothly to zero.



Cubic: The density decreases extra smoothly to zero.

4th Degree: The density decreases proportionally to the 4th power of the radius. The smoothest way to blend metaballs.

Rendering:

- Triset - the surface is rendered polygonially using the Marching Cubes Algorithm. This is usually the fastest way, but the result shape often suffers from various rendering errors which are typical to polygon models.
- Procedural - the surface is resolved during ray trace rendering without polygonization. This method renders a perfectly smooth surface, when the Accuracy setting is high enough. It is also extremely memory efficient.

Hierarchical: If set, the full sub hierarchy of the metaball level is examined to find metaballs, otherwise only the direct children.

Fluid simulation using metaballs



The next tutorial example, flowing water, combines fluid dynamics and metaballs. The idea is to use a creator object to produce new metaballs (= water drops) during the animation. First read the manual chapter Simulation if you are not yet familiar with the simulation tools of Realsoft 3D.

Tutorial level: Advanced

Example project: 'tutorprojects/modeling/metaball/metafluid'

1. Use the select window's New/Creator to add a creator object to the hierarchy.

2. Create an analytic sphere (radius about 5 cm) and drop it into the creator level. The sphere represents a water drop.

3. Open the property window and go to the Phys tab. Change the mass of the sphere to about 0.15 kilograms.

4. Switch to the Sim tab. Define Gravity = Affected and Fluid Dynamics = Affected for the sphere.

5. Select the creator object. Go to the Spec tab of the property window, and define (approximately) the following values:

- Creation Interval = 0.15 seconds. This value defines how much water comes out from the water pipe. Note: You may have to set the Time System in the Options/Window/Animation tab to 'seconds'.
- Life Time = 3 seconds. The lifetime should be long enough so that water drops have time to fall outside the rendered images.
- Velocity = '0.3 0.05 0'. This value defines how fast water bursts out from the pipe.

Note that these values represent only one possible combination creating a slow water flow. You can experiment and find other values that work as well.

6. Switch to the Phys tab and define a spin value '2 0 0' for the creator object. This changes the direction of water drops slightly during the animation, making it more interesting.

7. Activate the Simulation/Fan tool from the toolbar, and draw a fan object in a view window. Set Turbulence = 1 from the Spec tab of the property window. Then go to the Phys tab and set Density = 10 (dense air will enable slow motion effect). Change Fluid Velocity to 1 m/s.

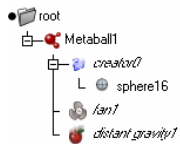
8. Start the Simulation/Gravity tool and draw a gravity line downward in a view window. Using the Spec tab, change Strength of gravity to 0.2.

9. Activate the Simulation option for the parent level of the created objects.

Now it is time to play the animation to check how the water drops move. It is faster to test the simulation part now when the metaball system is not yet included. Rewind the animation and adjust the above mentioned properties if necessary.

10. Select New/Metaball from the select window's popup menu. Drag and drop the creator, the fan and the gravity into its sub hierarchy.

The hierarchy of the scene is now ready. It is shown in the adjacent image.



The structure of the simulation

11. We dropped the simulation components to a new hierarchy level after initial experiments. Therefore, select the metaball level and using the Spec tab of the property window, turn on Simulation again.

12. With the metaball level selected, drag & drop the glass material from the material library to the view window.

Add a nice environment and lighting to the scene before rendering the animation - the kind of environment that gives a realistic look for transparent objects.

Material controlled density field

The regular spherical density field of metaballs can be modified using a material. The following example explains the required steps.

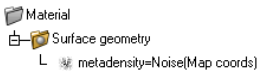
Tutorial level: Medium

Example project: 'tutorprojects/modeling/metaball/asteroid'

1. Create a group of analytic spheres. Select them and apply the tool Creation/Metaball.

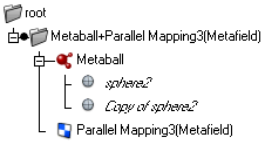
2. Go to the Channels tab of the select window and create a new Float channel. Rename it, for example, as metadensity.

3. Go to the Materials tab of the select window and create a new VSL material called as 'Metafield'. Open the property window and check the Advanced option. First drop a Shader object from the VSL objects frame into the material root folder. Change the type of the shader to Surface Geometry. Then drop a Noise object into the Surface Geometry shader. Change Output of the Noise object to metadensity.



Density field material

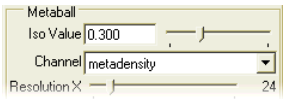
Leave the Base value of the noise to zero and Amplitude to 1, as the noise should vary between 0 and 1 just like the original density field of metaballs. Set the 1D option to obtain faster noise computation (density is a one-dimensional value). Activate the Smooth option, which produces more continuous surface distortion.



Density field mapped to metaballs

4. Select Map/Parallel from the popup menu of the material library. Hold the Shift key down to get an uniform map geometry and draw a faily large parallel map cube over the metaballs in a view window.

5. Go back to the object hierarchy tab of the select window and activate the metaball level, which is now inside a new level. Open the property window. Go to the Spec tab and select metadensity from the Channel gadget. You may also set Rendering to Procedural; that method can properly represent very complex surfaces. However, procedural rendering is often much slower than triangle based rendering. On the other hand, memory consumption is very low and the render quality is much better.



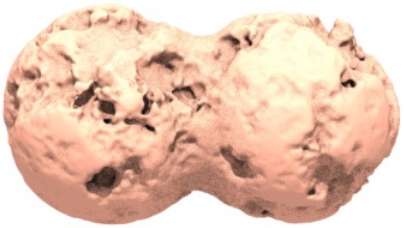
Metadensity channel modifies the density field of a metaball system

6. Adjust the Iso Value so that a suitable amount of the density field generates a surface around it.



Note

The total density field is the product of the spherical density fields and the material defined density field.



An asteroid defined by two metaballs and a density field material

Link Object

A link object points to a geometric object. It can be used where a geometric object would be evaluated.

The first link tutorial shows you how to evaluate a curve object through a link.

Tutorial level: Beginner

Example project: 'tutorprojects/modeling/link/linkpath'

1. Create a curve using the NURBS/Curve tool.



The Link tool selected

2. Select the curve. Go to the Creation tab of the toolbar and apply the Link tool. new object called 'link (nurbs3)' appears in the object hierarchy.

3. Create an analytic sphere near the first point of the curve.

4. Select the sphere and the link object in that order.

5. Go to the Lattice Mapping tab of the toolbar and activate the Path tool.

Click the Accept button.

6. Click the Play button on the Animation window to see how the sphere follows the curve.

Using a link object with particles

Tutorial level: Beginner/Intermediate

Example project: 'tutorprojects/modeling/link/linkparticle'

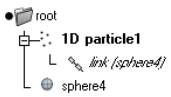
1. Go to the Creation tab and activate the Particle tool. Create a cloud of 1D particles and accept the tool.

2. Select the particle object. Open the Property window. Select the Spec tab, then the Constructor sub tab. Set the Constructor gadget to Distribute over Surface.

3. Create an analytic sphere.

4. Select the sphere. Apply the Link tool.

5. Drag & drop the link object inside the particle level object. The object hierarchy should be as in the adjacent image.



The object hierarchy

The particles are now distributed over the sphere's surface.

Using a link object with fluid dynamics

Tutorial level: Intermediate

Example project: 'tutorprojects/modeling/link/linkfan'

1. Go to the Creation tab and activate the Particle tool. Create a cloud of 1D particles and accept the tool.

2. Select the particle object. Open the Property window. Select the Spec tab, then the Constructor sub tab. Set the Constructor gadget to Emitter.

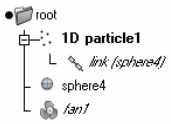
3. Select the Phys tab and set the Mass gadget (inside the General frame) to 0.01.

4. Select the Sim tab and set the Fluid Dynamics gadget to Affected.

5. Create an analytic sphere.

6. Select the sphere. Apply the Link tool.

7. Drag & drop the link object inside the particle level object.



The object hierarchy

8. Create a Fan object pointing at the sphere (The Fan tool is in the Simulation tab of the toolbar). The object hierarchy is shown in the adjacent image.

9. Click the Play button on the Animation window to see how the particles are emitted from the sphere and blown away by the fan.



Chapter 3. Subdivision Surfaces

Subdivision Surface Modeling

Subdivision surfaces combine all the good features of NURBS and polygonal surfaces; they allow organic shapes to be modeled easily and define perfectly smooth shapes with a varying surface.

Realsoft 3D provides a very powerful set of tools for modeling with subdivision surfaces.

This tutorial goes through a number of basic modeling examples for beginners.

Creating a coffee cup

The SDS toolbar allows you to create basic subdivision surface objects.



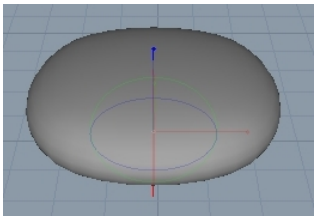
SDS tool bar

Click the SDS/Cube tool button. As usual, the control bar now shows you a number of subdivision cube specific options.



The options for the subdivision surface cube tool

Now switch to the top view either by clicking the top view button in the view control bar at the right side of the view window or from the view's popup menu Camera/Top. Define the cube through the view window by entering two points with the mouse. While doing this, you may hold down the Shift key in order to achieve a symmetric cube. A subdivision cube is created.



A subdivision surface defined by a cube-like control polygon

As you can see, the created cube defines an object which doesn't look like a cube at all. In fact, the cube tool created an object, which more or less looks like a sphere!

However, it is relatively easy to turn this spherical object to a coffee cup, or whatever other shape, as we will see soon.

This is actually a very common situation when modeling with subdivision surfaces. One starts from a very simple object, such as a cube, and adds details to it using different subdivision modeling tools.

The new subdivision object should be automatically selected after its creation. If you accidentally deselected it, click its name in the select window. As usual, the tool control bar automatically shows you the appropriate subdivision modeling tools as soon as a subdivision object becomes selected.



The control bar shows you the tools, which can be applied to the selected subdivision object

As you can see, most of the tools are disabled. The reason for this is that most of the subdivision modeling tools can only be applied in the 'Edit' mode. Subdivision objects provide three kinds of editing handles: faces, edges and points.



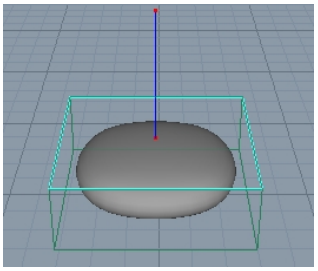
Select Face edit mode

Point handles allow you to single point edit the subdivision object. There are also a number of tools, which can be applied to points. There are also edge and face handles available - they modify several points at a time.

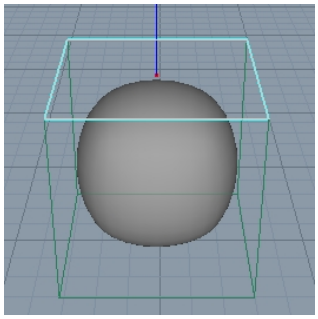
In order to turn our cube into a cup, we will be using the face handle mode. Select Face from the Control Bar's Edit field.

Select the top face by clicking it with the left mouse button. The selected face is highlighted.

Rotate the view window (drag downwards with the right mouse button Alt key held down) so that you can see the object's front and top sides. You can see a so called *face normal handle* sticking out from the selected top face. The face normal handle is perpendicular to the selected face and allows you to move the selected face.



Top face selected



Top face moved along its normal

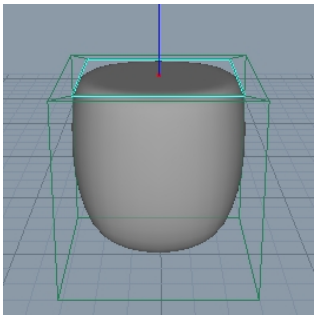
Let's move the face to make our 'sphere' a bit higher. To do this, simply move the mouse over the center of the face normal handle and drag it. Note: don't drag the end points of the handle. They scale and rotate the face.

When you selected the top face, many of the subdivision modeling tools in the control bar were enabled. This indicates that they can be applied to the selected face.

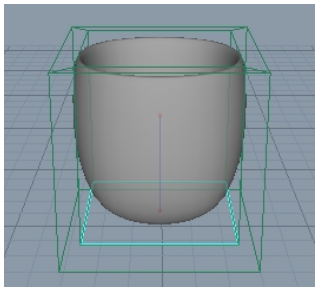
Activate the Subdiv tool (Subdivide Face) and LMB click in the view window.

Subdivide Face tool

When you now move the mouse up/down, you can see the effect of the subdivision tool. When an appropriate subdivision degree is achieved, click the left mouse button again to accept the operation.



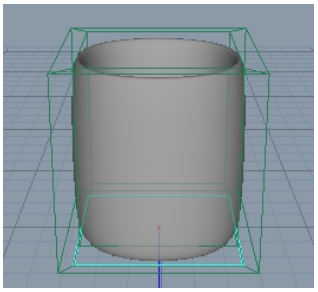
Top face subdivided



Top face extruded inwards

Now activate the Extrude tool and enter one point through the view window. By moving the mouse up/down you can extrude the top face up/down. If you extrude it downwards, the object starts to look like a cup.

The bottom of the cup is too round. Subdividing the bottom face adds new geometry, which will help to flatten the base. So, alt-drag with the right mouse button held down to rotate the view so that you see the cup from below. Click the bottom face to select it. Instead of pressing the Subdiv tool button, let's use the object handles. Hold the Ctrl key down and LMB drag the outer end knob of the face handle line. A new face appears and changes its size while you move the mouse. Make the new face almost as large as the original bottom face and release the mouse.



Bottom face subdivided

As you noticed, many useful subdivision tools can be activated by dragging the handles while holding a suitable modifier key down. The extrude tool which was applied above can be activated by Ctrl-dragging the actual face handle line, not its end knob. The handles usually provide the fastest way to achieve a certain action, because the tool is available at the same place where your focus is.

Now our cup is able to stand stable which is one of the basic requirements for a decent cup.

As you can see, with just a dozen of control points, we can define a cup with a smooth surface.

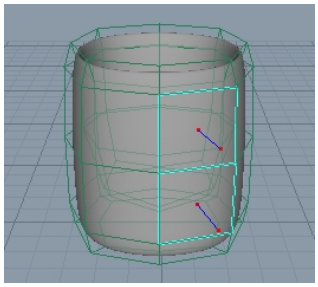
Extruding the handle

Most coffee cups have a handle. Let's see how we can use the Realsoft 3D modeling tools to create one.

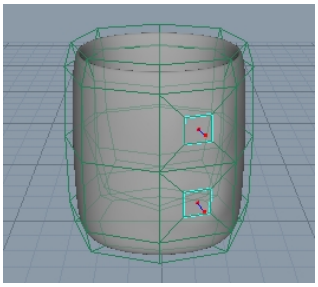
There are numerous ways to create handles for the cup. Which one is the best method, depends on the case. For example, we can use the Extrude tool to extrude two "tentacles" out and connect the ends of the tentacles. Let's try this.

Click the Smooth tool. This subdivides the entire control polygon of the cup, producing a more dense control polygon. However, the actual shape of the cup is not changed by this operation.

Now, select two of the side faces (one on top of the other). Keep the Shift modifier pressed when selecting multiple faces.



Select two side faces and..



..apply the Subdivide Face tool

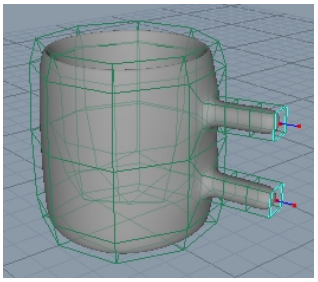
Now apply the Subdiv tool. Subdivide the selected faces just like we did when we subdivided the top face in the beginning of this example.

Activate the Extrude tool. When the tool is activated, the control bar shows you the available extrude tool specific options. Enter 3 into the Sections field in the control bar.

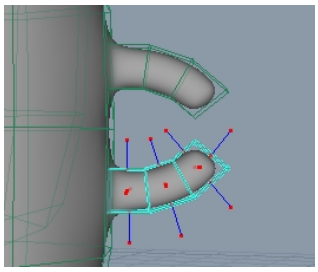


Extrude three cross sections

Now enter two points through the View window to get two tentacles extruded.



Two faces extruded with 3 sections



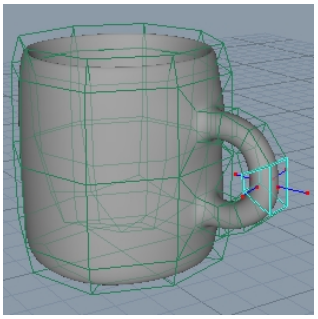
Extruded tentacles bent

Use the Realsoft 3D Bend tool (in the Transformation toolbar tab) to give the desired shape to the handle tentacles. To do this, rotate the view until you see the tentacles directly from the side. In the view window, LMB-drag a selection box, which includes the faces of the upper tentacle. The faces inside the box become selected. In edit mode, transformations modify only selected handles, and we can now bend only the upper tentacle.

Click the bend tool to activate it. Click once in the middle of the first cross section of the tentacle and click another time at the end of the tentacle. Move the mouse to find a suitable bend and click to finish the tool. Drag-select the bottom tentacle area and bend it, too.

Now we only need to connect the ends of the tentacles to finish the cup. To do this, zoom and rotate the view window to see the end face of the upper tentacle from below. Click to select it. Rotate the view upwards to see the end of the lower tentacle well. Press the Shift key (=multi select modifier) down and click the end face of the lower tentacle. When both faces are selected, the Tunnel tool in the toolbar becomes enabled. Click it to connect the selected faces. The cup is now ready.


The Tunnel tool


Tentacles connected

Using Compass Menus in SDS Modeling

The context sensitive control bar allows you to easily access any subdivision object specific tool. Just select a subdivision object and the control bar shows you the tools, which can be applied to the selected subdivision objects.

However, clicking buttons in the toolbar is not the most productive way to access tools in Realsoft 3D.

The compass menus provide a much faster interface to most modeling tools. Compass menus are context sensitive just like the control bar. In other words, the structure of the compass menu depends on the selected objects. For example, when you select an edge, the compass menu shows you the edge specific tools.

Compass menus are also completely user customizable. You can define any number of compass menus and bind them to any key or key combination. This makes compass menus a very powerful interface to access frequently needed tools.

This chapter demonstrates how the compass menus can be used to increase the productivity in subdivision modeling.

Another coffee cup

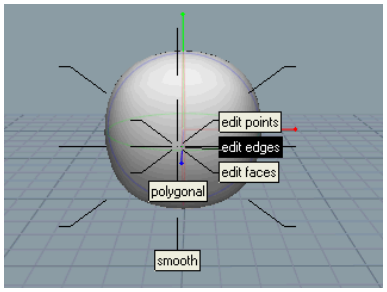
Let's create another coffee cup. This time we will use a slightly different approach.

Tutorial level: Beginner

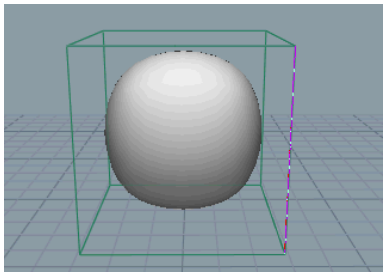
Example project: 'tutorprojects/modeling/sds/cupcreatedin20s'

1. Create a subdivision cube, select it and select the near east compass menu.

This switches the SDS object to edge editing mode.

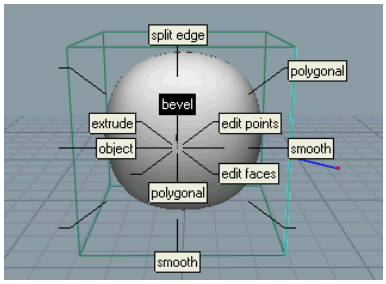


Switch to the edge editing mode by selecting the 'near east' menu

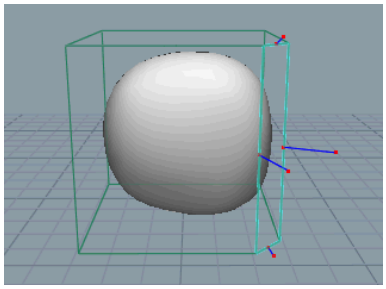

Mouse moved over an edge handle

2. Move the mouse over one of the vertical edges. The edge in question is highlighted and the cursor is changed to indicate that the mouse really hits the edge in question. Click the edge to select it.

Now select the near north menu. This activates the 'Bevel' tool and starts beveling.



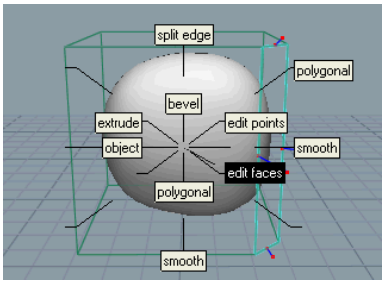
Select an edge and start the Bevel tool



The edge beveled

You only have to move the mouse up/down to control the beveling size. Bevel the edge so that the size of the beveling corresponds to the thickness of the handle to be created. This creates a new face into the cube.

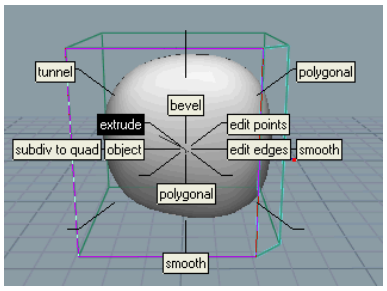
3. Switch to the face mode.



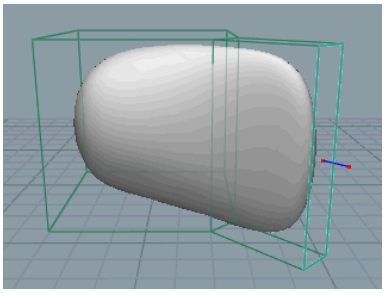
The near southeast compass menu switches to face editing

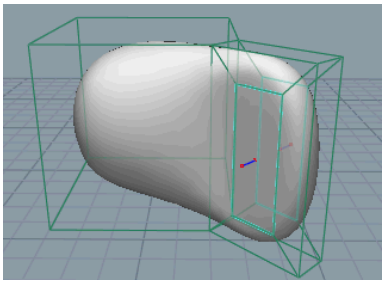
Again, you can use the Edit field in the tool control bar. The same can be accomplished by selecting the near southeast compass menu.

4. Move the mouse over the face so that the face gets highlighted and the cursor is changed. Click to select the face. Then select the near northwest compass menu. Move the mouse up/down to control the level of extrusion.



Select the near northwest compass menu to activate the extrude tool





Extrude a tentacle and subdivide the side faces of the tentacle

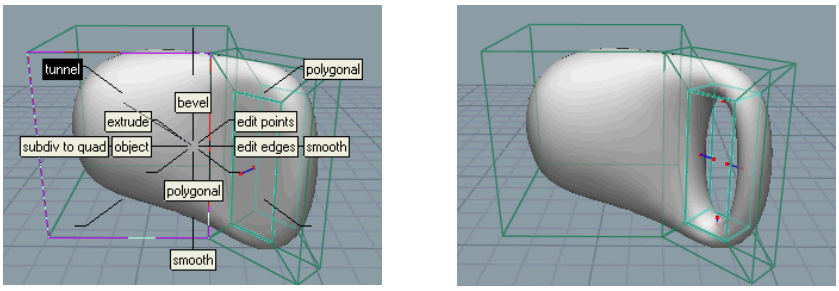
Next we have to drill a hole through the extruded handle tentacle.

5. Select both side faces of the extruded handle tentacle (select the first side face, rotate the view to the opposite side, hold down the Shift key and click the other side face).

Now, subdivide the selected faces by activating the 'Subdivide' tool from the tool control bar and by entering two points through the view window.

As a result of subdivision, the newly created inner faces become selected. The only thing you have to do is to apply the 'Tunnel' tool to the selected faces.

To do this, select the far northwest compass menu.

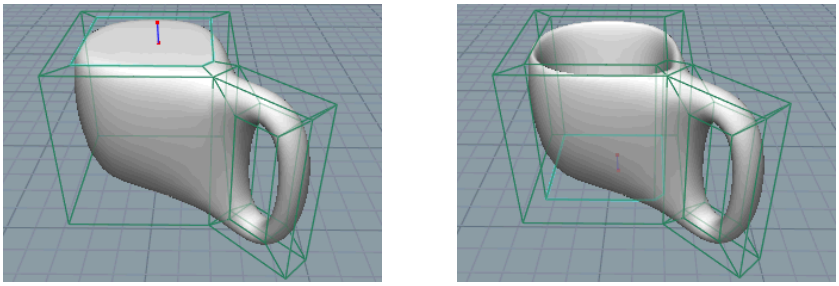


Create a hole through the handle using the Tunnel tool

6. Rotate the view downwards to see the bottom face of the cup. Select it and apply the 'Subdivide' tool to get a flat and stable base for the cup.

7. Rotate the view upwards, select the top face and apply 'Subdivide' to achieve a smaller inner face.

8. Activate 'Extrude' using the compass menu and extrude the inner top face inside the cup to make the cup hollow.



Subdivide the top and bottom faces and extrude the inner top face inside the cup

If you practice this tutorial a couple of times, you will be able to model the cup in less than 20 seconds!

Using Point, Edge and Face Handles

Object handles allow you to apply most of the commonly needed operations simply by dragging the appropriate handle. For example, dragging a point handle allows you to move the selected point.

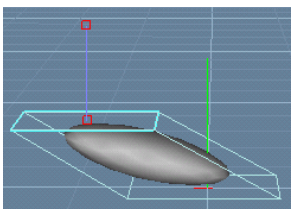
This tutorial demonstrates how the SDS object specific handles can be used to improve productivity in subdivision modeling.

Face handles

Faces can be selected and moved by simply clicking or dragging the face itself.

All the default handle selection rules also apply to face handles. For example, you can multi-select faces by holding down the Shift key while clicking them. You can use the drag box to select multiple faces in one operation.

The comma (",") key is the depth modifier. By holding down the comma key, you can select faces behind rather than in front. In conjunction with the Shift key, one can select faces in the front AND behind - allowing you to easily select back faces without having to rotate the view. For example, if you want to select the 'ear' faces from the side of a head, simply hold down the comma and the Shift keys while clicking the left mouse button. The first click selects the nearest 'ear' face and the second click selects the face on the other side of the head.



An object sheared by moving a face

Face edge handles

The edges of a face can be used for activating the following face tools:

- drag: move on edge (move face in the direction of the edge)

- drag + Shift: scale face about the edge center

- drag + Alt: rotate face about the edge.

Face normal

Face normal handles are only shown for the selected faces. The face normal handle is always perpendicular to the face.

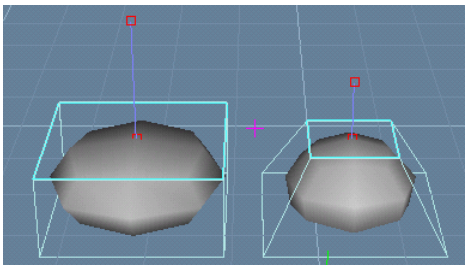
A face normal handle consists of three separate sub handles:

- end point - scales, subdivides
- rotate handle (face center)
- move handle (connecting line)

As you may already know, the face handle allows you to move the face along its normal. We lifted the top face of our coffee cup in the 'create a cup' tutorial with this handle.

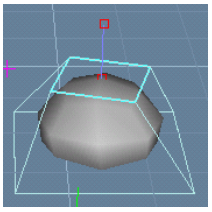
The length of the surface normal handle is relative to the size of the face. If you scale the face smaller, also the length of the surface normal handle gets smaller. Correspondingly, by changing the length of the surface normal handle, the size of the face can be changed.

To do this, simply drag the end point of the face normal handle.



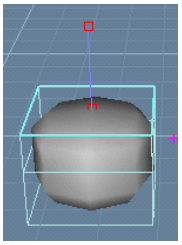
Drag the end point of the face normal handle to scale the face

The other end of the surface normal handle i.e. the *face center handle* allows you to rotate the face.

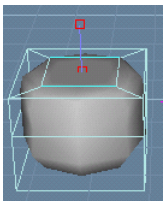


In most operating systems, you can duplicate files, icons, etc. by holding down the Ctrl key while dragging the icon. In Realsoft 3D, you can do the same with geometric objects; holding down the Ctrl key while moving an object duplicates the object.

In addition to entire objects, this works for faces too. Holding down the Ctrl key while dragging a face normal handle duplicates the face and connects the edges of the copied face to the edges of the original face. In other words, it extrudes the face.



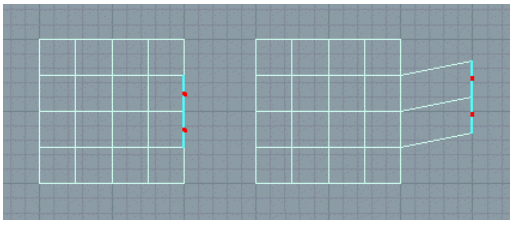
By dragging the end point of a face normal handle (scale handle) with the Ctrl key, you can subdivide the face.



Edge handles

Edges work similar to faces in that they allow easy selection and moving. Simply click or drag the desired edge to select or move it.

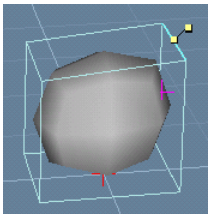
Dragging an edge handle while holding down the Ctlr key copies the selected edges and connects the copied edges to the original edges by creating new faces. The new edges are automatically selected and can be readily moved with the mouse. In short, this can be used to 'grow' new faces to selected edges.



A subdivision object before and after the 'copy selected edges' interactor.

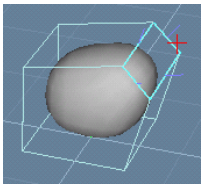
Edge normal

When an edge is selected, its edge normal handle is shown. This handle works similar to a face normal handle: it allows you to apply move on normal, rotate and scale operations to the edge.



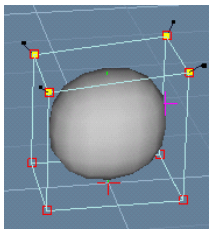
An edge selected and its edge normal handle shown

Edges cannot be subdivided or extruded. Therefore, Ctrl +normal handle is the same as the normal handle alone, and Ctrl+scale handle is used for beveling.



An edge beveled by dragging the end point of the edge normal while holding down the Ctrl key

Point handles



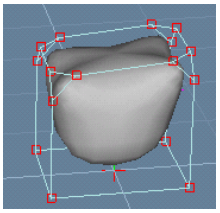
Four point handles selected

Point handles support most of the features described above. Dragging the actual point allows you to move it freely in 3D space.

When a point handle is selected, it shows its point normal handle.

The point normal handle works similar to the face and edge normal handles. It allows you to move the point along its normal.

You can't rotate or scale point handles because a point does not have a size. Therefore, the end point of a point normal handle always applies the bevel vertex tool.



Selected points beveled by dragging the end point of the point normal handle

Pointwise Creation of Arbitrary SDS Shapes

The basic creation tools (rectangle, cube, sphere, circle) allow you to create SDS objects with most faces having four points. However, Realsoft 3D does not set any restrictions on the number of points per face. A face can include any number of points.

This tutorial demonstrates how to use the 'Subdivision' modeling tool to create subdivision objects point by point (or face by face).

The 'Subdivision' creation tool can be found from the toolbar.



Subdivision creation tool

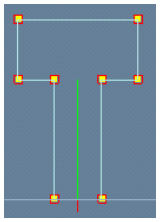
1. Activate the tool. The control bar now shows you the tool specific options.



Make sure that the Auto Accept checkbox is reset. This allows you to add any number of points to a face. Set Type to Smoothen to NURBS.

2. Now, enter eight points, which form the letter 'T, then click Accept.

3. Select the created object, switch to the face editing mode and select the face.



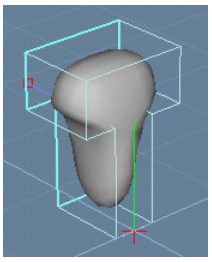
A face consisting of eight points

4. Activate the Extrude tool. The Control Bar shows you its options.



Set the Leave Floor option

5. Check the Leave Floor option and extrude the face. Our 'T' object is ready.



An object consisting of faces with four and eight points

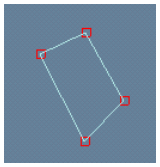
Creating multiple faces

You can also use existing points with the Subdivision creation tool. This can be achieved by holding down the Alt key while entering the points. Then the entered points become snapped to the nearest existing point (new points will not be added to the object).

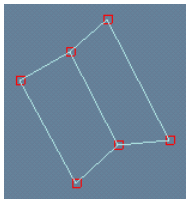
Let's create a head, which consists of multiple faces.

1. Activate the Subdivision creation tool and set the 'Auto Accept' option. Make sure that the 'Points per Face' field is set to '4'.

2. Create the first face by entering four points.



First face defined, four new points created



Second face created, two new points inserted

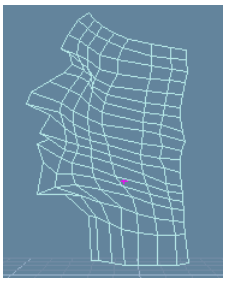
Because we activated the Auto Accept mode, the face is automatically created and you can start to define the next face. Do not click 'Accept'!

3. Now, hold down the Alt key and 'snap' the two first points to the existing points. Then release the Alt key and enter the remaining two points.

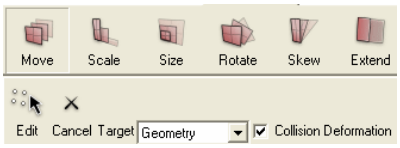
4. Continue adding new faces to the mesh by repeating step 3 so that the mesh finally represents a face from the side. When done, accept the tool. We now have a planar mesh model of one half of a face.

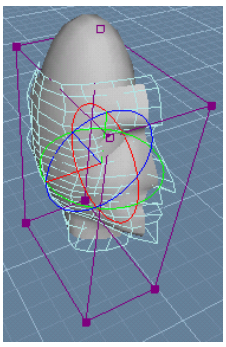
5. Reset the editing mode and create an analytic sphere behind the face. We will use collision detection to deform the face to make it a real 3D surface.

6. Activate the Transformation/Move tool and check the Collision Deformation option. Then move the subdivision mesh towards the sphere to get the face deformed.

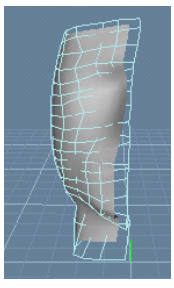


A planar subdivision mesh representing a face





Select the mesh, activate the move tool with collision deformation and move and deform the mesh against the sphere



Left side of the face

7. Switch to the face editing mode again and take the front view.



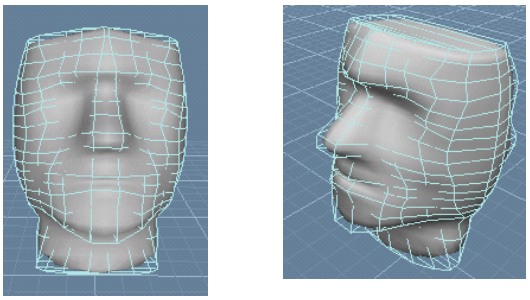
Duplicate tool

8. Click the Duplicate tool in the control bar. This duplicates all the faces in the object, creates new faces, which connect the edges between the original faces and the duplicated faces, and leaves the newly created faces selected.

9. Activate the Transformation/Mirror tool and mirror the duplicated faces to the right side.

The head is now ready.

Note: this is not the best way to model a head. The purpose was to demonstrate the Subdivision creation tool.



A complete head

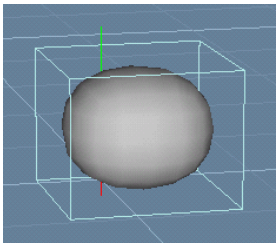
Modeling a Bird

This is a very basic subdivision modeling example which mostly uses two tools: subdivide and extrude.

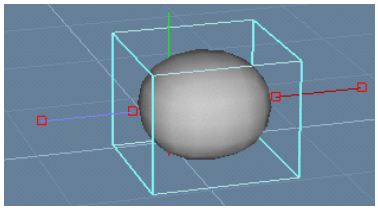
Recall that these tools can be activated as follows:

- *Subdivide*: Hold down the Ctrl key while dragging the face normal end point.
- *Extrude*: Hold down the Ctrl key while dragging the actual face normal line.

You can also activate these tools using the keyboard short cuts, compass or the context sensitive toolbar.



The body of a bird



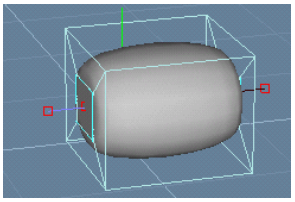
Side faces selected

1. Create a SDS cube and switch to the 'Edit/Face' mode.

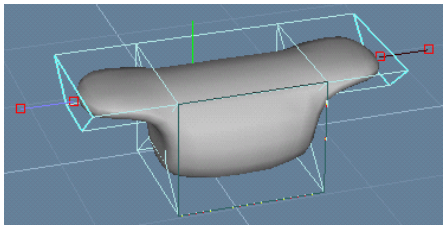
2. Select the left and right side faces. Now you can see their face normal handles.

3. Subdivide the selected faces: hold down the Ctrl key and drag the left normal end handle towards the face center.

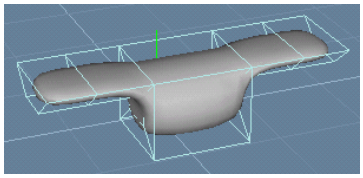
4. Select the topmost left and right faces and drag the 'move on normal' handle while holding down the Ctrl key to extrude. Do this twice.



Faces subdivided



Head and tail extruded

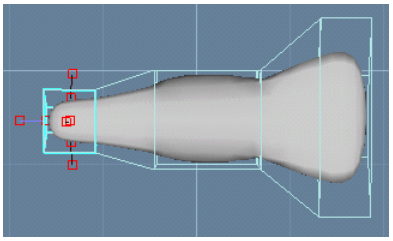


Head and tail extruded again

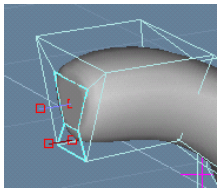
5. Now, switch to the top view and multi-select the right end faces. Stretch them wider (use the Transformation/Extend tool) to form a tail. Then use the Extend tool again to make the opposite end, the head, narrower.

6. Subdivide the end face of the head so that we can extrude a beak out of it.

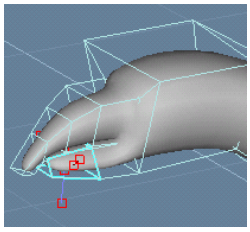
7. Extrude two end faces a couple of times and scale them to look like a beak.



The bird from the top view

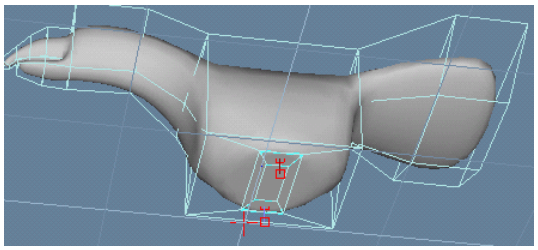


Head end face subdivided and two faces selected for extrusion



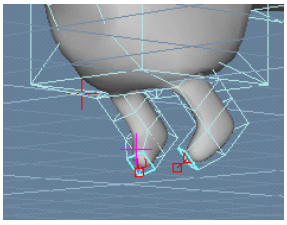
The head finished

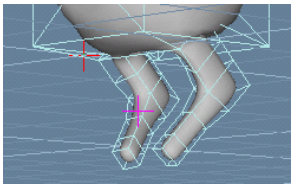
8. Subdivide the bottom face of the bird. Subdivide the created middle bottom face again. Scale a suitable pair of the bottom faces smaller so that they can be used for extruding legs out of the body.



The bottom face subdivided twice and two faces selected for the leg extrusion

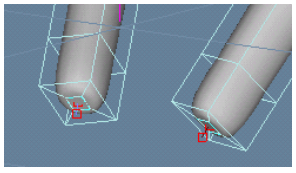
9. Now extrude the selected faces a couple of times to get the legs extruded.

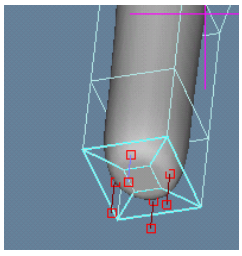




Legs extruded

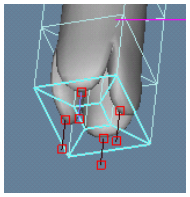
10. Subdivide the end faces of the legs. Then select the four outermost end faces in both legs for toe extrusion.



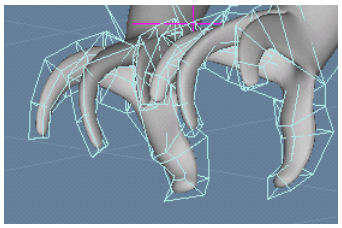


Subdivide the end faces of the legs. Four outermost faces selected.

11. This time start the 'Extrude' tool from the control bar. Set the Region option to Separately and extrude the toes.



Toes extruded



Toes finished

12. Modify the extruded toe faces making three toes to point forward and one toe to point backwards.

The next task is to extrude wings out of the side faces of the body to get our bird completed. However, this would just be repeating the steps described above. Therefore, we complete this example here.

Converting Objects to Subdivision Form

The Convert to SDS tool allows you to convert other geometric objects, such as NURBS surfaces and quadrics, to SDS objects.

Curves are converted to sets of SDS vertices. No faces will be created.

Surfaces are converted to SDS surfaces consisting of triangles (faces with three vertices) and quadrangles (faces with four vertices).

The standard Realsoft 3D installation contains a scanned NURBS head. You can convert it to the SDS form as follows:

1. Go to the select window's geometry tab and select 'Paste from file'. Load the file 'models/head'.

2. Select the loaded head. Activate the To SDS tool from the 'SDS' tab of the toolbar. By default, the tool examines the complexity of the selected object and attempts to generate an appropriate number of faces for the SDS object. This automatic resolution detection is usually desired so just select 'Accept' from the View's pop-up menu to finish the conversion. A SDS head is created.

3. Delete the original NURBS head.



Note

You can also check the 'Del orig.' option in the control bar, which deletes the original object after converting it to SDS. Then step 3 is not needed.

Inverting SDS surfaces

Conversion to SDS form works by evaluating surface points of the selected object to form the control polygon of the SDS object. Therefore, the generated SDS surface is typically slightly smaller than the original surface. A SDS surface usually shrinks inwards from its control polygon. The shrinking can be compensated by appying the 'Invert' tool. The actual surface will then pass through the previously used control points.

Converting polygonal models to the SDS form

The following three tools can be used to achieve a successful conversion from polygonal models to SDS surfaces:

Convert to SDS

The To SDS tool handles polygonal (typically triangle based) models by simply converting the polygons to corresponding SDS faces. Polygonal models often consist of triangles. Unfortunately a non-symmetric triangular structure, when used as a SDS control polygon, usually gives a very poor rendering quality. The triangular nature of the model can be seen from the rendering.

Merge Triangles to Quadrangles

The 'Merge Triangles to Quadrangles' tool scans through the selected triangles (or the whole object if it is not in the editing mode) and attempts to merge them to quadrangles. Whenever two associated triangles form an approximately rectangular face, the tool merges the triangles to one rectangle. This simplifies the model. Also, quadrangles usually produce much nicer SDS surfaces.

Note that often it is not possible to merge all the triangles.

The tool can be found from the main menu bar: 'Methods/Sub Division Surface' submenu includes a full SDS specific toolkit.

Invert

Finally, you can invert the generated surface so that the surface passes through the original polygon points.

Create a coffee cup

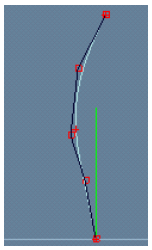
Any geometric object that defines a true surface can be converted to a subdivision surface. You can use, for example, all NURBS modeling tools to create subdivision objects easily. This is demonstrated in the following tutorial.

Let's first create two simple NURBS surfaces, which represent the body and the handle of a coffee cup.

1. Create a vertical curve representing the rotation axis of a coffee cup. To do this, select the NURBS/Curve tool and click the left mouse button in the view window. Finish the curve by selecting Accept from the view's pop-up menu.



Nurbs Curve tool



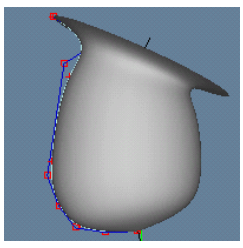
Points of the curve

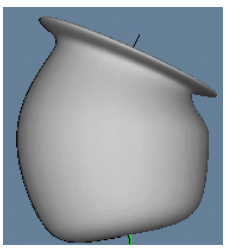
2. Select the created axis curve and activate the NURBS/Curve tool again. When the curve tool is active, the control bar shows you a number of curve specific creation options. Set the Pen option to Rotate.



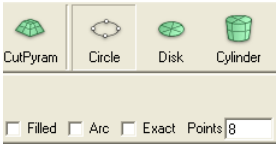
Activate the Curve tool and set Pen to Rotate

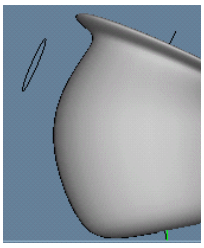
3. Enter a number of points through the view window to define the profile curve for the cup. You can see the rotated surface in real time. Accept when the profile is ready.





Define a profile curve for the cup

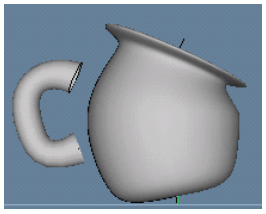




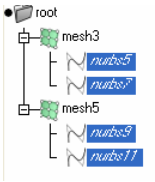
4. Switch to the side view and create a circle representing the cross-section of the handle. For example, select Nurbs Circle tool from the NURBS tools menu. Then enter two points through the view window to define the center and the radius for the circle.

Create a NURBS circle representing the cross-section of a handle

5. Select the created circle, activate the NURBS/Curve tool again and set the Pen option to Sweep. Enter a number of points through the view window to create a handle for the cup. Then click Accept.



Two NURBS surfaces

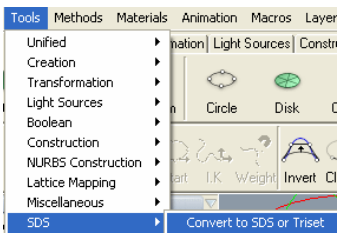


Delete the construction curves

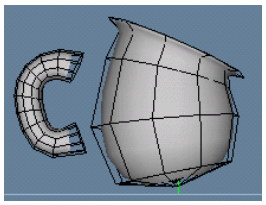
Now the NURBS modeling specific part of this tutorial is done and we can convert the NURBS objects to subdivision objects.

Before we do this, delete the construction curves, which were used to create the surfaces. These curves are located inside the NURBS meshes and allow you to edit the shape of the meshes. However, we only want to convert the surfaces to subdivision objects. So, select the curves and delete them.

6. Select both meshes and select the pull-down menu Tools/SDS/Convert to SDS or Triset. This activates the conversion tool. The control bar allows you to define the number of faces to be created. Default options are OK, so just click the Accept button. Two subdivision objects are created. Delete the original NURBS surfaces.



NURBS meshes converted to SDS objects





Note

You can also check the Del orig option in the control bar, which deletes the original objects after converting them to SDS.

We now have to weld the handle to the body of the cup. To do this, we need to merge the two separate SDS objects into a single object.

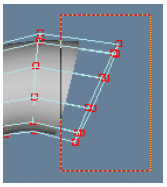
Merge O tool

7. Select both SDS objects and click the Merge O (merge objects) button in the control bar. This merges the two selected objects into a single object.

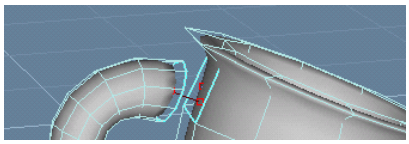
We could now blend the handle to the cup by creating faces manually between the edges of the handle and the body of the cup. However, this is quite a time consuming job. We use the 'Tunnel' tool instead. This tool allows you create a surface through any two faces. The only problem is that there is no face in the ends of the handle surface. This problem is easy to fix:

8. Switch to the point editing mode and select all the points of the upper end of the handle. Then click the Add button to create a closing cap face. Repeat the operation also for the lower end of the handle.


Add face tool


Select the end points of the handle and click the Add tool

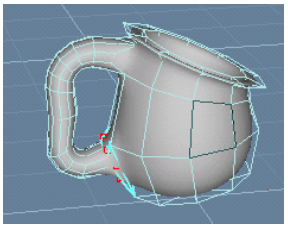
9. At this point, make sure that the handle is placed so that the end faces of the handle match well two faces of the body. If not, select all points of the handle, move them from the front view and rotate them from the top view. To deselect points, press the Alt key and click on, or drag around, the points you wish to deselect. Pressing Ctrl + t deselects all selected points.


Select the faces to be welded...

10. Now switch to the face mode and select the upper end face of the handle. Hold down the Shift key and select a suitable face from the body of the cup. Then click the 'Tunnel' tool to create a surface through the selected faces.


...and click the Tunnel tool

Repeat this for the second end of the handle.


The handle merged to the cup

Defining Pointwise Properties

You can define surface properties of SDS objects as you would define those of any geometric objects: using materials and constant attributes listed in the 'Col' tab of the property window. In addition to this, you can also set surface properties for individual control points of SDS objects.

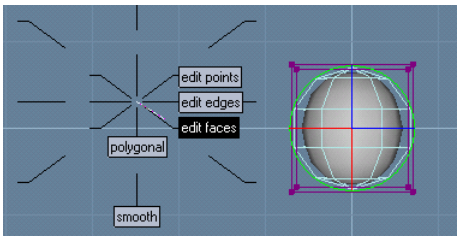
You can define surface properties through selected points, edges and faces. This tutorial demonstrates how to manage pointwise properties.

Pointwise colors

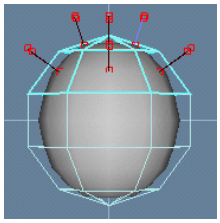
Let's create a sphere whose north pole is red and south pole is blue.

1. Take a Top view and then create a SDS sphere.

2. Take a front view and switch to point, edge or face editing mode (for example, select the 'Edit Faces' from the compass menu).



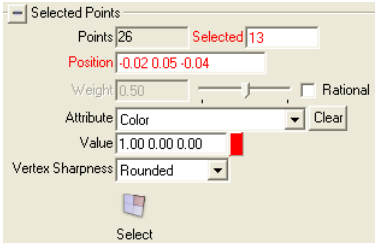
Enter Edit Faces mode



Northern faces selected

3. Select the faces of the upper half of the sphere.

4. Open the property window and go to the 'Spec' tab. Make sure the 'Attribute' field in the 'Selected Points' group is set to 'Color'. Then enter the value '1 0 0' (red) to the 'Value' field.

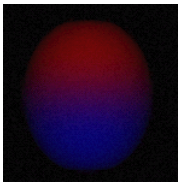


Color property for the northern faces set to red

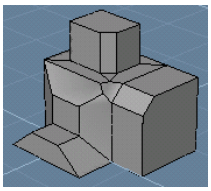
5. Select the southern faces of the sphere and define a blue color through the 'Value' field.

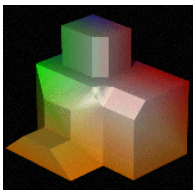
6. Render the subdivision sphere to see the result.

You can define all surface properties, such as transparency and refraction factor, this way.



A smooth SDS sphere with pointwise colors

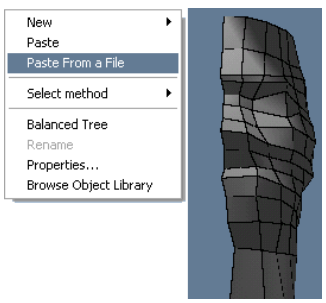




A polygonal SDS object with pointwise colors

Creating Symmetric Objects with the Duplicate Tool

Many real world objects are symmetric in some way. For example, left and right sides of a human head are quite similar. The SDS 'Duplicate' tool is provided for utilizing this observation in SDS modeling. You only have to model 50% of the object and use a transformation tool to get the other part modeled.

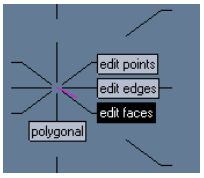


Load 'tutorprojects/sds/halfhead'

The Duplicate tool takes care of three things. It first duplicates all points, edges and faces in the selected object. Then it finds all open edges in the original and duplicated faces and connects them with an appropriate number of new faces. Finally, the tool selects the duplicated faces so that you can immediately move, mirror, scale or otherwise transform them.

1. Go to the Select Window's geometry tab and click the right mouse button. A pop-up menu is opened. Select Paste from a file option and load the 'tutorprojects/modeling/sds/halfhead' object.

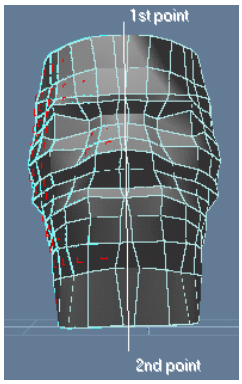
2. Select the loaded face and enter 'Face' mode by selecting the near southeast compass menu.



Enter face editing mode

3. Click the Duplicate tool.

Duplicate tool



Duplicated faces mirrored

Note: Duplicate tool automatically selects the duplicated part of the object. Be careful not to change the selection, because re-selecting the exactly overlapping faces is very difficult. If you accidentally change the selection, undo the 'Duplicate' operation and execute it again.

4. Activate the Mirror tool; it's located in the Transformation tab.

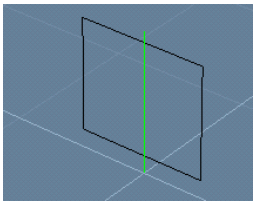
Mirror tool

Then enter two points through the view window to define vertical mirroring axis.

The result is a complete face. The tool connects also the back, top and bottom sides of the face. If this is undesirable, select the connecting faces using the top view and apply the 'Destroy F' tool to them.

A cube

A cube is an object, which clearly has an axis of symmetry: its left and right side faces are identical. So, we only have to create one face and get the other face by using the Duplicate tool. Side faces are created automatically by the tool.


Enter two points through the view window to create a SDS rectangle

Activate the SDS Rectangle tool...

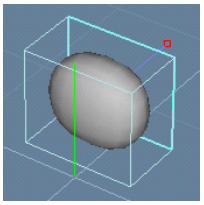
1. Create a SDS rectangle consisting of 2x2 points.

2. Select the rectangle and enter the face editing mode. Now the control bar shows you all the SDS specific tools. Click the Duplicate tool.

This duplicates the single face of the rectangle object and creates four new faces to connect the edges of the original and the new face.

Note: the tool automatically selects the duplicated face.

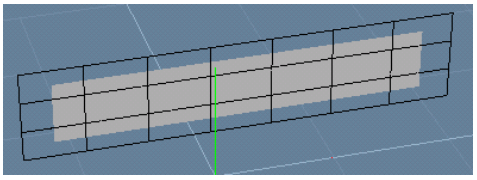
3. Activate the Transformation/Move tool and enter two points through the view window to displace the duplicated face from its original position.


The duplicated face moved

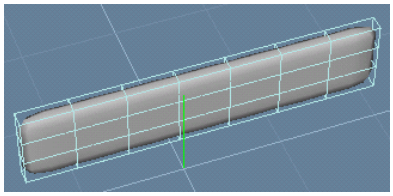
Extruded mesh

In this example, we create an object used by body builders to train their biceps.

1. Activate the SDS/Rectangle tool. This time set the resolution to 8x4 through the control bar (U=8, V=4).

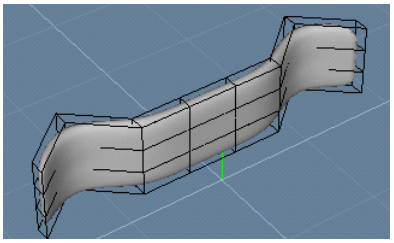


SDS mesh consisting of 8x4 control points



Extruded mesh

2. Enter Edit/Face mode and click the Duplicate tool. Use the Move tool (or one of the normal handles) to displace the duplicated faces from the original faces.



Bent mesh

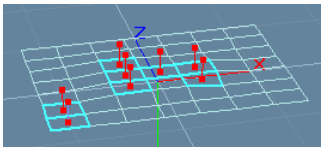
3. Switch to the point editing mode. Select both cross-sections next to the ends of the object and move them backward to bend the object as shown below.

A mesh with holes

The Duplicate tool finds all open edges in the surface to be extruded, including 'holes':

1. Create a SDS rectangle consisting of 8 x 8 control points.

2. Enter the face editing mode and delete a couple of faces from the mesh using the 'Destroy F' tool.

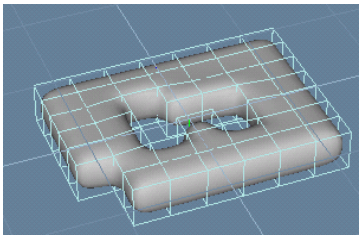


Select a couple of faces and destroy them with the Destroy F tool



Destroy Face tool

3. Click the Duplicate tool and use the Move tool to move the duplicated faces apart from the original faces.

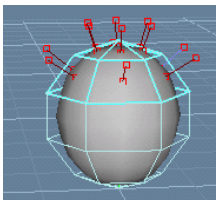


A mesh with holes extruded with the Duplicate tool

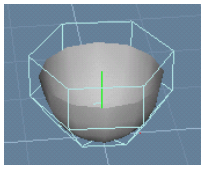
A coffee cup

In this example, we create a coffee cup by modeling the outer surface for the cup first. The inner surface for the cup is created with the 'Duplicate' tool (the inner surface is otherwise identical to the outer surface but slightly smaller).

1. Take a top view and create a SDS sphere. take a front view and select the faces near the north pole as shown in the image.



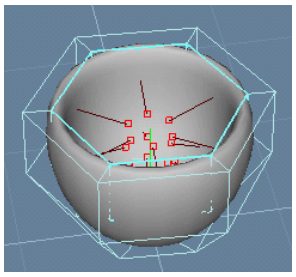
Select the topmost faces



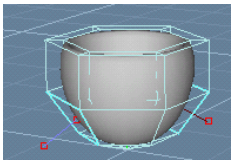
Upper faces destroyed

2. Apply the 'Destroy F' tool to the selected faces.

3. Click the 'Duplicate' tool. This creates the inner surface for the cup. Drag one of the scale handles at the end of normal handles to scale the inner faces smaller.



Outer faces duplicated and scaled inwards to form the inner surface for the cup

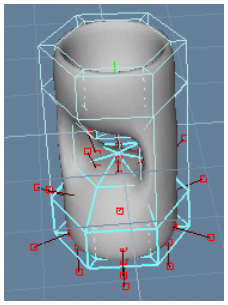


Select two opposite faces

We could create a handle for the cup as demonstrated in some of the earlier tutorials. However, let's be more creative this time. The cup has a round base, and therefore it is not a practical cup to use. We can duplicate and mirror the cup in vertical direction to create a proper base for it.

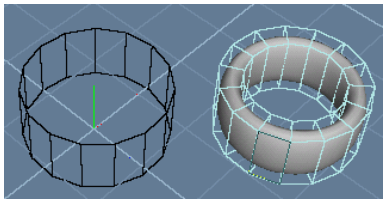
4. Select two opposite faces near the bottom of the cup.

5. Click the 'Destroy F' tool to delete the selected faces. This creates two rectangular holes in the outer surface of the cup. Click 'Duplicate' and then use the 'Mirror' transformation tool to flip the duplicated faces under the cup as shown in the image on the right.

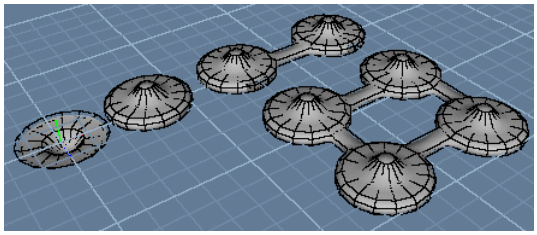


A cup base created by mirroring the duplicated faces

Other examples



Duplicate + Scale

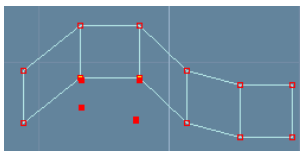


Duplicate + Mirror

Adding New Points and Faces

You can add new points to a SDS object by using the standard Ctrl + drag method. In other words, hold down the Ctrl key while dragging a vertex with the left mouse button.

This duplicates and moves all the selected points.



Two points duplicated by dragging them while holding down the Ctrl key

Creating faces through existing points

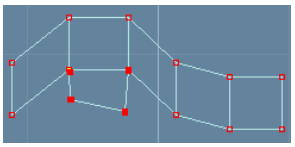
To create new faces through existing points:

1. Switch to the Point editing mode.

2. Select three or more points in the order in which you want them to be inserted to the new face.


Add face tool

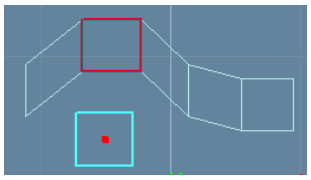
3. Click the Add tool.


A new face added to the selected vertices

Copying faces

You can copy faces similar to points - hold down the Ctrl key while dragging a face.

This duplicates and moves the selected vertices or faces.


A new face created by dragging an existing face with the Ctrl key

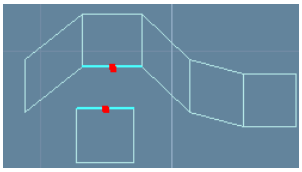
Adding new faces through edges

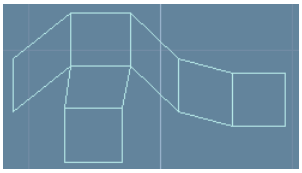
You can create new faces also through existing edges by using the Add tool. The tool creates a new face through the selected edges. For example, to connect two edges belonging to separate faces:

1. Switch to the Edge mode.

2. Select the desired edges.

3. Click the Add tool.


Two edges selected


Add Face applied

Creating new faces on the fly

The Add face tool allows you to add new faces through existing edges or points. However, sometimes there are no suitable points or edges from which the new faces could be created. You can solve this problem by first creating the points (by dragging existing points with the Ctrl key as described earlier) and then using the Add tool to create a face through the points.

The Faces tool speeds up this process by allowing you to enter new points and faces simultaneously. This tool also allows you to snap to existing points.

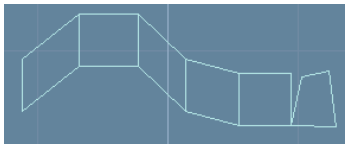

Faces tool

For example, if you have a 5x2 SDS mesh and you would like to make it 6x2 mesh by inserting a new face to the tail of the mesh:

1. Select the mesh to be modified.

2. Click the Faces tool of the control bar.

3. Click the left mouse button twice to create two new points for the SDS mesh. Then press down the Alt key and move the mouse near the existing edge points to attach the remaining two points. By holding down the Alt key, you can snap to existing points and no new points will be inserted to the selected SDS object.

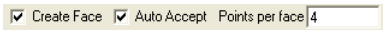

A new face under construction

4. When all the four points are inserted, select Accept to terminate the face creation.



Note

If you want to create several faces which all will have equal point count, you can use the Auto Accept mode. For example, to create faces with four points, set Points per face to 4. A new face is created every 4th entered point.



A new face is created automatically after every 4th entered point



Note

If you only want to insert new points to the selected object (not faces), reset the Create Face option.

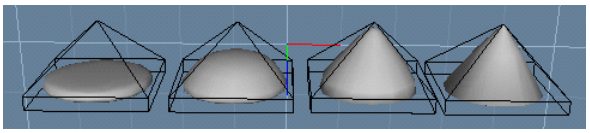
Rational Subdivision Surfaces

Rational subdivision surfaces introduce a new parameter called 'weight'. The higher the weight, the stronger the control point affects the surface.

You can turn any SDS object into its rational version by dragging points, edges or faces with the left mouse button while holding down the Alt key. By moving the mouse up, you can increase the weight. By moving the mouse down, you can decrease the weight.

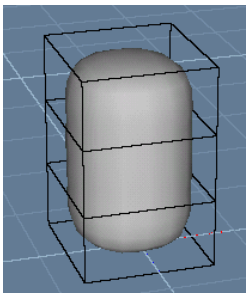
The following image demonstrates the effect of changing the weight of the apex of a pyramid.

In the next tutorial we create a simple 'jug' object which demonstrates the power of rational SDS surfaces.

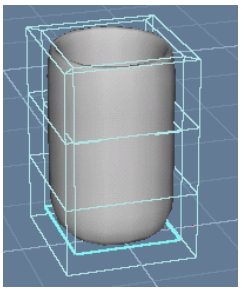


Weights from left to right: 0.1, 1, 10, 100

1. Switch to the top view and create a SDS cube of the resolution 2x2x4 (click the SDS/Cube tool and enter U=2, V=2,W=4 through the control bar). Switch to the face mode and subdivide and extrude the top face as shown below. We have created so many cups by now that we don't need to repeat the detailed steps here anymore.



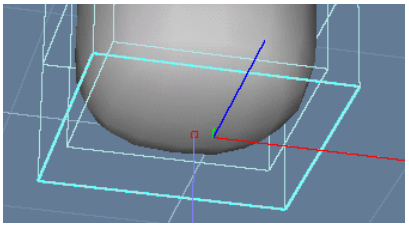
A SDS cube with 2x2x4 control points

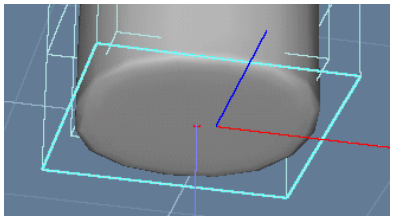


The top face subdivided and extruded inside the cube

The first problem is that our jug has a pretty round base and it wouldn't be able to stand on a table. We could subdivide or bevel the bottom face to get it more 'flat'. However, increasing the weight of the bottom face has the same effect.

2. Select the bottom face.

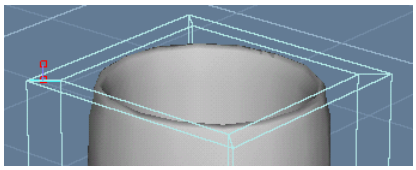




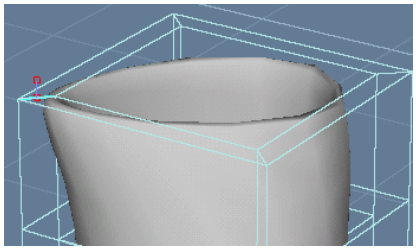
Bottom face weight increased

3. Press down the Alt key and drag the bottom face with the left mouse button. Drag upwards a couple of times until the desired sharpness is achieved.

Let's create a spout for the jug next.



Select one of the top face edges



Weight of an edge increased

4. Switch to the edge mode and select one of the diagonal 'top face' edges.

5. Zoom in so that you can clearly see the edge and drag the edge with the left mouse button while holding down the Alt key. Instead of moving the edge, the weight of the edge is modified. Drag up to increase the weight until the desired sharpness for the spout is achieved.

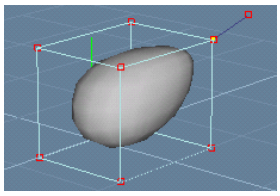
Defining weights through the property window

You can also make the surface rational by setting the Rational checkbox in the Property Window/Spec tab.

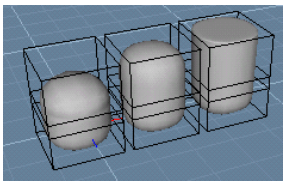
You can then select any of the control points, edges or faces and define the desired weight through the Weight field.



Set the Rational check box in the property window



Weight of the selected point increased

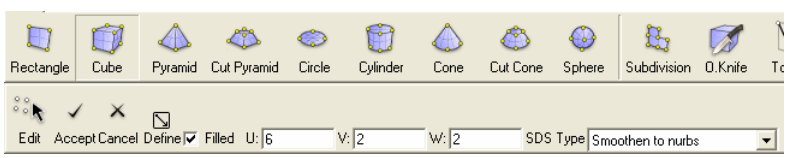


Top face with different weights

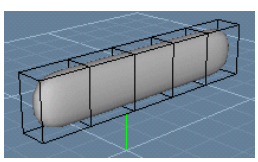
Create a propeller

In this tutorial we demonstrate weights by creating a very simple propeller.

1. Create a SDS cube consisting of 6x2x2 control points.

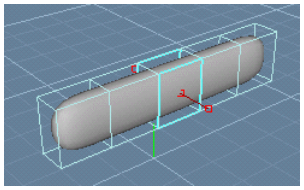


Options for creating a 6x2x2 SDS cube



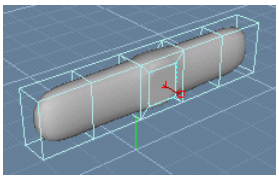
6x2x2 SDS cube

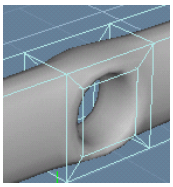
2. Enter the face editing mode and select two faces as shown in the figure:



Two opposite faces selected for the 'axis' hole

3. Subdivide the selected faces (for example, hold down the Ctrl key while dragging an end point handle of a face normal). Then click the Tunnel tool to drill a hole for the screw shaft.

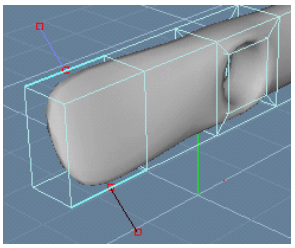




Subdivide and Tunnel tools applied

4. Switch to the edge mode and select two opposite edges at both ends of the screw. Then open the property window, set 'Rational' check box and enter the value 5 to the 'Weight' field.

5. Repeat the step 4 at the other end of the propeller, but this time use the other edge pair to twist the shape to the opposite direction.



Weights of two opposite edges increased

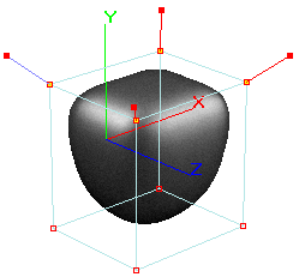
Creases

Creases allow you to create sharp edges for SDS surfaces.

Creases are similar to 'weights' in that they allow you to control how strongly a certain point, edge or face 'attracts' the actual surface.

Realsoft 3D supports three crease levels: sharp, semi sharp and rounded. In addition to this, there are two kinds of creases: point creases and edge creases.

Point creases are affected whenever you use the 'Sharpness' tool in the point editing mode. If you are in edge or face editing mode, edge creases are affected. This means that the result of setting creases for all points of a face is different from setting crease values for the edges of the face.

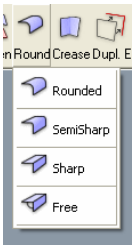


The four top points have a semi sharp crease value. The bottom vertices have the default smooth crease value.

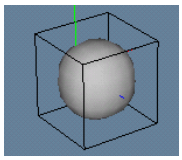
The following tutorials demonstrate how different crease values and types affect the shape of the actual surface.

Point creases

You can define crease values for selected points using the 'Sharpness' SDS tool.



Sharpness tool in the control bar

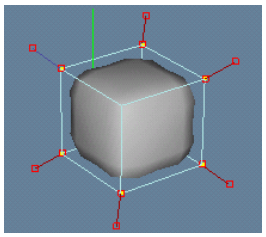


A SDS cube with default crease values

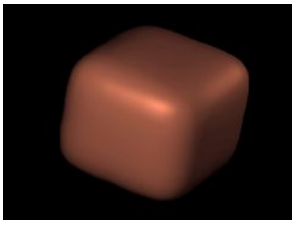
1. Create a SDS Cube. By default all crease values are set to 'rounded' and the resulting surface looks like a sphere.

2. Enter the point editing mode and select all points of the SDS object. The 'Sharpness' control in the tool control bar becomes enabled.

3. Set sharpness to Semi Sharp.



The SDS cube with semi sharp point creases



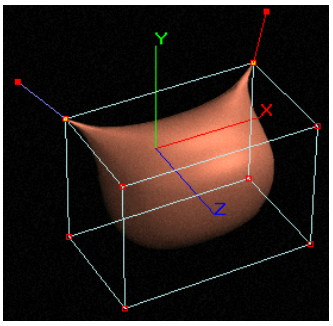
A ray traced SDS cube with semi sharp point creases



Note

OpenGL shading can represent the true shape of a SDS surface only with a limited quality. In photo realistic rendering, creases become rendered with a proper quality.

If you set a point crease to sharp, you'll get a sharp peak, as shown in the image on the right.



Two points with a sharp crease

Edge creases

Edge creases are defined in edge or face editing modes. Defining a crease value for a face corresponds to defining a crease value for all edges of the face.

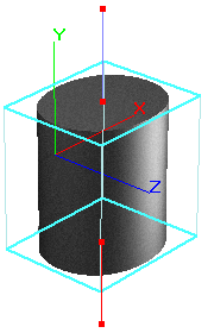
Let's see how edge creases differ from point creases.

1. Create a SDS cube just like in the previous example.

2. This time, enter the face mode and select the top and bottom faces.

3. Set crease to 'Sharp' using the sharpness tool.

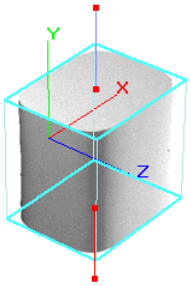
The actual surface looks like a cylinder now.



A SDS cube with 'Sharp' top and bottom face creases

Mixing point and edge creases

Point and edge creases can be mixed.



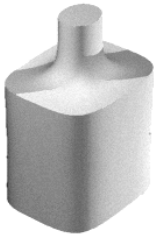
The edge creases of the top and bottom faces are 'sharp'. All point creases are set to 'semi sharp'. Side edges are also semi sharp.



Note

If you apply tools which create new points or edges for SDS object, all inserted points will have a default 'smooth' crease value.

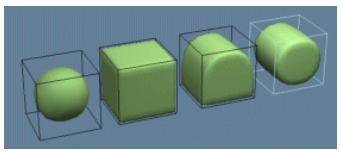
For example, if you subdivide and extrude the top face of the SDS object in the image above, the following object is the result:



Top face subdivided and extruded

Rounding

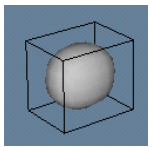
The rounding feature allows you to round edges with a desired rounding radius. Rounding is different from creases in that it can only be applied to edges. The biggest advantage is that the size of the rounding is freely adjustable, whereas creases have only three different values (sharp - semi-sharp - rounded).



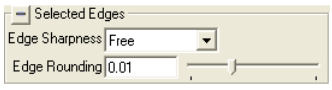
Four identical SDS cubes with different edge rounding

Rounding is an edge specific attribute, which can be set in the edge editing and face editing modes. Rounding a face corresponds to rounding all the edges of the face.

1. Create a SDS cube. By default, no rounding values are defined for the edges and the surface looks like a sphere.



SDS cube with no edges rounded



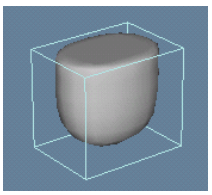
2. Switch to the face editing mode and select the top face. Open the property window and set the 'Edge Sharpness' to 'Free'. This allows us to specify exactly the desired rounding radius. Change the 'Edge Rounding' field in the 'Spec' tab to 0.001 meters. In other words, define 1 mm rounding for all the edges of the top face.



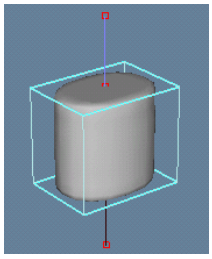
Note

You may have to set the number of decimals to a value of 3 in the Preferences Window - Metrics tab to be able to enter the value of 0.001.

The cube now defines the following surface:



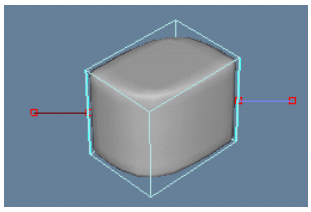
Top face with a 0.001 m rounding



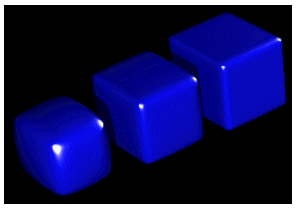
Edges of the top and bottom faces rounded

3. Now, select the bottom face and set the rounding of it to 0.001 as well. You now have a cylinder whose both ends are slightly rounded.

4. Switch to the edge editing mode, select two opposite vertical edges and define a 0.001 rounding for them as well.



0.001 rounding defined for two opposite vertical edges



Identical SDS control polygon cubes with different rounding values

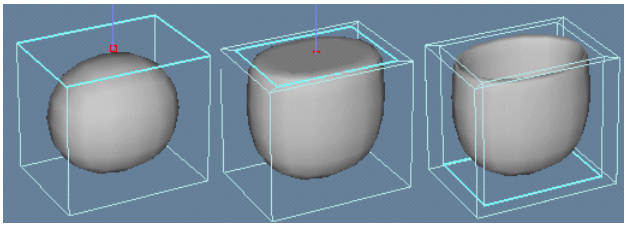
It is already easy to guess what happens if you define rounding for all edges: the result is a cube with rounded edges.

A box with rounded edges

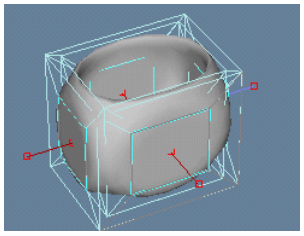
Let's create a box with rounded edges.

1. Create a SDS cube.

2. Switch to the face editing mode and select the top face. Subdivide and extrude the top face inwards as shown in the figure below.



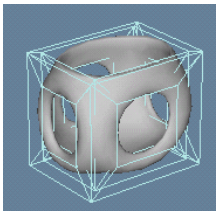
Select, subdivide and extrude the top face



Vertical faces subdivided

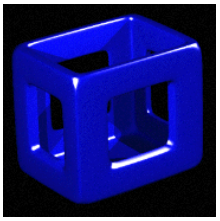
3. Now, select all the vertical faces (including the inner side faces) and subdivide them. Make sure you set the 'Region' to 'Seperately' in the Subdivide Face tool control bar.

4. Select a face pair in the middle of a cube 'wall' and apply the tunnel tool. This drills a hole through the wall. Repeat this for all four sides of the box.



Walls punched with the tunnel tool

5. Use the drag box to select all faces and set a 0.001 meter rounding radius through the property window. The box is now ready.



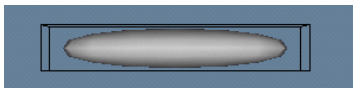
Edges with roundings

Model a sink

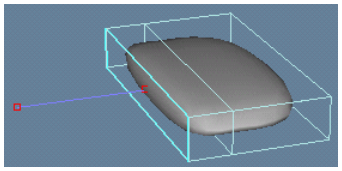
Let's create a beautiful sink. Similar to those you can find in any modern bathroom.

Example project: 'tutorproject/modeling/sds/sink'

1. Switch to the front view and create a SDS cube representing the sink from the front.



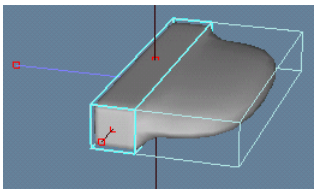
A SDS cube in the front view



One of the faces extruded

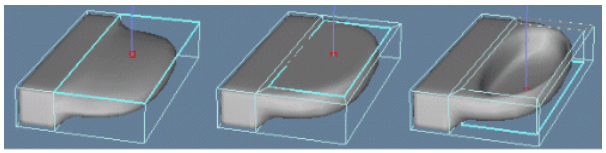
2. Enter the face editing mode and extrude the 'back face' of the SDS cube.

3. Use dragging to select all the faces in the extruded part and define a small rounding (e.g. 0.002 meters) through the property window.



0.002 rounding defined for the selected faces

4. Select, subdivide and extrude downwards the remaining 'top' face of the sink as demonstrated by the attached image.



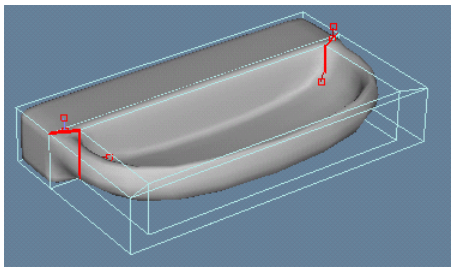
Rightmost top face selected, subdivided and extruded downwards

5. Switch to the edge editing mode and round the four edges shown in the image to value 0.002.

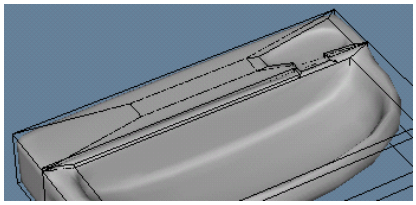


Note

It can be difficult to select edges in the shaded mode because the edges are partially hidden by the actual surface. You can switch to wire frame rendering mode to select the edges easily.



Edges marked with red color rounded



Top face subdivided. One of the new faces subdivided and extruded inwards to form a place for soap.

Our sink starts to look like a real sink already. However, all real sinks have some additional details like a place for soap. Use your imagination to further edit the sink. For example, to create a place for a soap by subdividing and extruding.

After some practicing, you will be able to create a real sink in just a couple of minutes.



The finished sink

Assigning Materials to Faces

You can attach textures and materials to SDS objects in several ways:

- Using geometric mapping objects, such as the parallel and the sphere mapping. This method is suitable for procedurally defined (3D volume) materials. The disadvantage is that rigid mapping coordinates cannot follow deforming SDS objects in animations.
- With a default mapping object, which uses a vertex channel to define the texture or shader coordinates. The number of vertex channels is not limited: in addition to the UV Coordinates channel, which is available in every SDS object, the user can add his/her own channels 'myuv1', 'myuv2' etc. The disadvantage of vertex channels in surface texturing is that it is often difficult or impossible to unfold a complex, branching surface into a UV plane continuously enough. Two points that are close in 3D space may get dislocated in UV space.
- With face materials or textures. These are attached using a private set of UV coordinates for each face, which approach solves the unfolding problems of vertex channels. On the other hand, interpolation of face material UVs is less continuous as vertex channel interpolation, because neighbor faces do not contribute.
- With UV set objects. UV sets are sub meshes of SDS objects defined in UV coordinate space. You can use any number of UV sets to split the parent mesh geometry into suitable UV areas (e.g into front and back projections). Each UV set can contain any number of disconnected face groups.

Assigning Materials to Faces Using Drag & Drop

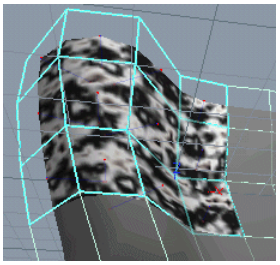
The easiest way to attach materials to faces is using drag & drop.

1. Put the SDS object into the face edit mode.

2. Select the target faces

3. Zoom and rotate the view so that target faces are relatively parallel to the view window and fill it. UV coordinates for face materials will be the projection of vertices into the view.

4. Drag & drop a suitable material into the view window. This assigns the material to the selected faces.



Granite material assigned to the selected faces

You can assign any number of materials to a single face. Materials are evaluated in the order in which they are assigned.

Editing Texture Coordinates in UV Editor

Each material gets an unique set of texture coordinates through which you can control the position and size of the texture. You can edit these texture coordinates in the UV Editor window.

The UV editor allows you to edit the UV coordinates of the currently selected face material (there can be any number of materials per face, but the *current face material* allows you to select, which one of them is shown in the UV editor).

When you open the UV editor for the first time, it automatically evaluates the current face material to a bit map image which is then shown as a backdrop image in the UV Editor.



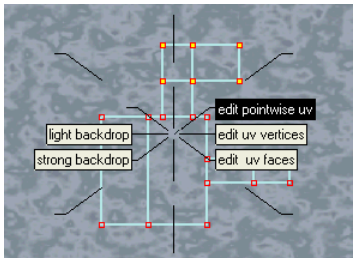
Note

Evaluating the backdrop image can take several seconds depending on the size of the UV editor window. However, this process is run as a separate thread so you can start using the UV editor as soon as it opens.

Through the UV editor you can edit the UV coordinates of the currently selected face material. Use any transformation or deformation tools, such as Scale, Rotate, Mirror, Bend etc. to modify the UV coordinates.

UV editor also supports compass menus. You can put frequently needed UV manipulation tools (mirror faces etc.) to compass menus. The default startup project defines an example compass menu for the SDS objects.

Pressing the Delete key on the UV Editor window removes the current face material from the selected faces (deletes the selected UV faces).

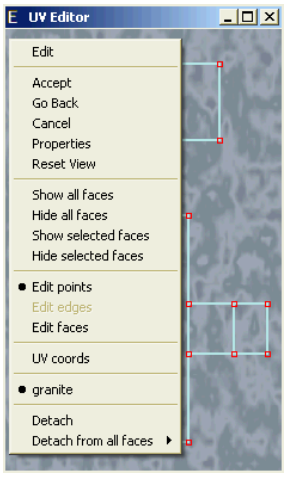


The UV editor and a SDS specific compass menu

Most of the standard modifiers also work in the UV Editor. For example, all selection modifiers (Shift = Add to selection, Ctrl = Intersection, Alt = Remove from selection) etc. work. When modifying faces, Shift activates angle grid and so on.

When modifying the UV coordinates in the UV Editor, 3D view windows update in real time.

There is also a context sensitive pop-up menu associated with the UV View.



Just like in 3D views, there are three editing modes available: Points, Edges and Faces. However, edge editing is not avalaible for face materials, only for vertex channels.

The following menu options control which faces are shown on the UV Editor (note that only faces that contain the selected face material can be shown):

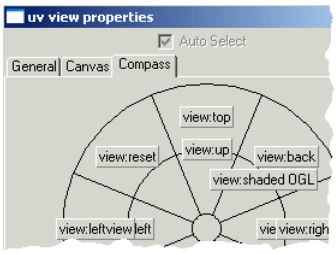
- "Show all faces" - Shows all the faces that contain the current face material
- "Hide all faces" - Hides all faces
- "Show selected faces" - Shows the selected faces (+ currently visible faces)
- "Hide selected faces" - Hides the selected faces

To show only the currently selected faces you can use the "Hide all faces" and "Show selected faces" menu items in succession.

The popup menu can also be used to select the current UV and the current face material name of the selected SDS object. The UV:s are listed first and the face material names last (there is a delimiter between the two groups). The SDS specific menu of the UV editor window allows you to detach materials from faces:

- Detach - detaches the current face material from the selected faces.
- Detach from all faces - detaches a face material from all faces. The face material is selected from the sub menu.

You can animate face uv coordinates by modifying them in animation recording mode.



UV editor's property window compass tab

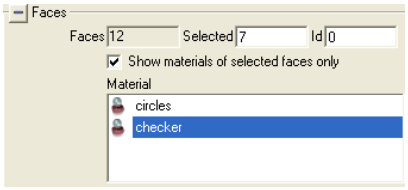
The Properties item in the UV View's pop-up menu allows you to open the property window for the UV View. Through this window you can control options like the contrast (weight) of the backdrop image, grid, rendering options and compass menus.

Note that you can record all options in this window to macros and bind those macros to desired keys, compass menus or icons for faster access.

Editing Face Materials through the Property Window

The object specific part (the Spec tab') of the property window shows you the materials assigned to the selected faces. Click the desired material to make it the *current face material*. Use drag & drop to change the order in which the materials are evaluated. The pop-up menu allows you to remove materials from faces.

The SDS property gadget has a checkbox 'Selected faces only' inside the Faces frame. If checked, the Material list works as before: shows the materials of the first selected face and its Detach popup menu removes the selected face material from the selected faces. If not checked, the Material list shows all face materials and Detach removes the selected face material from all faces.



In the Property window/Spec tab you can select the current face material, delete materials and change the order in which materials are applied

Map2Obj

This tool provides you with more control over assigning face specific materials.

Map2Obj tool (located in the Materials tab) allows you assign materials to faces and define face specific UV coordinates using any mapping object, such as the parallel map. For example, you can use a parallel mapping to attach an 'eye' texture to the faces representing an eye of a head.

1. Select the desired SDS faces
2. Create a parallel mapping object (from any material)
3. Multiselect the SDS object and the parallel mapping object and activate the Map2Obj tool
4. Set Target to Selected Faces or All Faces, select the material to be assigned using the Destination Material field and click Accept.

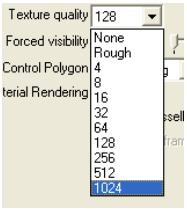


Assign marble material to selected faces using the Map2Obj tool

Previewing Face Materials in OpenGL

To preview a single face material in OpenGL, open the property window, go to the Wire tab and set the Material Rendering option to Single. Specify also a suitable texture quality. Material Rendering=Full evaluates all mapped materials and is usually slower.

After the texture quality is specified, the system converts the current face material to a bit map image (texture map). You can find the generated images from the 'temp' folder. The image is named as 'materialname_texture.bmp'. Next time you attach the material to a face, OpenGL will be able to show the result instantly.



Define texture quality through the Property window's Wire tab

Using the Knife tools

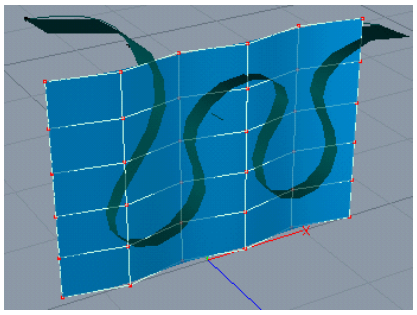
The SDSPlus plugin must be installed to be able to use these tools. Two knife tools are available: the Object Knife Tool and the Freehand Knife Tool.

Object Knife Tool

You can use the SDS Object Knife tool to create new points and edges to selected SDS objects. The surface of the last selected object is used as a knife to cut the selected SDS objects. New vertices are added to edges where the edges pass through the surface of the knife object.

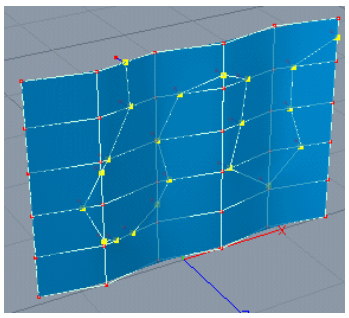
You can find this tool from the SDS tab of the default working environment. The tool has the following options:

- Selected - if checked only the selected edges are affected
- Add edges - new edges are added between the new vertices



A Nurbs surface (knife) passing through a rectangular mesh

- Select - which faces to select. The SDS object must be in face selection mode. The values are:
  - No selection - does not change selection
  - Inside - select faces whose center point is inside the knife object
  - Outside - select faces whose center point is outside the knife object



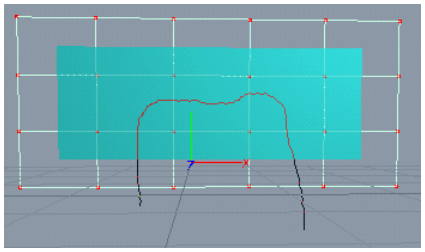
The knife tool applied

Freehand Knife tool

A curve or a line segment can be drawn defining a surface used as a knife. New vertices are added to edges where the edges hit the surface of the knife. This tool can be activated buy clicking the "Knife" button on the SDS control bar or by selecting the "Methods/Sub Division Surface/Knife" menu item.

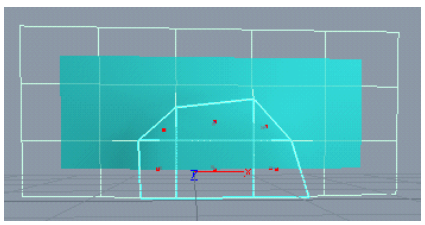
The tool has the following options:

- Selected - if checked only the selected edges are affected.
- Add edges - new edges are added between the new vertices.
- Freehand - if checked a freehand curve can be drawn. Otherwise a line segment can be drawn.

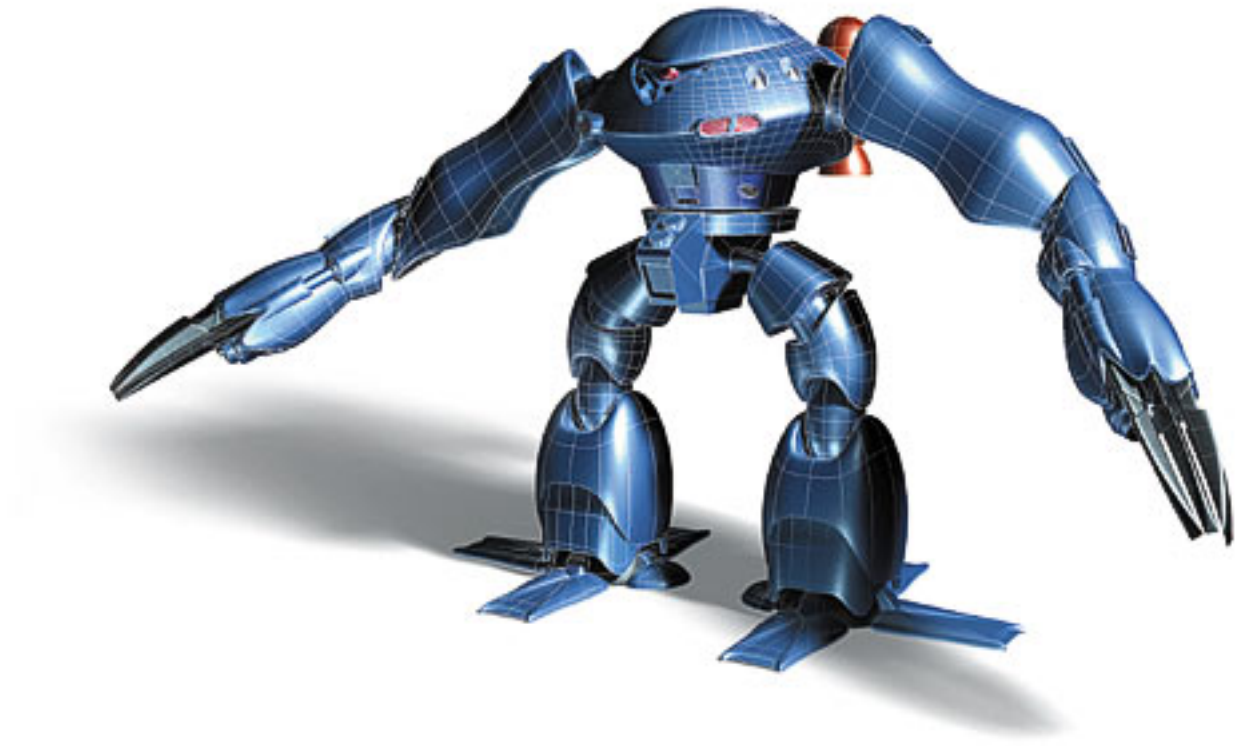


The Knife tool activated and freehand curve drawn over a SDS mesh

- Select - which faces to select. The SDS object must be in face selection mode. The values are:
  - No selection - does not change selection
  - Inside - select faces whose center point is inside the knife object
  - Outside - select faces whose center point is outside the knife object



Freehand knife applied



Chapter 4. NURBS Curves and Surfaces

NURBS Curves

NURBS modeling is often based on the idea that the user first creates one or more freeform curves and then applies appropriate surface construction tools to the curves.

In this tutorial we explore various NURBS curve manipulation tools.

To create a cubic NURBS curve:

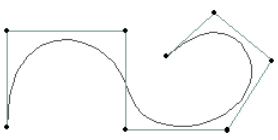
1. Click the NURBS/Curve tool.

As always, the control bar shows you the curve tool specific options. The default options are set for open cubic NURBS curves.



The curve tool activated. The control bar shows the curve specific options.

3. Click the left mouse button in the View window to enter a desired number of control points for the NURBS curve. If you click a new point in a wrong place, select Go Back from the view's popup menu to remove the last entered point. When done, select Accept from the tool control bar or from the view popup menu.



Create a cubic NURBS curve by entering a desired number of control points

Editing NURBS curves

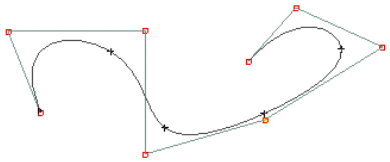
Similar to most other geometric objects, also NURBS curves specify object specific handles which can be accessed in the Edit mode.

For example, to change the position of the first control point:

1. Select the curve and enter the Edit mode (hit the Space bar, for example).

You can now see the control polygon of the curve.

2. Drag the first control point to move it.



Curve modified in the edit mode

You can also modify the curve using any transformation tool, such as the Move tool. First select some control points by using dragging or by clicking them while holding down the Shift key. Then use the move tool the usual way to move the selected points.

You can also modify the knot points (the actual curve points). Knot points are represented as small crosses on the curve. Just drag them to move them.

Interpolation curves

You can also create a NURBS curve by defining the actual curve points (the knot points). This is called 'interpolating' or 'inverse design'.

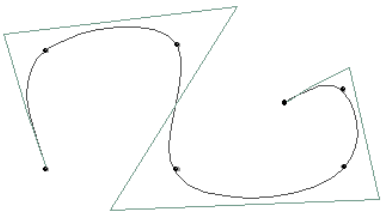
To create a curve which passes through the desired points:

1. Activate the curve tool and set the Interpolate check box.



Set the Interpolate option

2. Enter a number of points through the view window and select Accept.



A curve passing through the specified knot points

Creating freehand curves

The curve tool can be used for creating freehand curves. When using the Freehand option, you don't define curve by clicking the left mouse button. Instead, you drag with the mouse to define a freehand shape. The tool then creates a curve which represent the specified path with a minimal number of control points.

To create a freehand curve:

1. Activate the curve tool.

2. Set the Freehand option.



Freehand option set

3. Press down the left mouse button in the view window and move the mouse to define a desired freehand shape. When done, release the mouse and the curve is created.



Hold down the left mouse button while moving the mouse to create a freehand curve

Controlling the order

The Order option controls whether the created curve is polygonal, quadric or cubic.



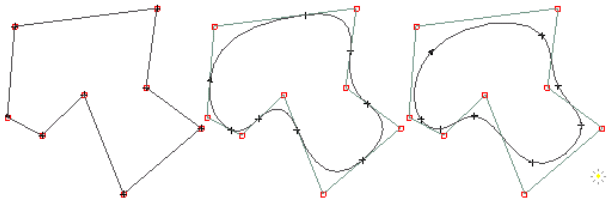
Select Polygonal order and check the Closed option

To create a closed polygonal curve:

1. Activate the curve tool.

2. Set Order to Polygonal and check the Closed option.

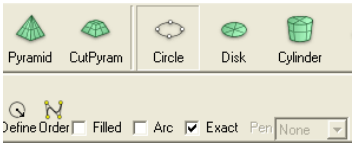
3. Click a desired number of points through the view window. Finish the curve by clicking the Accept button.



Polygonal, quadric and cubic NURBS curves

Creating circles and rectangles

The NURBS tab also has tools for creating circles and rectangles. These tools can be used exactly the same way as the Curve tool to create NURBS curves. In many cases you have to create circles or arcs. The circle tool can be used for this purpose.



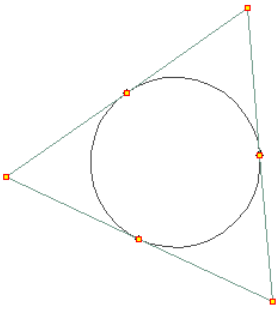
Circle tool selected with the Exact option set

For example, to create an exact NURBS circle:

1. Activate the NURBS/Circle tool.

2. Check the Exact check box. Make sure that the Filled option is not set.

3. Enter two points through the view window to define the center and the radius for the circle.



An exactly circular NURBS curve

You can also create non-exact circles. In this case, you can freely control the number of control points used in defining the circle. Such a circle is not mathematically exact but is often useful when creating various freeform objects.

To create a circle consisting of 8 control points:

1. Activate the NURBS/Circle tool.

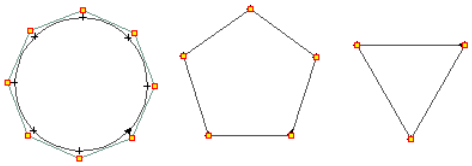
2. Define a desired number of control points to the Point field of the control bar.

3. Define the center and the radius in a view window.



Note

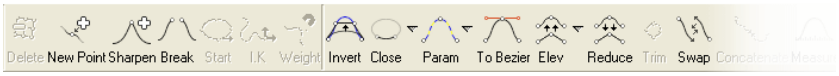
You can also create polygonal circles. For example, a triangle can be represented as a closed, polygonal NURBS curve consisting of three control points.



A cubic circle consisting of eight control points on the left. Two polygonal circles on the right.

Inserting points

When you select a NURBS curve, the tool control bar automatically shows the curve specific tools, which can be applied to the selected curves.



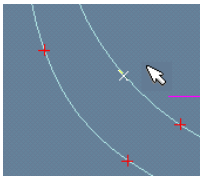
Control bar with the NURBS curve specific tools

To add a point to a curve:

1. Select a NURBS curve.

2. Click the New Point tool in the tool control bar.

3. Drag along the curve to find an appropriate position for the new point. The new point is indicated as a small cross. When you release the left mouse button, the new point is inserted to the curve.



Find a position for the point by dragging near the curve

Inserting new points to a curve does not change the shape of the curve. You can also create a new point by duplicating existing points. Holding down the 'Ctrl' key while dragging a knot point in the edit mode does this.

Creating sharp corners

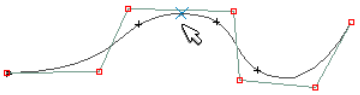
You can also add a new point to a curve so that the new point allows you to create a sharp corner to the curve. To add a sharp point to a curve:



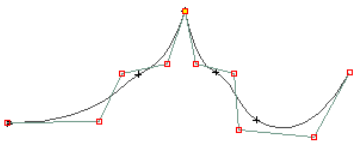
The Sharpen tool

1. Select the curve and activate the 'Sharpen' tool.

2. The Sharpen tool works exactly the same way as the Add Point tool: Move the mouse near the NURBS curve and drag with the left mouse button to find an appropriate position on the curve. When found, release the mouse button and the new point is inserted to the curve.



Drag along the curve to find a position for the new point



A Sharp point added and moved

The tool does not change the shape of the curve in any way. The nature of the inserted point is revealed when you modify the point.

3. Move the new point in the Edit mode. A sharp corner is formed.

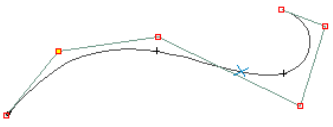
Breaking curves

The Break tool splits a curve into two parts.

The break point is defined the same way as the position of the new point in the 'Add Point' and 'Sharpen' tools:

1. Select the curve to be split and activate the 'Break' tool.

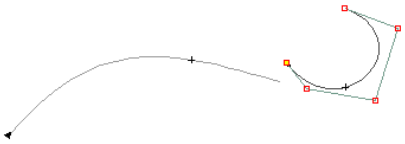
2. Drag along the curve to find a desired break point. Then release the mouse to accept the tool.



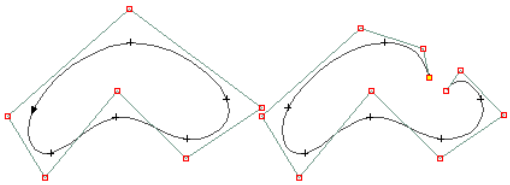
Drag along the curve to define the break point

The shapes of the resulting two curves represent together exactly the shape of the original curve.

You can apply this tool to a closed curve to open it from a desired point. Again, the shape of the curve remains unchanged.



A curve split into two parts. The first point of the tail part moved.



The original closed curve on the left. The Break tool applied and end points modified on the right.

Defining the start point of a closed curve

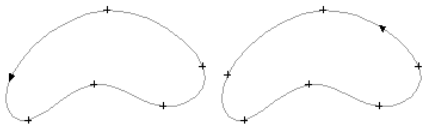
NURBS curves are 'parametric curves'. The beginning of a curve corresponds to the parameter value 0 and the end point of the curve corresponds to the parameter value of 1. However, a closed curve does not have an apparent beginning - any point could start the curve.

The 'Start' tool can be applied to translate the parameter space of a closed curve so that the given point becomes the start point of the curve. The start point is rendered as a small triangle. This can be relevant when using closed curves in path animations or as parameters for constructing surfaces.

The new 'start point' can be set to any position of the curve. The operation does not change the shape of the curve.

1. Select a closed curve and activate the 'Start' tool.

2. Drag the mouse along the curve. Release the left mouse button at a desired point.

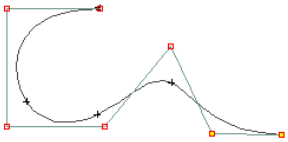


Two identical curves with different start points indicated by a small black triangle

Deleting control points

To delete one or more control points from a curve:

1. Select the curve and hit the space bar to enter the Edit mode. The curve now reveals its control polygon. Select the control points to be deleted.

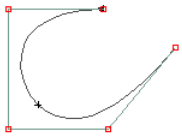


Two rightmost control points selected

3. Click the Delete button in the control bar.



Delete Control Points tool



The selected control points deleted

The shape of the remaining curve is changed by the operation. You can use the 'Break' tool to get rid of a part of the curve without affecting the shape.

Weights

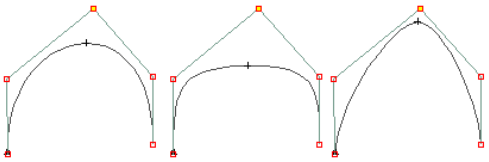
NURBS curves support a parameter called 'weight'.

The weight parameter allows you to control how strongly a control point affects the corresponding curve point (knot point). The higher the weight, the stronger the control point pulls the curve towards the control point.

You can modify the weight parameter interactively as follows:

1. In Edit mode, select the desired control point.

2. Activate the Weight tool from the control bar and click the mouse in the view window. By moving the mouse up/down you can increase/decrease the weight of the selected control points. Click the mouse again to accept the tool.



Three NURBS curves with different weights at the topmost control point



Note

You can also modify the weight of the selected control points by dragging them while holding down the Alt key.

Degree elevation and reduction

The NURBS Curve tool can create polygonal, quadric and cubic nurbs curves using the selected Order parameter.

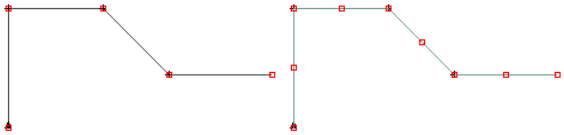
The order of a polygonal curve is two. Mathematically speaking, such a curve is of first degree. Second degree curves are called 'quadrics' and they can represent accurately circles and ellipse arcs, for example. Cubic curves are often a very good compromise between smoothness and performance. However, even higher order curves can be created. The higher the degree, the smoother the curve.

The Elevate and Reduce tools modify the degree of a curve.

Assume that you have a polygonal curve and you need to bend its tail smoothly. A polygonal curve does not allow smooth bending. Therefore:

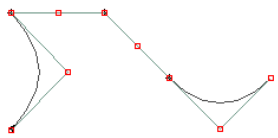
1. Create a polygonal curve.

2. Select the curve and click the Elevate button on the control bar.



The original polygonal curve on the left. The degree elevated quadric curve on the right.

As you can see, the degree elevation didn't change the shape of the curve. This is possible because quadric NURBS curves are flexible enough to represent also polygonal shapes. To see that the curve is really quadric, edit the last control points to bend the tail.



Two control points moved

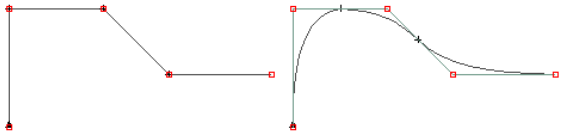


Degree elevate and smooth

You can also elevate the degree of a curve so that the increased order is used to smooth the curve. The Elevate tool includes a pop-up menu, which selects the elevation method. Click the small arrow associated with the Elevate button to open the menu.

High order NURBS curves are useful when you need extra smooth and continuous curves for motion paths. High level of smoothness is important in camera fly animations, for example.

To reduce the degree of a curve, just click the Reduce tool. Note that degree reduction always changes the shape of the curve.



The original polygonal curve on the left, Elevate by Smoothing applied on the right

Concatenating curves

As demonstrated above, NURBS curves can represent both polygonal and smooth shapes. How about creating a single curve, which contains both polygonal and smooth sections?

An easy way to achieve this is by using the Concatenate tool.



Concatenate multiple curves to one

1. Create a cubic curve.

2. Create a polygonal curve, which starts exactly from the end point of the cubic curve.



Note

Use dragging to snap the position the first point of the polygonal curve exactly over the last point of the cubic curve.

3. Select both curves (using the Shift key) and click the Concatenate tool to get single curve with cubic and polygonal curve segment.



Concatenating two curves of different order



Note

If the end points of the curves are displaced, the tool creates one additional curve segment to fix the displacement. If the end points overlap, the result matches exactly the shape of the original parameter curves.

You can select the curves for concatenation in any order. The tool automatically finds correct order by studying the distances between the end points of the selected curves.

Concatenate is a very powerful curve construction tool, because it can join curves with different parametrization and order to one single curve.

Parametrization

Parametrization of NURBS curve is a rather mathematical issue and there is no easy way to explain it. A great deal of the power behind NURBS curves relies on the possibility to use a suitable non-uniform parametrization. Fortunately, in most cases, you don't have to worry about the parametrization issues.

The Param tool changes curve parametrization. The following parametrization methods are supported by the tool:

- Uniform.
- Chord Length

- Centripetal

Typically, Chord Length or Centripetal parametrization give the smoothest curve shape (and most constant speed when used as an animation path). However, sometimes you may want to use also Uniform parametrization.



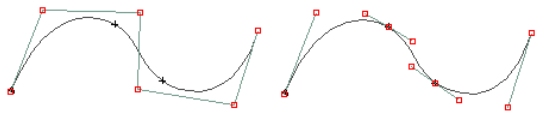
Note

Changing the parametrization always affects the shape of the curve. In fact, many of the tools we have demonstrated above modify the curve parametrization to achieve the desired effect. For example, the Break tool increases so-called 'knot multiplicity' so that a sharp corner can be represented. By applying the Param tool, you lose the custom parametrization needed for sharp corners. The result is a curve, which smoothly approximates its control points.

Bezier curves

Thanks to non-uniform parametrization, NURBS curves are powerful enough to represent also Bezier curves. However, this parametrization technique can only be used if the number of control points is appropriate. Therefore, it is not available as an option for the Param tool.

Instead, a special Bezier tool is included, allowing you to convert any NURBS curve to the Bezier form.



A standard cubic NURBS curve with a chord length parametrization on the left. The Bezier tool applied on the right.

The Bezier form provides you with a great deal of power over the shape of the curve. However, it also allows you to define curves, which are not as smooth as curves with uniform, chord length or centripetal parametrization. This can be a significant disadvantage when using curves to construct surfaces or when using them as motion paths in camera animations.

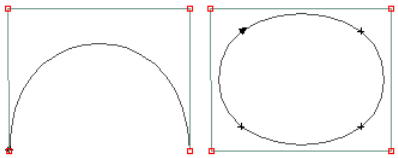
Opening/closing curves

You can close an open curve and open a closed curve by clicking the Close tool. If you have selected various curves and some of them are open, some closed, the pop-up menu associated with the Close button can be used to select whether all the curves should be opened or closed.

There are two ways to open curves: exact and approximate.

When using the exact method, the shape of the curve is not changed. The curve is broken from its start point and you can see that it is really open only by displacing its end points.

The second method simply takes the existing control polygon to define a new open curve. The shape of the curve is changed dramatically at the ends.



An open curve on the left. The result after applying Close on the right.



Note

If you have used Sharpen, Bezier or other tools which create a custom parametrization, the Close tool may introduce a sharp point where the curve is closed. You can fix the problem by changing the parametrization to uniform, centripetal or chord length.

Changing the direction of a curve

The Swap tool inverts the direction of a curve. The direction of a curve is significant when using it as a motion path or as a parameter for various surface construction tools. The direction of a curve is indicated by the small black triangle.

Just select desired curves and click Swap button on the control bar to change their direction.

Inverting curves

The Invert tool changes the curve so that it passes through its original control points. The result is the same as if you created the curve using the Interpolate option in the first place.

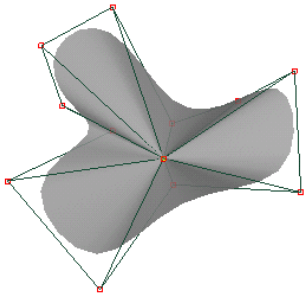
Using the Extrude option

Another way of creating curves is using the curve tool that is in the Tools/Unified menu. This curve tool also supports the Extrude option. When the Extrude option is set, the curve tool creates various NURBS surfaces instead of curves. For example, to create a conical surface with a freeform cross section:

1. Activate the Unified/Curve tool.

2. Set the Extrude option.

3. Enter a number of control points to define a cross section for the extruded surface. When done, click Accept. You can now define the position for the apex. Click the mouse in the desired point and select Accept.



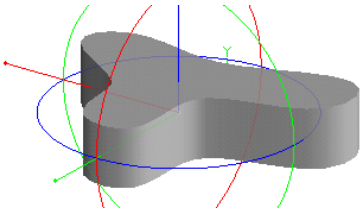
A conical surface with a freeform cross section

If you want to create a cylinder with a freeform cross section:

1. Activate the Unified/Curve tool. Set the Closed option from the control bar.

2. Set Extrude. You may also set Filled to get lids for both ends of the cylinder.

3. Enter the cross section points and select Accept. The curve tool starts to outline the position for the apex. Don't enter the apex point. Instead, select Accept again.



A cylindrical object created by the curve tool with Extrude and Filled options

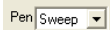
Using the Pen option

The NURBS Curve tool allows you to construct surfaces interactively. This can be achieved using the Pen option. By default, the pen is just a single point - sweeping it in 3D space creates a curve. When the Pen option is activated, the selected object is taken as a 'pen'. For example, by using a circular curve as a pen for the NURBS Curve tool, one can create circular tubes.

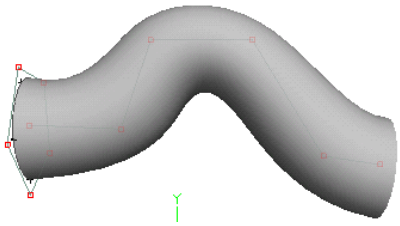
Let's try this:

1. Create a NURBS circle using the NURBS Circle tool from the side view.

2. Make sure the circle is selected and activate the NURBS Curve tool. Then set the Pen option to Sweep.



Set Pen to Sweep



Create a tube by using a NURBS circle as a pen for the curve tool

3. In the front view, define a desired profile curve for the tube. Make sure you click the first profile point over the selected circle. A circular tube is shown while you define the profile curve.

You can create open surfaces by using an open curve as a pen. If you use a polygonal curve as a pen for a cubic curve, the result is a surface which is polygonal in one direction and cubic in the other direction.

The Rotate pen rotates the drawn curve about the selected axis object. For example, to create a banana:

1. Create a slightly bent axis curve, which travels in the middle of the banana.

2. Select the axis curve, start the NURBS Curve tool, set Pen to Rotate and enter a profile curve for the banana.

Constructing surfaces is described in more detail in other NURBS modeling tutorials.

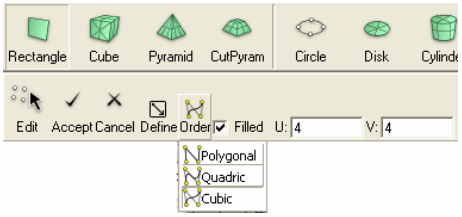
NURBS Surfaces

Many of the direct curve manipulation tools, such as Break and Sharpen, can also be applied to surfaces. The only difference is that surfaces have two parameter directions (called u and v) in which they can be manipulated.

In this tutorial, we will demonstrate these direct NURBS mesh manipulation tools.

The NURBS tab contains tools to create basic NURBS surfaces with freeform, circular and rectangular shapes.

Let's create a rectangular cubic NURBS mesh, which consists of 4 x 4 control points:



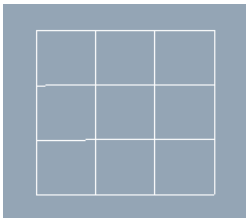
NURBS Rectangle tool activated. Options for creating a 4x4 cubic mesh.

1. Activate the NURBS/Rectangle tool.

As always, the control bar shows you the tool specific options.

Set Order to cubic, make sure Filled is checked and specify the control point count 4x4 through the U and V fields.

2. Then enter two points through a view window to define a geometry for the rectangle.

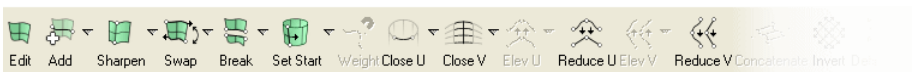


A rectangular NURBS mesh consisting of 4x4 control points

Curve handles

To switch to the curve editing mode:

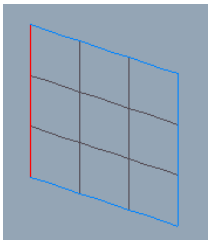
1. Select the mesh. As usual, the control bar shows you the tools, which can be applied to the selected mesh.



NURBS mesh specific tools

2. In the control bar, set the Edit control to Curves. Just click on the Edit icon and select Curve from the drop down menu. The mesh now shows you two vertical and two horizontal curve handles. When you move the mouse over a curve handle, the color of the curve is changed. As usual, you can move any curve simply by dragging it with the mouse.

You can also multi select any number of curves and apply any transformation or deformation tool to them.



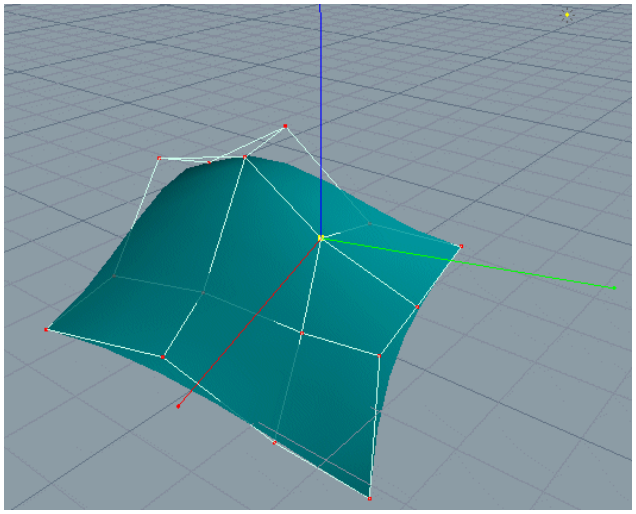
The left vertical curve selected and moved upwards

Control point handles

To enter the control point editing mode, set the Edit control in the control bar to Point to access the control point handles.

Hint: use Compass menus to switch between different editing modes easily.

Just drag the control points, using the modification handles with the mouse to move them.



Control point handles shown.

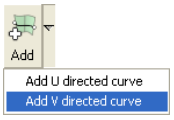
Adding new curves

Just like you can add a new point to a curve, you can add a new curve to a mesh.

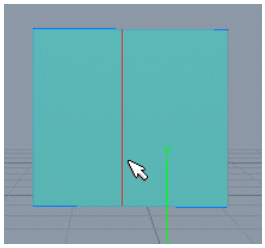
1. Use the NURBS/Rectangle tool to create a planar cubic NURBS mesh consisting of 4x4 control points.

2. Select the Add V Directed Curve pop-up menu item of the Add tool.

Hint: you can also use the compass menu to activate the tool.



Activate the Add Curve tool with the V Directed Curve option



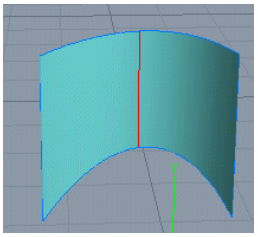
Activate the Add tool and LMB drag over the mesh to find a desired position for the new curve

3. Now press down the left mouse button over the mesh. Move the mouse while still holding down the left mouse button. The tool renders the new curve over the mesh. When a desired position is found, just release the mouse.



Note

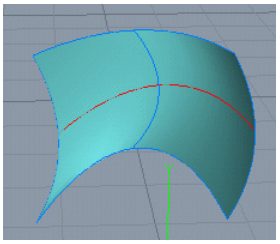
The shape of the mesh is not affected by the new curve. So it does not make any sense to add new curves without modifying them.



The new curve moved and scaled

Similarly, you can add a new horizontal curve to the mesh. Just select Add U Directed Curve from the pop-up menu of the Add tool and drag over the mesh to define the position for the curve.

Most of the direct mesh manipulation tools work this way.



A new U directed curve added and moved

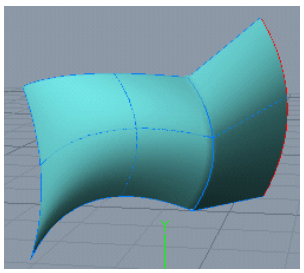
Creating sharp curves

The Sharpen tool works similar to the Add tool. The only difference is that the inserted curve breaks the continuity of the mesh so that a sharp edge can be formed.

Let's imagine we need to add a sharp V directed curve to the mesh we created above.

1. Select the Add V Directed Curve of the Sharpen tool and drag over the mesh to specify a position for the curve. The workflow is exactly the same as demonstrated in the previous example.

The Sharpen tool doesn't change the shape of the mesh. However, the nature of the new inserted curve is revealed as soon as you modify the mesh. For example, move the rightmost vertical curve upwards a bit.



Sharp v-directed curve added and the rightmost curve moved upwards

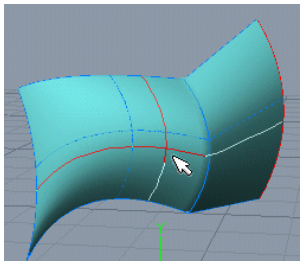
Breaking meshes

The Break tool breaks the mesh through the specified curve. Again, the workflow is similar to the previous Add and Sharpen tools.

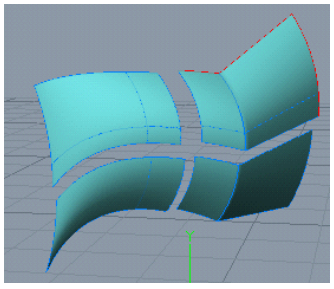
Let's split the mesh we created in the previous example into four parts:

1. Click the Break tool button.

2. Drag over the mesh to define the positions for the U- and V-directed curves.



The break tool activated with both U and V options

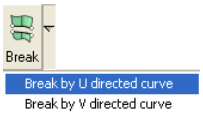


Pieces moved apart

As you can see, two perpendicular curves passing through the specified point are rendered over the mesh. The reason for this is that clicking the Break button selects both U and V options. This is also true for Add and Sharpen tools.

When a desired position for the curves is found, just release the mouse and the mesh is split into four parts by the curves.

If the mesh is closed in the given direction, the specified curve just opens the mesh through the specified curve. Let's demonstrate this by applying the break tool to a NURBS sphere. The surface of a sphere is closed in its U direction but open in its V direction.



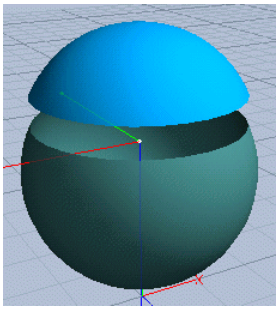
Break by U Curve

1. Switch to the top view and create a cubic NURBS sphere by using the NURBS/Sphere tool.

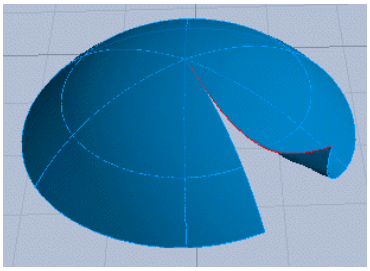
2. Switch to the front view and select Break by U Directed Curve pop-up menu item of the Break tool.

3. Drag over the sphere to find a desired break curve. When found, release the mouse to accept the tool.

Because the U directed curve forms a closed loop (the mesh is closed in its U direction), the tool split the surface into two separate parts. Correspondingly, the result is two separate NURBS meshes.



North pole cut off



Hemisphere broken by a v-directed curve and edited

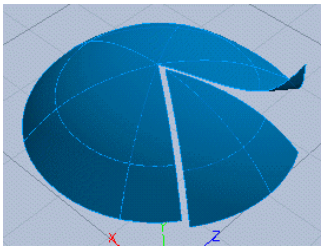
So, what happens if we break the mesh in its V direction? In that direction the surface is open: All V directed curves have end points at the poles of the sphere.

4. Select the north-pole mesh and activate Break by V Directed Curve. Then drag the mouse over the hemisphere to find a desired position for the break curve. When found, release the mouse.

As you can see, new objects were not created. Also, the shape of the mesh remains exactly the same. However, the mesh was opened through the specified curve, as you can see by modifying the mesh.

Now the semi sphere is open in the both parameter directions. Let's apply Break by V Directed Curve again.

Now the result is two separate meshes.



Defining start point of a closed mesh

Just like you can redefine the start point for a closed curve, you can redefine the start curve for a closed surface. Again, the only difference to the curve start tool is that now there are two parameter directions U and V in which the start curve can be defined.

Weights

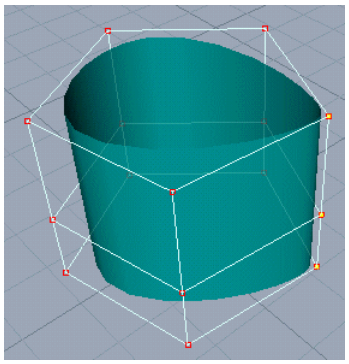
NURBS meshes support a parameter called Weight.

The weight parameter controls how strongly the control point in question affects the corresponding surface point (knot point). The higher the weight, the stronger the control point pulls the surface towards the control point.

You can modify the weight parameter interactively as follows:

1. In Edit mode, select the desired control points.

2. Select the Weight tool and click the mouse in a view window. By moving the mouse up/down you can increase/decrease the weight of the selected control points. Click the mouse again to accept the tool.



A high weight value set to three selected control points



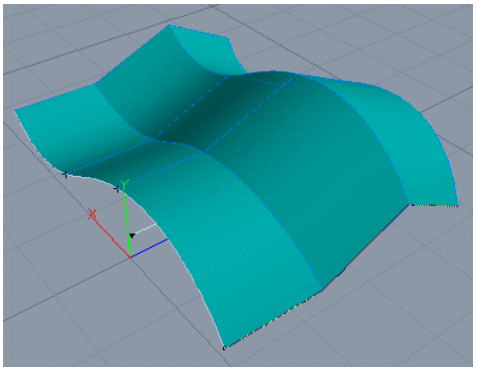
Note

You can also modify the weight of the selected control points by dragging them while holding down the Alt key.

Degree elevation/reduction

The *Degree Elevation* and *Degree Reduction* tools modify the order (degree) of the mesh. We already demonstrated the effect of degree elevation/reduction in the context of NURBS curves. The only difference to NURBS curves is that NURBS meshes have two directions in which the order can be controlled independently.

Polygonal, quadric and cubic meshes are sufficient for most practical applications. Surfaces with higher degree are sometimes needed when an extremely high surface smoothness and rendering quality is desired.

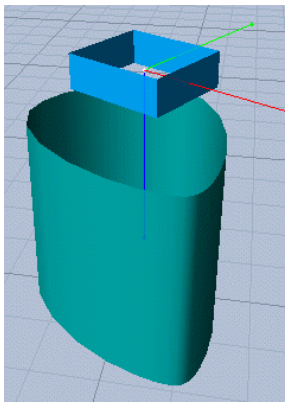


A mesh, which is cubic in u-direction and polygonal in v-direction

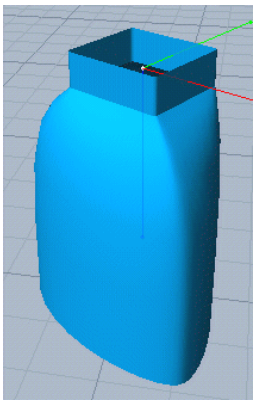
Concatenating meshes

The Concatenate tool concatenates two meshes with each other provided that the meshes have an edge i.e. they are open at least in one parameter direction. The tool concatenates the two nearest edges.

To use the concatenate tool, move the meshes to be concatenated so that the edges, which should be connected with each other, are placed side by side. The tool finds the nearest edges and connects them.



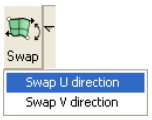
Original meshes



Concatenate tool applied

Swap

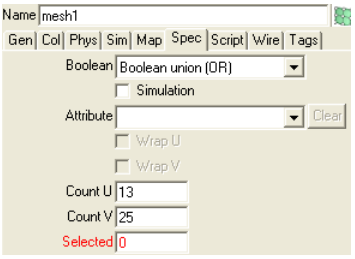
Sometimes you have to swap the direction of a mesh. For example, it may be necessary to invert the direction of a mesh to get it concatenated properly to another mesh.



The Swap tool swaps the direction in both u- and v-directions. To swap the direction in one direction only, select the appropriate pop-up menu item from the tool button.

Controlling the resolution of a mesh

Let's imagine you have a mesh and you need to double the number of control points without changing the shape of the mesh. Editing the resolution attributes in the property window/spec tab does this. For example, you can double the value of the Count V field.



Edit the Count U/V fields to control the resolution of the mesh



Note

If the resolution is increased, the shape of the mesh is not changed. Decreasing the resolution usually changes the shape.

Opening/closing meshes

You can close an open mesh and open any closed mesh by clicking the Close tool. The tool opens the mesh from its start curve.

If you have a number of meshes and you want to make them all closed, select the Close pop-up menu item associated with the Close tool.



Note

You can also use the Break tool to open a closed mesh from an arbitrary position.

Inverting meshes

The Invert tool allows you to change the mesh so that it will pass through its original control points. Just click the tool button to apply the tool.

For example: if you import a polygonal mesh and want to convert it to a smooth cubic mesh, inverting the converted mesh compensates for the shrinking effect of degree elevation by smoothing.

Editing the current selection

The arrow keys (Up, Down, Left, Right) can be used for changing handle selection (curves/control points) in the Edit mode. The left and right arrow keys can be used to move the current selection in the U parameter direction. Correspondingly, the up and down keys allow you to move the selection in the V direction. This is very handy when

modifying complex meshes. For example, to expand the current selection in the U direction, hold down the Shift key and hit the left and/or right arrow keys to select more curves.

Converting other objects to NURBS mesh form

The To Nurbs tool converts geometric objects to NURBS form by approximating them with a set of NURBS meshes. For example, to convert an analytic cube and sphere to meshes:

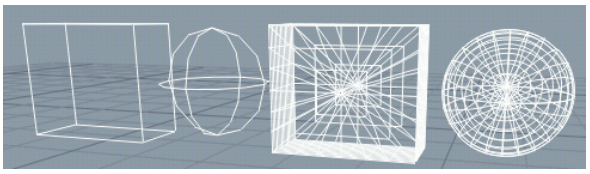


To NURBS tool

1. Multi select the cube and the sphere.

2. Activate the To Nurbs tool from the NURBS tool tab.

3. Specify the approximation density values Density U and Density V in the tool control bar. The default values U=1, V=1 define the minimum density. Set U=2, V=2. These values are sufficient for most cases. Click Accept to finish the conversion.



An analytic sphere and a cube converted to NURBS meshes

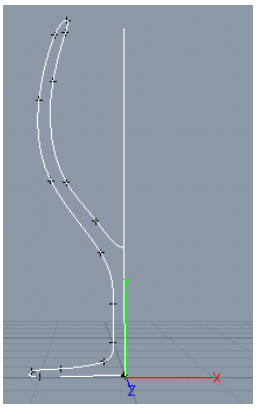
Rotational Surfaces

Objects with a circular cross section can be constructed from two NURBS curves: an axis of rotation and a profile curve. We use the classical example of creating a wine glass for the demonstration.

1. Use the NURBS/Curve tool to create a vertical line in the view window. The length of the line does not matter, except that a long line is helpful when drawing the glass profile. Rename it to 'axis curve'.

2. Create another curve representing a suitable wine glass profile. Rename it to 'profile curve'.

3. Multi-select the curves using the Shift key, the axis curve first, and the profile curve next.



An axis curve and a profile curve

4. The control bar now shows you the tools, which can be applied to the selected curves. Select the Rotate tool.

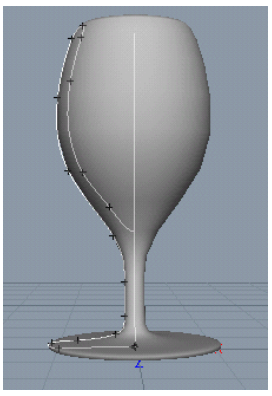


Rotate tool

The Rotate tool takes the first selected curve as an axis of rotation and the second curve as a profile curve.

If you accidentally selected the curves in the wrong order (in which case the tool rotates the axis about the profile curve), simply select the curves in the correct order again. There is no need to restart the tool.

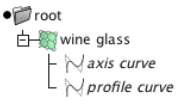
5. The default options are fine for this example - just click 'Accept'.



A wine glass

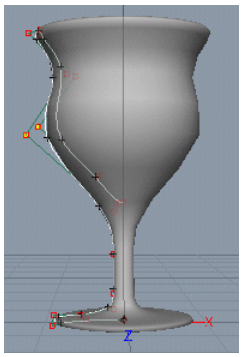
Editing the construction curves

NURBS mesh construction tools allow you to access the original construction curves and edit them in order to modify the shape of the mesh. For example, let's assume that the profile of the glass is not exactly what you expected and you need to make the wine glass slimmer.



The hierarchical structure of the wine glass

1. In the select window, click the '+' handle of the created NURBS mesh object. You can now see that the NURBS mesh includes two sub objects - the curves you used to create the wine glass in the first place.



Control points of the profile curve modified

2. Select the profile curve and enter the 'Edit' mode (press Space bar) to access the control points of the curve. Modify the profile curve. The shape of the wine glass is updated with the curve.



Note

In addition to moving points, you can add new points, delete existing points, add sharp edges, and so on. You can even replace the entire profile curve with another curve to rebuild the shape of the glass. If you model wine glasses a lot, you may build a library of profile curves. Just drag & drop the desired profile curve inside a wine glass mesh to get a desired wine glass constructed!



Note

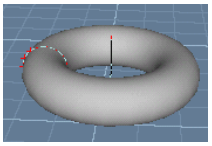
You might want to control the 'Forced visibility' attribute of the NURBS curves when working with the shaded OpenGL mode. This attribute makes the curves visible even if there they are hidden by the actual surface of the wine glass. The higher the 'forced visibility', the better they can be seen. You can access this option in the property window's 'Wire' tab.

Tutorial project: 'tutorprojects\modeling\nurbs\rotate\wine glass'

Torus

A torus surface can be created by rotating a NURBS circle about an axis.

Tutorial project: 'tutorprojects\modeling\nurbs\torus.r3d'

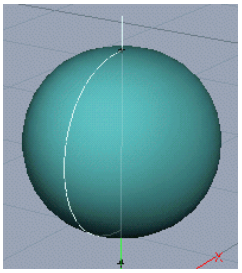


Rotate a circular curve about an axis to get a torus

Sphere

Rotating a semi circle about an axis creates spheres and ellipsoids.

Tutorial project: 'tutorprojects\modeling\nurbs\rotate\sphere'



Rotate a semi circle about an axis to build a sphere

Interactive Method

The Nurbs Curve tool's Pen option allows you to create rotated surfaces interactively.

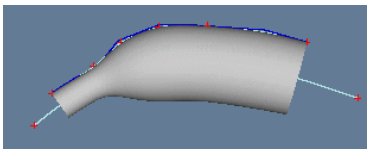
1. Create or select a NURBS curve representing an axis. Note: the axis curve does not have to be a straight line. Any curve will do.



Set the Pen option to Rotate

2. Activate the NURBS/Curve tool and set the 'Pen' option to 'Rotate'.

3. Enter points for the curve as usual. New curve points are rotated about the selected 'axis' curve while you enter them.

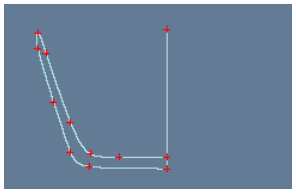


Create a banana with the Rotate pen

Tutorial project: 'tutorprojects\modeling\nurbs\rotate\banana'

Scale curve

If only two curves are selected, the rotation tool rotates the second curve about the first curve. A third curve can be selected to control the properties of the mesh in its 'v' direction.

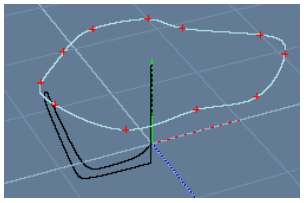


A side view of the axis and the profile

For example, to create a vase with a non-circular cross section:

1. Create an axis and a profile curve from the side view.

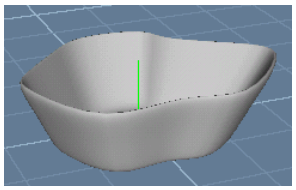
2. Switch to the top view and create a closed scaling curve.



The axis, the profile and the scale curves created

3. Select the axis, profile and scale curves (in this order) and click the Rotate tool in the control bar. Then click 'Accept' to finish the mesh construction.

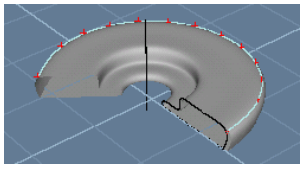
Tutorial project: 'tutorprojects\modeling\nurbs\rotate\vase'



A non-circular vase

If the scale curve is open, also the mesh is open in the corresponding direction. By using a 'helix' for the scale curve, one can create screws and other objects with a spiral shape.

Tutorial project: 'tutorprojects\modeling\nurbs\rotate\open scale curve'

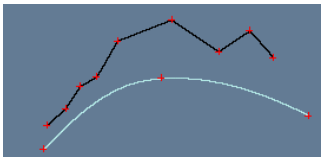


A circle arc used as a scale curve

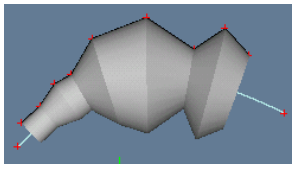
Controlling the order of the surface

Let's create another banana.

1. Create two NURBS curves: a slightly bent axis curve and a profile curve. However, this time, make the profile curve polygonal by setting the 'Order' option of the Nurbs Curve tool to 'Polygonal'.



Rotate a polygonal profile curve



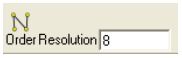
Polygonal curve rotated

2. Select both curves: first the axis curve and then the polygonal profile curve. Activate and accept the Rotate tool.

The result is a mesh, which is polygonal in the 'v' direction and cubic in the 'U' direction.

Tutorial project: 'tutorprojects\modeling\nurbs\rotate\polygonal profile'

We can also create a banana, which is polygonal in its closed direction but smooth in the axis direction. The Rotate tool provides the necessary controls for this.

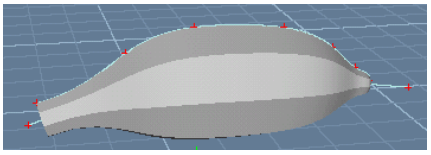


Rotation controls, which determine mesh properties in the direction of rotation

1. Create two smooth cubic NURBS curves: a bent axis and a profile for the banana.

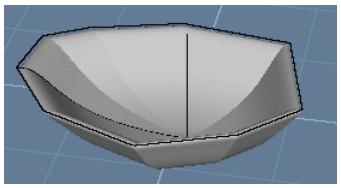
2. Select the curves and activate the 'Rotate' construction tool as shown above. The control bar now shows you the rotation specific options.

3. Set Order to Polygonal and click Accept.



Polygonal rotation

When using a 'scale' curve, the order of the mesh is fetched from it. For example, the following image demonstrates a combination of a cubic profile curve and a polygonal scale curve.



A cubic profile and a polygonal scale curve. The mesh is polygonal in the direction of the scale curve.

Sweep and Bi-Rail Tools

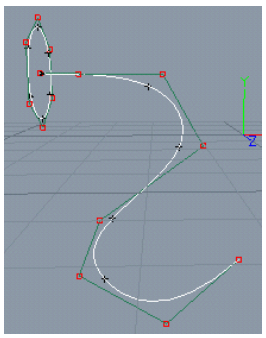
The Sweep tool constructs a surface by moving a cross section curve along one or two rail curves.

An 'S' shaped tube can be created by sweeping a circular cross section curve along an 'S' shaped rail curve:

1. Switch to the front view and create a closed NURBS curve.

2. Switch to the side view and create an 'S' shaped open NURBS curve which starts from the center of the first curve.

3. Select the cross section curve and the rail curve (in this order). As always, the control bar shows you the tools, which can be applied to the selected curves.



A closed cross section curve and an open rail curve

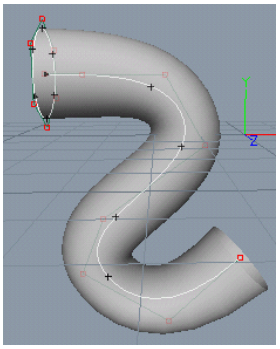
4. Click the Sweep tool.

There are two ways to sweep the circular cross section curve on the rail curve.

- Orthogonal sweep. The direction of the cross section curve is rotated with the curvature of the rail.
- Coplanar sweep. In this method the direction of the cross section circle remains the same.

Let's use the orthogonal sweep, so set this option in the control bar. Then click 'Accept'. Leaving the 'orthogonal' option unchecked in the control bar, results in a coplanar sweep.

Tutorial project: 'tutorprojects\modeling\nurbs\sweep\tube'



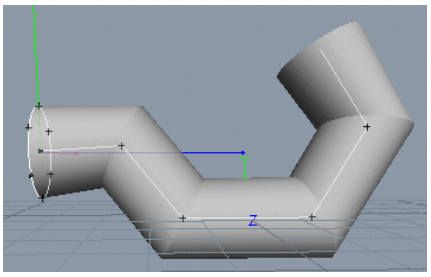
A tube created by sweeping a closed curve along an S-shaped open curve

Controlling the order of the surface

Let's assume you want to create a circular tube, which consists of straight cylinders. This can be constructed by sweeping a smooth circular cross section curve along a polygonal rail curve.

To create such a tube, just replace the sweeping curve of the previous example with a polygonal curve.

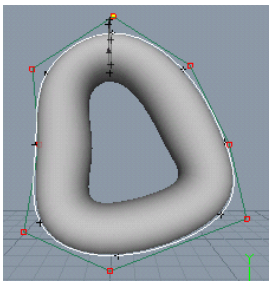
The curves, from which the surface is constructed, determine the mesh properties. If you want to create a completely polygonal NURBS surface, use two polygonal curves.



A cubic circle swept along a polygonal rail

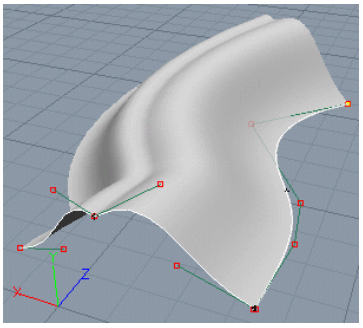
Tutorial project: 'tutorprojects\modeling\nurbs\sweep\polygonal rail'

In addition to Order, also the Open/Closed attributes of the mesh are determined by the corresponding state of the construction curves. A torus surface is closed in both directions. Such a surface can be swept by using two closed curves. Remember to start drawing the rail curve from the middle of the cross section curve.



A torus surface created by sweeping a circular cross section curve along a closed rail curve

The weight and parametrization of the construction curves control parametrization and weights of the mesh. For example, if you increase the weight of a control point, the weight of the corresponding 'isoparametric curve' is changed. Or, if you use a Bezier curve as a rail, also the surface will be Bezier-parametrized in the corresponding direction.

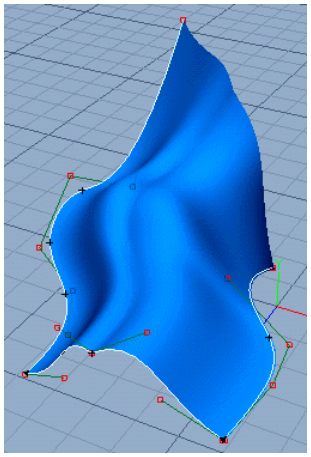


A Bezier curve used as a cross section curve

Bi-rail

The Sweep tool supports also two rail curves. Bi-rail sweeping activates automatically when you select three parameter curves for the tool. The first selected curve is taken as a cross section curve and the remaining two curves are used as rails to move, rotate and scale the cross section curve.

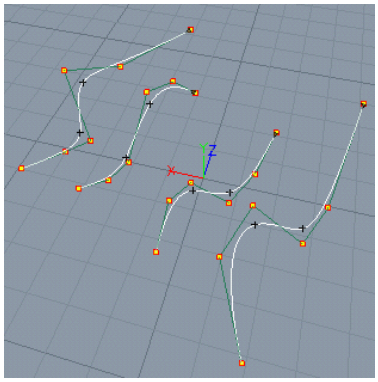
Tutorial project: 'tutorprojects\modeling\nurbs\sweep\bi-rail'



The bi-rail tool: a Bezier curve swept by two rail curves

Cross Section Surfaces

The Cross Section tool constructs a surface, which passes through a given set of cross section curves. To create a cross section surface:

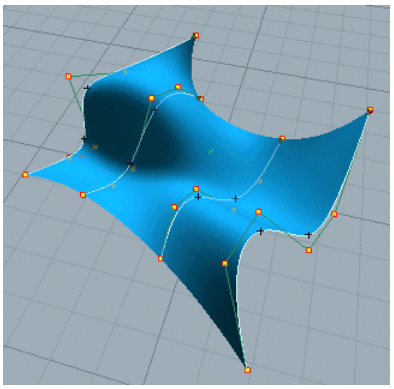


Four cross section curves

1. Create a set of NURBS curves.

2. Multi select the curves in the order in which the surface should interpolate through the curves. The control bar now shows you NURBS specific construction tools. Click the X-Sect tool.

 *Cross Section tool*



A NURBS surface created through the selected curves

You can now see two cross section tool specific options in the control bar: Closed and Order. We will create a cubic mesh through the curves. Therefore, set the Order to Cubic and click Accept to create a surface.

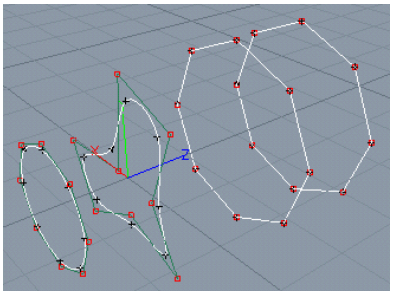
Tutorial project: 'tutorprojects\modeling\nurbs\cross_section\cross_sections'

The cross section tool can handle curves of any order. Some of the curves can be polygonal and some cubic, for example.

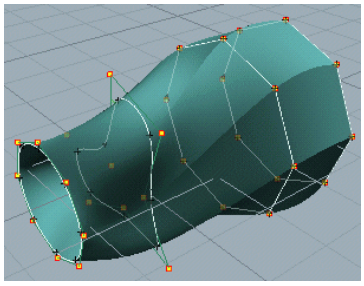
Let's create a tube, which starts with a round smooth shape and ends to a polygonal form.

1. Create two closed cubic NURBS curves and two closed polygonal curves, as shown in the next picture.

2. Select the curves from left to right (in the order in which you want to connect them). Activate the XSect tool, set Order to Cubic and click Accept.



Four closed curves defining the cross sections of the tube



A tube created through four closed curves with different orders

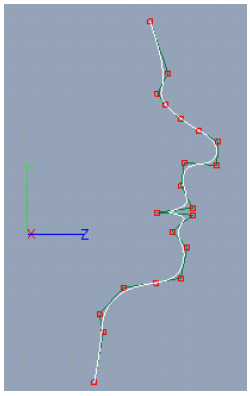
Creating a face from cross section curves

This tutorial demonstrates how to create a human face using the cross section tool.

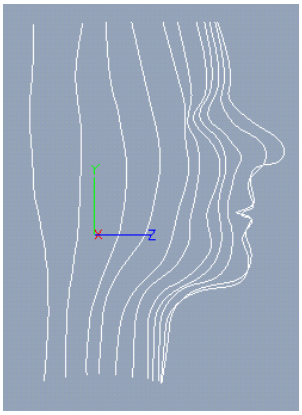
1. Create a curve representing the profile of a face.

2. Duplicate the profile curve and modify it to get the next 'slice' of a face. Continue this way until you have the necessary cross section curves for one half of a face.

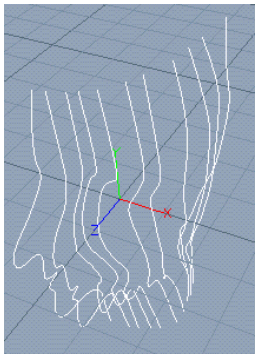
3. Now displace the curves by rotating them from the top view. You can also single point edit the curves from the front view to get non-planar curves.



A NURBS curve representing the profile of a face



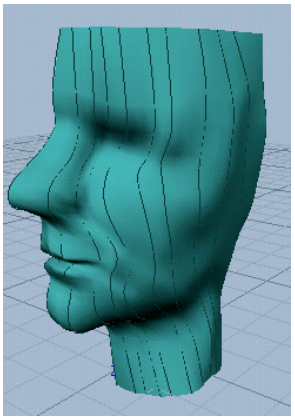
Cross section curves for one side of the face



Face curves seen from the top

4. Multi select the curves in the correct order and apply the XSect tool. Set the Order option to Cubic and click Accept.

Tutorial project: 'tutorprojects\modeling\nurbs\cross_section\semi head'



One half of a face defined by a number of vertical cross section curves

Editing construction curves

As usual, the original construction curves are placed inside the mesh and you can modify them in order to control the shape of the mesh.

If you delete one of the construction curves, the corresponding curve is also removed from the actual mesh. For example, to make the head we created in the previous example complete, duplicate and mirror the cross section curves and then drop the duplicates to the beginning of the construction history in inverted order (the order of the doubled curve set must be correct).

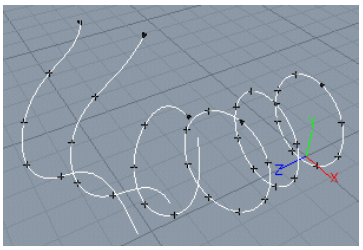


Note

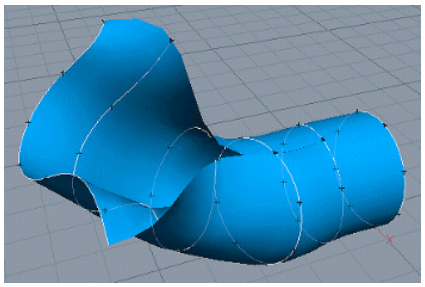
Note: If you need to delete the cross section curves but you don't want to delete the mesh, switch the Constructor of the mesh to None from the property window's Spec tab before deleting the curves.

Mixing open and closed curves

You can freely mix open and closed curves to define a cross section surface. For example, you can create a tube through three open and three closed curves, as demonstrated by the following image:



Three open and three closed curves



The result surface contains both open and closed sections

Extruded Surfaces

The Extrude tool can be used, for example, for creating 3D logos. The tool can handle nested curves by automatically detecting which one of the curves defines holes. You can also control the beveling shape, radius and other extrusion options.

Tutorial project: 'tutorprojects\modeling\nurbs\extrude\logo'

Let's create a simple logo whose depth is 0.1 m and edge rounding is 5 mm.

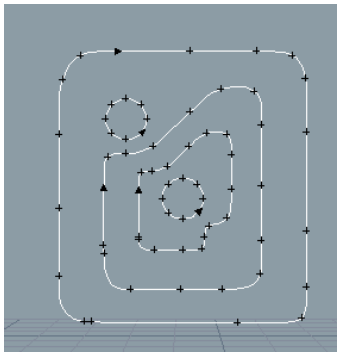
1. Create a set of closed NURBS curves representing a desired logo shape.



Extrude tool

2. Select the curves in any order and click the Extrude tool in the control bar.

3. As usual, the control bar now shows you a number of extrude tool specific options. Set Beveling Type to Rounded and find an appropriate beveling Radius and extrusion Depth. When found, click Accept.



Nested curves for a simple logo



Specifying extrusion options

The Extrude tool also creates an Extrusion Manager object. This object controls the beveling radius and other extrusion specific options - they can be changed any time.

For example, to change the type of the beveling of the logo we created above:

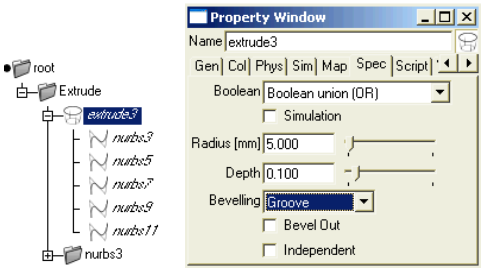
4. Select the extrusion manager object (it is called 'Extrude') and open the property window. Go to the Spec tab and select another beveling type.

The shape of the logo is updated instantly.

You can also modify the original construction curves, which are placed inside the extrusion manager object, to reconstruct the object.



The logo defined by the nested curves



Select the extrusion manager object and change the Beveling type from the property window



Note

The Extrude tool can extrude filled objects only from closed curves. For example, an 'S' shaped curve cannot be extruded that way, because the curve does not have well defined 'interior'. Therefore, the tool cannot determine the direction of the beveling or nesting of multiple curves. Extruding open curves makes sense if the No Lids option is set.



Note

If the Independent option is checked, each selected object is extruded independently into a separate level. By default this option is not set.

If you wish to create holes and you have checked the Independent option, you need to put curves into a level and select the level before extruding. In this case a hole will be created. If just just select your curves and set the Independent option, then each curve will be extruded separately and no hole will be created.

Trimming NURBS Surfaces

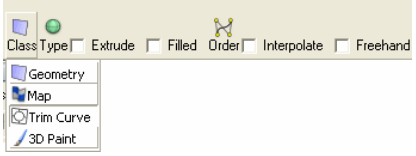
Trim curves can be used for cutting NURBS meshes. For example, one can drill a hole in a NURBS mesh using a circular trim curve. There are several ways to create a trim curve. You can select various trim curve tools from the NURBS tool tab. Also, you can use the Unified tools from the Tools - Unified menu, like shown in the following examples.

The unified creation tools create trim curves with a freeform, circular or rectangular shape.

To create a freeform trim curve:

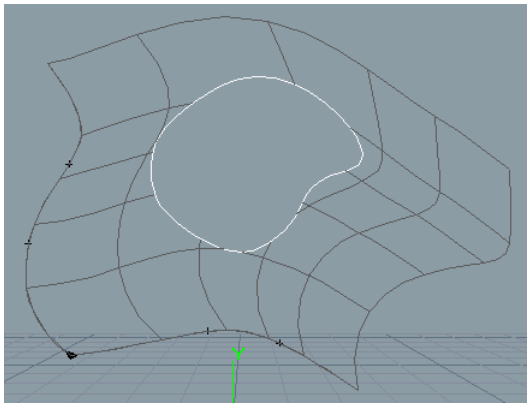
1. Create a NURBS mesh using the Sweep tool, or simply use the NURBS rectangle tool and set the U and V parameters to a value of 8.

2. Select the mesh and activate the Tools/Unified/Create curve tool from the menu. As usual, you can now see the curve tool specific options in the control bar. Set the Class option to Trim Curve and make sure the Closed option is checked.



Set the Class option to Trim Curve to create a trim curve

3. Click the mouse over the mesh to define control points for the trim curve. When done, accept the tool.

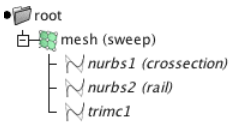


A trim curve created for the selected mesh



Note

A trim curve is an object whose geometry is defined in the *parameter space* (also called the uv space) of the parent mesh. Therefore, when creating trim curves, the position of the mouse pointer must be within the borders of the parent mesh.



A trimmed mesh

The mesh now consists of three curves: the original construction curves plus one trim curve.



Note

If you created the first mesh using the NURBS rectangle tool, the hierarchy consists of only the mesh and the trim curve.

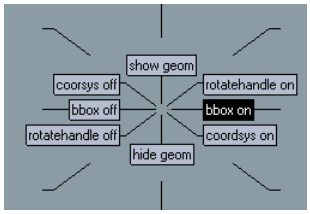
Modifying trim curves

Trim curves can be modified just like ordinary NURBS curves. For example, to move the trim curve to the left top edge of the mesh, just select the trim curve and use the Move tool as usual.



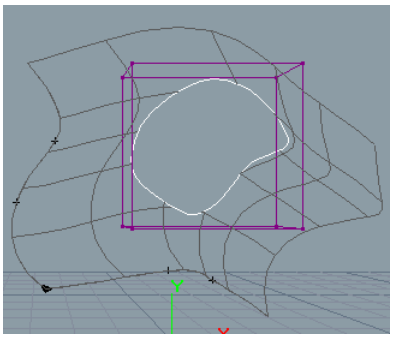
Note

In order to transform a trim curve, you always have to click the mouse **over** the mesh. Otherwise, the transformation tool will not be able to compute the necessary transformation in the parameter space of the mesh.



Switch on the bounding box through 'w' compass menu

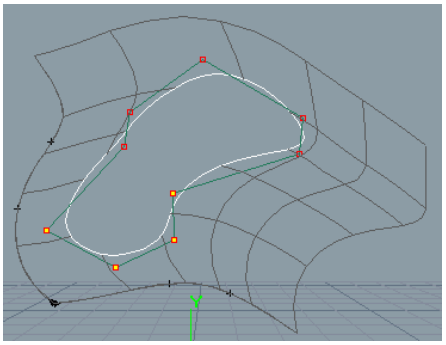
You can also use the standard transformation handles to modify trim curves. As usual, you can switch transformation handles on and off using the 'w' compass menu. For example, you can modify trim curves through their bounding box. To be able to see the bounding box, select the trim curve and right click in the view window, to get the pop-up menu. Select Handles/Bounding box. Then use the 'w' compass menu like shown here. If the planar mesh is flat, the bounding box has a rectangular shape.



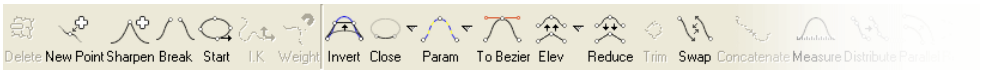
Move the trim curve by dragging the bounding box

Just like regular NURBS curves, also trim curves can be single point edited in the edit mode. As usual, hit the space bar to enter the edit mode and drag points to move them. You can also multi select any number of control points and apply standard transformation tools, such as move, scale and rotate, to the selected points.

When you select a trim curve, the control bar shows you a number of tools which can be used for manipulating the curve. We have already demonstrated most of these tools in the context of NURBS curves.



A trim curve in the Edit mode

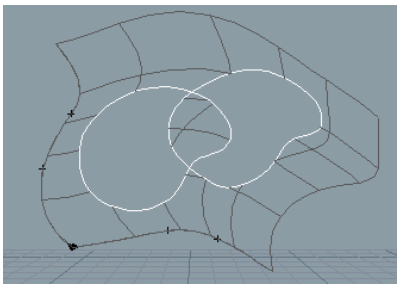


Trim curve specific tools

Trim curves can be duplicated in the same way as any geometric objects. For example, just hold down the copy modifier (Ctrl) while moving the trim curve.

You can copy trim curves using standard Edit/Cut, Paste and Duplicate tools. For example, cut a trim curve from one mesh and paste it inside another mesh. Or, use simple drag&drop in the select window to move trim curves in the object hierarchy. Because trim curves are defined in the parameter space of the mesh, all deformations you apply to the mesh will also deform the trim curve accordingly. For example, if you modify either of the two sweep curves, the trim curve gets modified with the mesh accordingly.

Tutorial project: tutorprojects\modeling\nurbs\trim curves\trimmed mesh



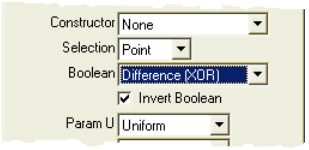
Ctrl+drag to duplicate a trim curve like any other geometric object

Boolean operations of trim curves

The general Boolean operations work also for trim curves. The difference is that since trim curves are defined in the two dimensional uv space', also the Boolean operations are applied in a two dimensional space.

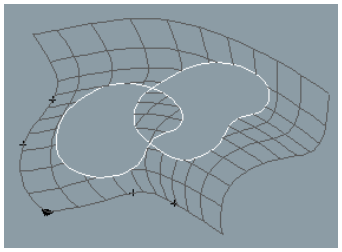
All the three standard Boolean operations are supported: Union (OR), Intersection (AND) and Exclusive Or (XOR).

Boolean operation is an attribute of the actual NURBS mesh. You can select the desired Boolean operation method from the property window's Spec tab.

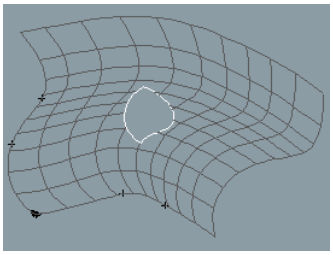


The Boolean attribute of the mesh

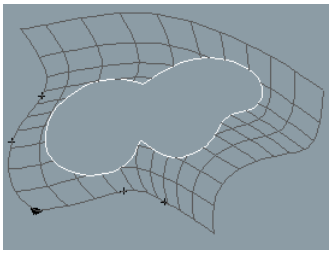
The following three images demonstrate the effect of different Boolean operations.



Difference (XOR)



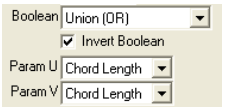
Intersection (AND)

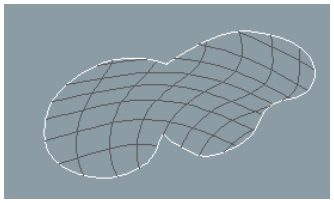


Union (OR)

Just like you can invert the Boolean operation of a regular 3D Boolean object, you can invert the Boolean operation between the trim curves of a mesh. Just check the Invert Boolean option through the property gadget of the NURBS mesh.

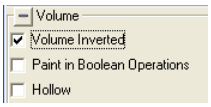
The Invert Boolean option inverts the inside/outside property of trim curves so that what was inside before becomes outside.





The property window for the NURBS mesh: Boolean operation set to Union and result Inverted

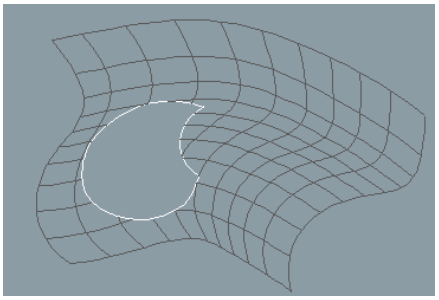
In addition to inverting the top level Boolean operation of trim curves associated with a mesh, you can invert an individual trim curve.



Trimming side inverted

To do this, select the trim curve and go to the property window's Gen tab. Check the Volume Inverted property (see the Boolean operations between true 3D objects).

These three attributes give you total control over Boolean operations between trim curves.



Trim curve 1 AND NOT trim curve 2

Converting trim curves to regular NURBS curves

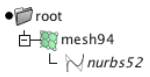
Let's assume you have a trim curve and you need to convert it to a standard NURBS curve. The To 3D tool does this. Just select a trim curve and click the To 3D tool to map the trim curve to 3D space.

This tool creates a regular NURBS curve, which passes through the control points of the trim curve. As a result, the shape of the 3D curve changes! The reason is that the uv space of a mesh is usually curved whereas the 3D model space is neatly linear. The more control points the trim curve has, the more accurately the curves match.

Converting NURBS curves to trim curves

It is also possible to map the control points of a regular NURBS curve to the parameter space of a mesh.

To create a trim curve from a NURBS curve:



Create a NURBS curve inside a mesh

1. Create a NURBS mesh using, for example, the NURBS/Rectangle tool.

2. Create a closed NURBS curve over the mesh. In the select window, drag & drop the curve inside the mesh. You now have a NURBS mesh, which includes one NURBS curve.

3. Select the NURBS curve. The control bar now shows you a number of NURBS curve specific tools.  Click the Trim tool.



Note

Use the Trim Parent tool in the second row of the toolbar, not the icon of the Create Trimmed Surface tool described in the next example.



Note

In this operation, the shape of the created trim curve does not usually match the shape of the original curve exactly. The more control points the curve has, the more accurately the trim curve follows the original NURBS curve.

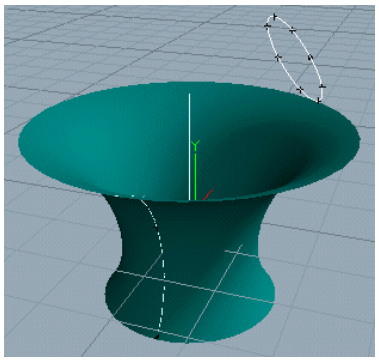
Projecting curves to trim curves

It is also possible to convert a NURBS curve to a trim curve by using the current transformation of a view window. In this method, the control points of a NURBS curve are not snapped to the nearest surface points. Instead, they are projected to the surface behind them.

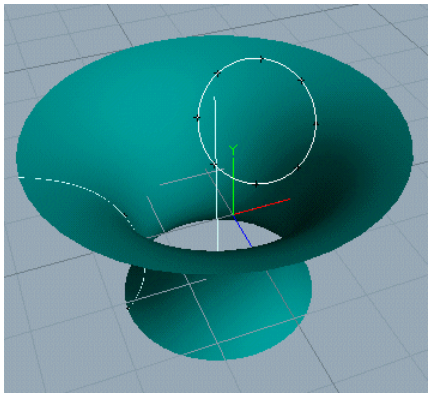
1. Create a curved NURBS mesh and a closed NURBS curve, as shown in the next picture.

2. Select the mesh and the curve. Then rotate the view window to find a desired projection.

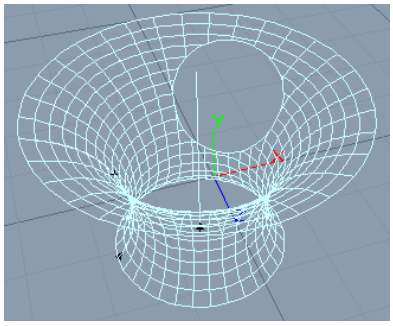
3. Then select the Tools/Nurbs Construction/Create Trimmed Surface tool and click Accept.



A NURBS surface and a NURBS curve



Rotate the view window until the curve is in right place over the mesh



A NURBS curve projected onto a mesh

Welding

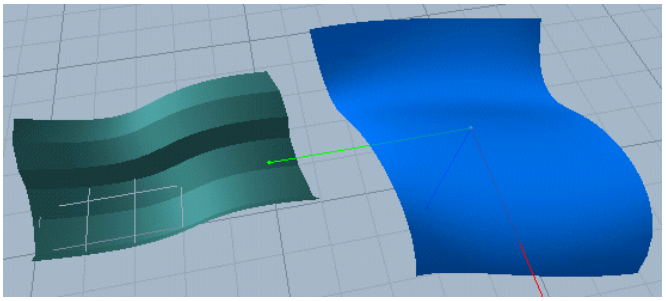
The Welding tool connects two surface edges with a special purpose weld object. Basically, the weld object is a NURBS mesh with some additional intelligence so that it maintains a seamless joint between the specified edges.

Any two edges with different point counts, orders and other properties can be welded.

To connect two meshes, one polygonal and one smooth surface, by welding:

1. Create two NURBS meshes: a polygonal one and a cubic one.

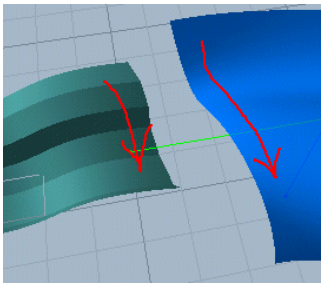
2. Select both meshes. The control bar now shows you the weld tool.



Two NURBS meshes with a different order and density

Weld tool

Activate the tool. The tool automatically welds the two surfaces. You can change the weld by dragging a line over the mesh, near the edge you want to weld. The line you draw also defines the direction of the weld mesh.



Mark two edges to be welded



Note

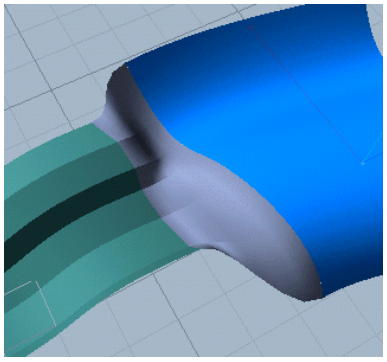
You have to drag **over** the meshes (inside the mesh boundaries), as shown in the example image

If you accidentally marked a wrong edge, you can drag again to fix the error. There is no need to restart the tool.

3. When the desired edges are welded, click Accept to finish the tool. The two meshes are now connected.

You can modify the original meshes after welding. The shape of the welding surface is updated accordingly.

Tutorial project: 'tutorprojects\modeling\nurbs\weld\weld'

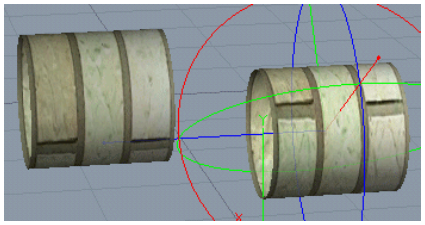


A polygonal and a cubic mesh welded

When welding closed surfaces (such as pipes, for example), there is an additional freedom to control the twist of the welding surfaces (the positions where the 'u=0' isoparametric curve meets the surfaces).

Let's connect two NURBS cylinders with each other to demonstrate this.

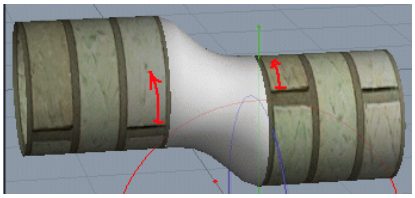
1. Use the NURBS/Cylinder tool to create two NURBS tubes (reset the Filled option).



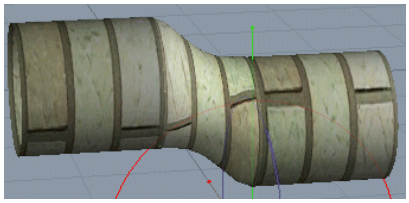
Create two NURBS tubes

In order to see the twist effect, switch to the wire frame rendering mode.

2. Select both NURBS tubes. The control bar now automatically shows you the Weld tool. Activate the tool and drag along the edges to be welded, then Accept the tool.



Mark an edge from both cylinders. Drag points determine the start and end points for the first isoparametric curve of the welding mesh.



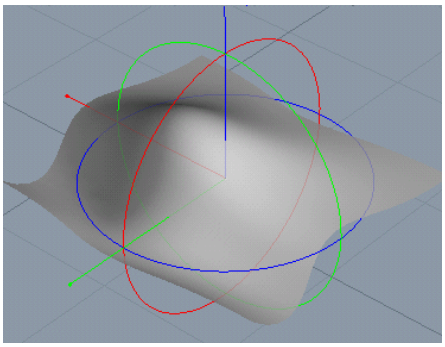
Welded tubes

Welding surfaces and trim curves

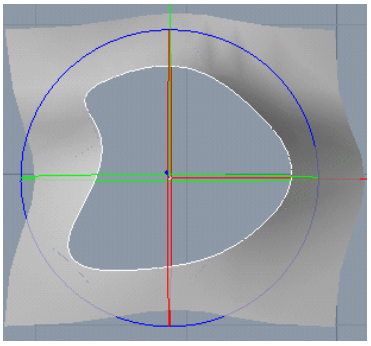
Welding tool also allows you to weld the edges of trimmed surfaces.

1. Create a NURBS surface.

2. Select the surface and select Unified/Create Curve from the Tools drop down menu. Set the Class option to Trim curve and enter a desired number of control points over the NURBS surface to trim it.



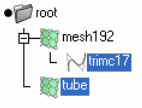
A NURBS surface



A trimmed surface

3. Use sweep or another appropriate tool to create a NURBS tube.

4. Select the trim curve and the tube and activate the Weld tool from the pull-down menu Tools/ NURBS Construction/Weld Edges.



Select the trim curve and the tube mesh



Note

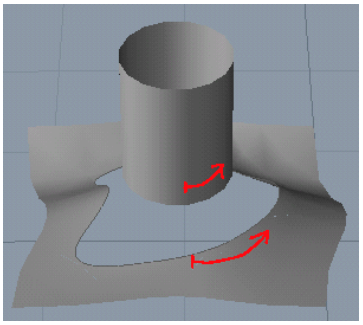
A mesh can contain any number of trim curves. You have to select the actual trim curve to which the weld mesh should be attached!

Mark an edge from both surfaces to weld them.

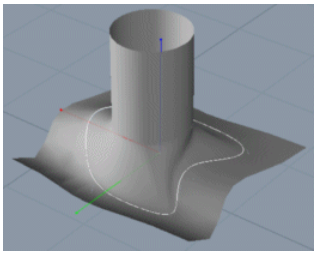


Note

Both edges are closed - by controlling the drag start points, you can control how twisted the weld surface will be.

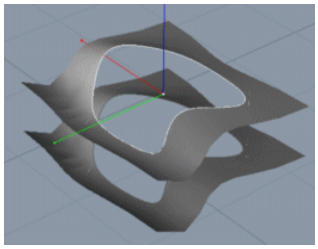


Mark the edges to be welded



The trimmed surface edge welded with the tube

Welding two trimmed surfaces

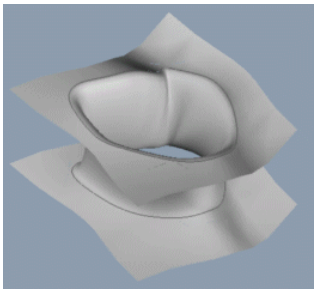


Two trimmed meshes

The Weld tool can also weld two trimmed meshes with each other.

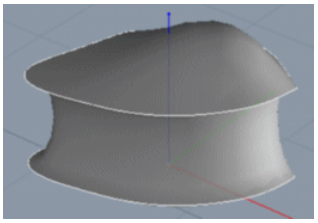
1. Create two trimmed NURBS surfaces somewhat parallel to each other.

2. Select both trim curves, click the Weld tool of the NURBS tool tab and mark edges to create a weld surface.



Two trim curves welded

The trimmed edges are now smoothly joined with each other. Let's play with this example a little bit more.



Trim curves inverted

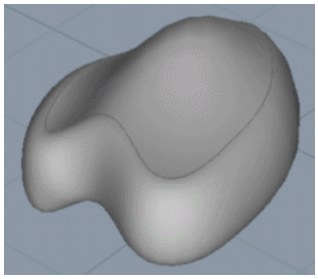
3. Select both trim curves and click the Invert tool in the control bar (it's located at the far right end).

This inverts the inside/outside property of the selected trim curves and they trim out the outer part of the surface.

4. Select the weld surface and open the property window. Go to the Spec tab. Here you can see two Size fields, which define how sharp the weld surface is at its boundaries. The smaller the value, the sharper the joint.



The radius fields of the weld surface



Weld surface with negative size attributes

If you want completely sharp edges, set these values to zero. We just want to change the direction of the weld surface (the edges of the weld surface should point outwards rather than inside).

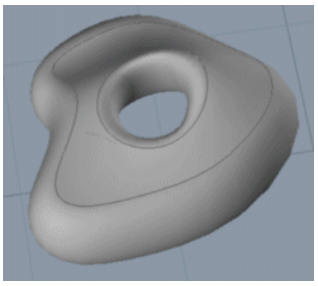
Change the sign of both the values (make them negative).

Tutorial project: 'tutorprojects\modeling\nurbs\weld\welded trims'



Note

You can create any number of trim curves to both surfaces and connect them with the weld tool. Also the weld surface itself can be trimmed and welded with another weld surface.



Two trimmed edges connected with the weld tool

Controlling the accuracy

When connecting the edges of the surfaces, the weld surface is capable of following the shape of both the edges exactly. However, when welding trim curves, it is mathematically impossible to create an exact weld surface for the trim curve.

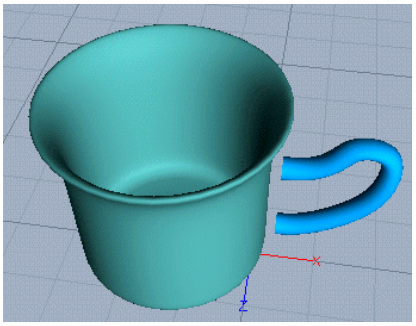
If the joint is not accurate enough, increase the number of control points of the trim curves. The higher the point count of the trim curves, the more accurate the weld becomes.

Fillets

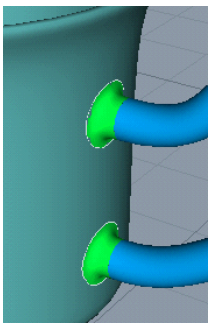
Let's imagine you have two NURBS surfaces representing a body and a handle of a coffee cup and you would like to blend the handle to the body seamlessly. The fillet tool can be used for this.

Let's try this.

1. Use the rotation tool to create a body of a coffee cup. Use the Sweep tool to construct the handle. Position the handle to its correct position, as shown by the example image.



Two meshes for blending



Note

Make sure both ends of the handle point towards the body of the cup.

2. Select the body and the handle (in this order) and click the Fillet tool. You can now see the fillet surfaces created between the body and the handle. As usual, the control bar shows you fillet specific options, such as the radius. Specify the desired radius and click Accept.

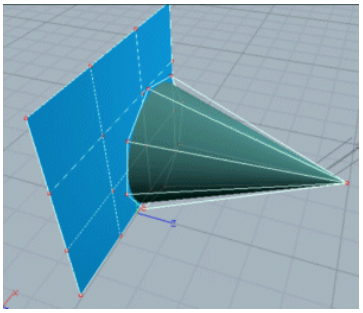
You can now move the handle (or the cup) and the shape of the created blending surfaces is updated. The handle will remain seamlessly blended with the cup.

Tutorial project: 'tutorprojects\modeling\nurbs\fillet\cup_handle_fillet'

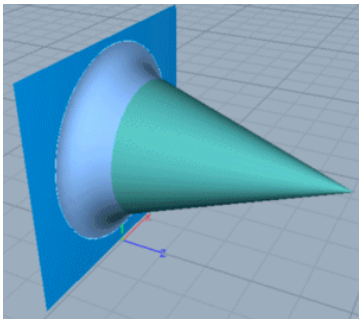
The handle blended with the body

The fillet tool can determine the shape for the blending surface by extrapolating the second selected object (the handle). The following example demonstrates this.

1. Create a 4x4 rectangular mesh. Then use the rotate tool to create a cone. Place the open end of the cone near the planar mesh as shown in the image:



A cone and a planar mesh



2. Select the mesh and the cone and activate the Fillet tool. Click Accept.

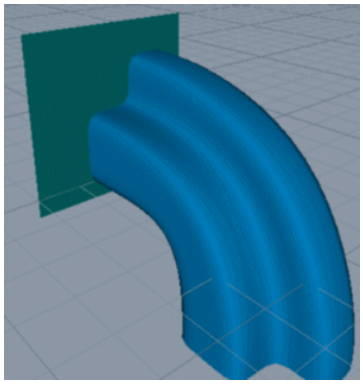
The created surface is tangential with both the cone and the planar mesh.

Note

If the cone is too big in relation to the planar mesh, the fillet tool may not be able to connect it to the planar mesh.

Tutorial project: 'tutorprojects\modeling\nurbs\cone_mesh_fillet'

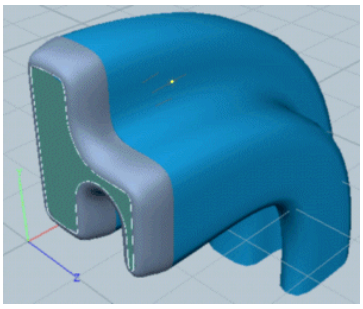
The fillet tool can also be used for creating a lid for the ends of a closed profile tube surface.



A planar surface and a tube

1. Create a closed profile curve and use the sweep tool to create a tube. Then create a planar surface, which will be used as a lid for the tube.

2. Select the planar mesh and the tube and activate the Fillet tool. Set the Trim option to Trim curves and this time specify a negative Radius. This makes the created surface bend inside rather than outside. Accept the tool.



A sweep surface blended to a planar mesh with a negative radius

3. Select the trim curve created inside the planar mesh object. The control bar now shows you the trim curve specific tools. The Invert option is set for the trimcurve. Uncheck the Invert option to change the trimming side.

Tutorial project: 'tutorprojects\modeling\nurbs\fillet\tube with lid'

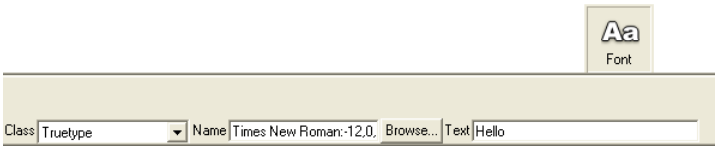
The Font Tool

The *Font* tool creates font outlines. Outlines are represented as regular NURBS curves. You can use them with all surface construction tools.

Let's experiment with the tool.

1. Activate the Font tool. It's in the NURBS tab.

You can now see a number of tool specific options in the control bar.



The Font tool activated

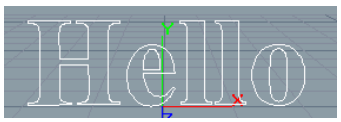
2. Specify the desired font properties and type in a string 'Hello' into the Text field.



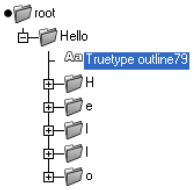
Note

Use the Browse button to bring up the standard font selector window and select a font, style and size.

3. Then enter two points through the view window to define a base line for the outline curves. When the second point is entered, the tool is automatically accepted and the curves are inserted to the scene.



Hello using Times New Roman

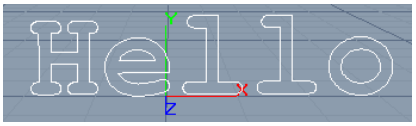


Select font manager

4. Let's assume we made a mistake. Instead of using the Times font, we should have used the Courier font.

In the select window, you can see an object whose name matches the text you specified into the Text field ('Hello', in this case). Open this level and you can see a font manager object. The font manager object is named according to the font class you specified (Truetype, for example). Select the font manager object.

5. Open the property window. In the Spec tab, change the font type to Courier New, for example. The curves in the view window are instantly updated to reflect the new selected font.



Hello with Courier font

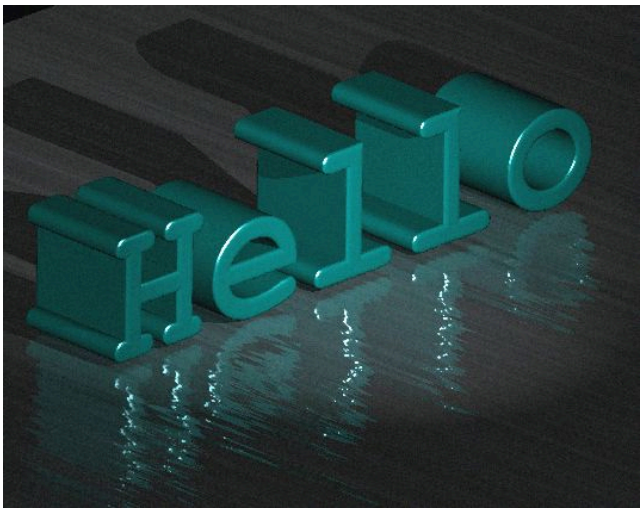
Let's end this tutorial by creating a 3D logo from the curves. The Extrude tool can be used for this.



Extrude tool

6. Select the entire 'Hello' folder from the Select Window. Click the Extrude icon in the NURBS toolbar.

7. Select and specify the desired extrusion options through the control bar. For example, set Beveling Type to Round. When done, click Accept to finish the tool.



Extruded outline fonts on a semi reflective floor

8. Drag and drop a material such as 'Gold' from the Select Window's material tab into the view window to assign a nice material to the new 3D text object. Add a point light source above the text. Use the Analytic Rectangle tool to model a floor below the text. Hit '0' key to render the view and check the result.

Tutorial project: 'tutorprojects\modeling\nurbs\fonts\hello'

Vectorizing Curves from Images

The Vectorize tool reads an image as an input, traces out contours from it and generates a set of NURBS curves which represent the traced contours.



The Vectorize tool

The generated NURBS curves can be used in various ways. For example, you can render them directly using the photo realistic rendering engine. Or, you can extrude the curves to get a 3D logo.

Let's imagine you have a photograph of a logo. You have scanned the photo to an image and now you want to create a 3D logo from it. Do this:

1. Activate the Vectorize tool. The control bar now shows you a set of tool specific options.

2. Use the Browse button in the control bar to locate the image to be vectorized.

Because the selected image is not an outline image, set also the Edge filtering option. If you have an image which already contains, say, black outlines on a white background, you do not need the Edge filtering option.

If the Fit to NURBS option is set, the tool runs a simplification procedure. It tries to represent the contour curves with a minimum number of control points. If this option is not set, the tool creates a point for each pixel.



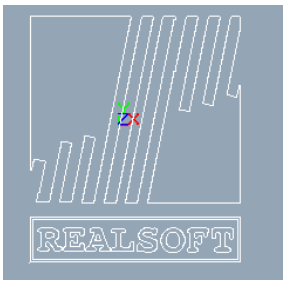
Specify options for the Vectorize tool

3. Click Accept. The tool creates a new level object, which contains the generated curves.

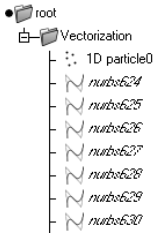


Note

Images can contain bright pixels which are not connected to any other pixels i.e. they form a point rather than a curve. The Vectorize tool uses 1D particles to represent these individual pixels.



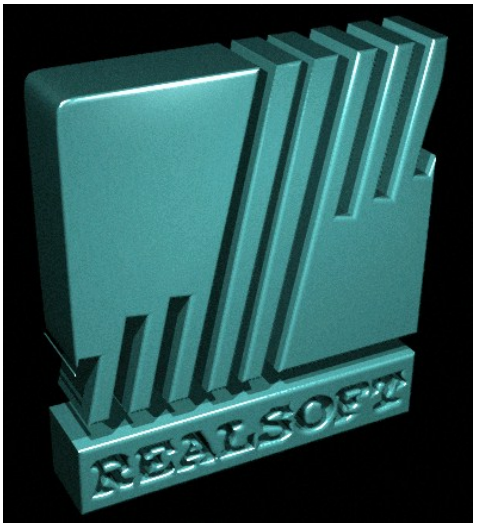
The image vectorized



The generated curves can then be rendered by resetting their Invisible in Photo Realistic Rendering option. Or, the curves can be extruded by using the Extrude tool.



Curves rendered with the Particlefur material

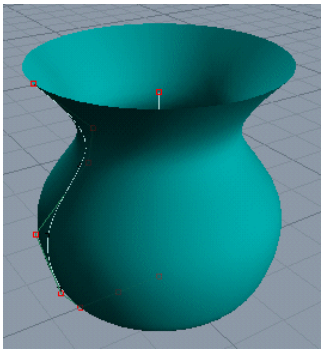


Extrude tool applied

Creating Curves from Surfaces

The Curves from Surfaces tool can be used for creating a set of NURBS curves from a surface with a specified accuracy. The curves can then be used for various purposes.

In this tutorial we will create a vase which consists of numerous fibers. The fibers are texture mapped NURBS curves.



Create a NURBS vase

1. Use the Rotate mesh construction tool to create a vase.

 *Curve extraction tool*

2. Select the vase and apply the Curve extraction tool from the NURBS tool tab. Use the control bar to specify suitable options.

Use the U Curves and V Curves options to specify whether curves should be evaluated in the U, V, or both the directions of the mesh.

The Curves field specifies the curve extraction density.

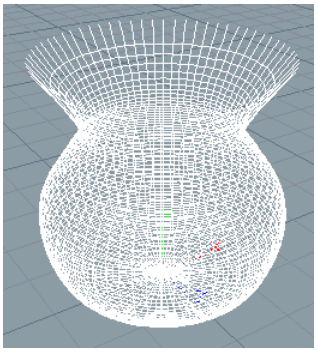


Note

This option doesn't define the total number of curves but the density. The value 1 gives the minimum number of curves. The higher the value, the higher the curve count.

The Points field specifies the number of control points per curve. Again, this is not the absolute point count but specifies the density for point insertion. The higher the value, the more points will be created to the curves.

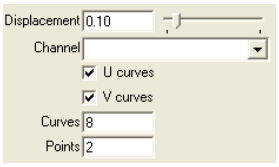
Check both U Curves and V Curves options. Set Curves to 8 and Points to 1. Then Accept the tool.



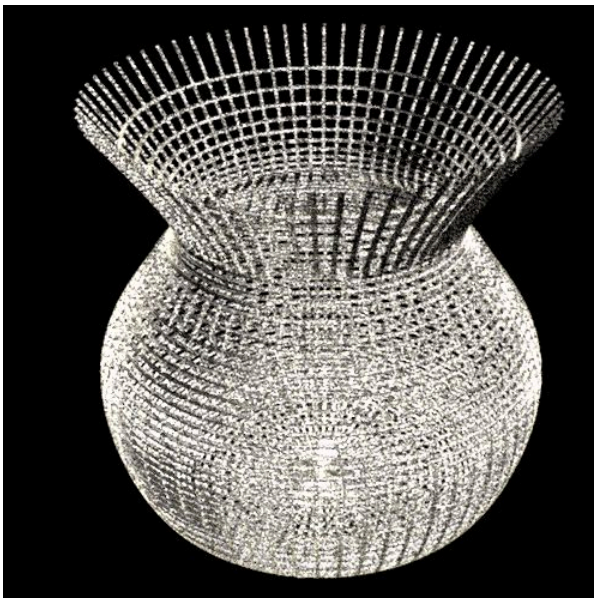
The vase converted to a set of NURBS curves

3. Select the root object and drag & drop the 'Particlefur' material into the view window. You can also use another material to define a desired color and other properties for the vase.

4. Select all curves (select the first curve, scroll down, press and hold the Ctrl key and select the last one), open the property window and go to the Gen tab. Uncheck the Invisible in photorealistic rendering option. Render the view.



Property window of the curve manager object



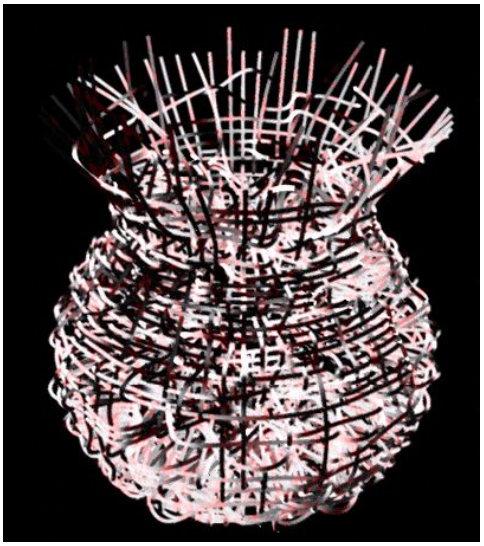
A basket created from texture mapped NURBS curves

If you need to modify the number of curves, select the Curve Manager level object from the select window and use the property window to specify a new curve count.

The Displacement attribute displaces the curves from the surface.

The Channel selector allows you to use any material to modify displacement values. To use this feature, map a material to the curve manager and specify a channel, such as color, which controls the density. The given channel is evaluated over the surface and values are used to scale the displacement defined by the Displacement option.

Tutorial project:
'tutorprojects\modeling\nurbs\surfacetocurves\vase'



Non-uniform displacement defined by the Spotty material

The Displace Tool

The Displace tool uses a bitmap image to define a translation for the control points of the selected NUBS meshes. You can use the tool to create real bumps to mesh surfaces (the material bump mapping modifies only the shading of the surfaces, not the geometry).

The following example shows how to use the tool.

1. Go to the NURBS tool tab. Activate the Rectangle tool.

2. Define a suitably high resolution for the mesh in the tool control bar. The displacement tool displaces the existing points of a mesh, so only a dense mesh can represent bumps defined by typical image resolutions. We will use a low-resolution image, so U=50 and V=50 should suffice for our test.

3. Finish the Rectangle tool by drawing a square mesh in a view window.

4. Make sure the mesh is selected. Go to the NURBS tool tab and activate the Displace tool.

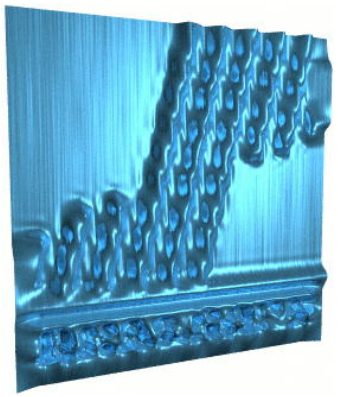


The Displace tool activated

5. Define a suitable maximum height for displacement using the Displacement field in the tool control bar, for example 0.1 meters.

6. Browse a suitable image file to the Image field, for example 'textures/logo_bw'. Now the program computes the displacement and you can see the result in the view window. Adjust the height if necessary. Then click Accept to finish the tool.

The result is shown here as a ray traced image. The 'brushsteel' material was slightly modified and mapped to the displaced surface to make it appear more interesting.



The displaced surface rendered as brushed steel



Chapter 5. Modeling Buildings

Creating Parametric Objects

The Packer object and its derivative Replicator provide tools for parametric object creation. Parametric objects can be rescaled and they keep the desired component dimensions. A kitchen cabinet is a good example of an object, which is frequently needed in various sizes. Rescaling a cabinet must not change the thickness of the construction material (wooden boards).

Modeling a parametric object is not a simple or automatic process. The shape must be divided into horizontal, vertical and depth layers. Behavior of each component must be defined. Inclusion of material mapping objects is wisest to plan in advance. These additional efforts in the modeling phase are rewarded later, because the reusability of the object is much greater than of models, whose dimensions are fixed.

As an example, we will create a complete window object shown in the attached image.

Tutorial level: Medium/Advanced

Example project: tutorprojects/modeling/packer/step1-step9

A window can be divided into depth layers in a natural way:

- Outer frame (mouldings outside the house)
- Panes and the surrounding panels
- Interior mouldings

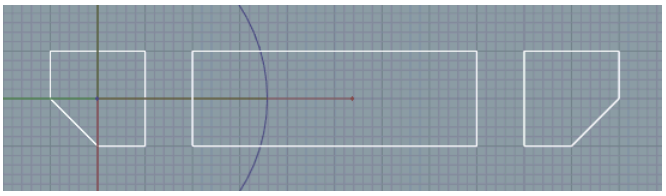


Outer Frame

Take a front view and enable parallel projection. Drag and drop a suitable grid, say 1 cm, from the grids tab of the select window into the view window. Zoom the view so that you see approximately one meter of space. We will model the window in its minimal size.

The topmost board of the outer frame of our example window has beveled edges. The beveling angle, 45 degrees, must not change when the window is resized. Therefore, we must split the board into 3 sections.

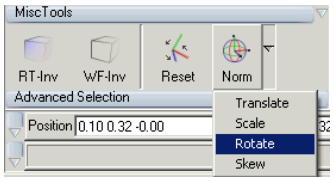
Activate the Analytic/Polyhedron tool from the tool bar. Draw the left end of the board as shown in the picture on the right. After the fifth point, click Accept to finish the tool. Change the depth of the board to 0.02 meters using the tool bar control.



Activate the Analytic/Cube tool and draw the middle section. Make it 30 cm wide and 10 cm high. When the cube is drawn, change its depth to 0.02 meters. Then duplicate the polyhedron at the left end and use the Transformation/Mirror tool to flip it horizontally. The gaps between the parts can be 0.05 meters at this stage. Do not make them much wider because they will contribute in the minimal width of the frame.

Let's rename the components so that it will be easier to manage them. Slowly double click the name of the first polyhedron in the select window and rename it as 'left'. Rename the cube as 'mid' and the second polyhedron as 'right'.

Multi select all three objects (selection order is not important). To make sure that geometry will be arranged in the correct direction, open the MiscTools bar below the select window, click the little triangle at the right side of the Norm tool and select Rotate from the list. The coordinate handles of all three objects become aligned with world's directions. Especially polyhedrons may have unusual handle directions because they include some diagonal lines. The packer tool, which we will apply next, will take its packing directions from the first selected object.



Make sure packed objects have proper handle directions

Having all three objects selected, go to the Measure tool tab and activate the Packer tool. The default options are suitable; Orientation=Detect means that the tool will examine how the selected objects are initially arranged and

pack them accordingly. In our case, the objects are placed side by side along the X axis, and the tool will select X directed packing accordingly. Hit the Accept button to finish the tool.



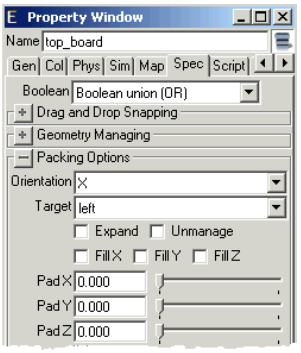
Note

Packing order is defined by the pivot points of selected objects. Pivot point is the center point of the transformation handles (3 RGB colored circles and axis lines), which become visible when the object is selected. The pivot point can be moved by dragging it with the mouse. You can guide the packer tool by arranging the pivot points in a suitable way.

A new item packer appears to the select window. The selected parts were placed to its sub hierarchy. In the view window, you will see a box-like container, which includes the components. Little arrows labeled with X and Y show the orientation of packing. Objects are packed along X direction as planned. Select the packer and rename it as 'top_board'.

Make sure the packer object is still selected and open the property window. Go to the Spec tab and open the Packing Options frame. The Target selector lists the three components managed by the packer. Select Target=left. Then clear the Expand option, if it is set. The packer will not expand the space reserved for the leftmost part proportionally to fill the packer space. The left part will occupy only the minimal space. Consequently, the left part is placed to the left end of the packer.

Pick the 'right' object from the Target list of the packer gadget and clear the Expand option if set. Then select 'mid' from the Target list. Enable Expand, as the middle part should indeed expand to fill the gap between the left and right ends. Furthermore, activate the Fill X Option, so that not only the container space of the middle part but also the actual geometry expands.



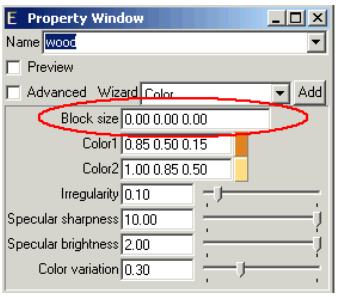
Packing options for the left-most part

The top board packing is now ready. The three parts should now be joined without gaps. The tool bar shows the width of the packer. Try changing it from the current value (should be about 0.6 meters) to say 1 meter. The middle part stretches to fill the increased dimension. Then set the width to 0.6 meters.

We have so far created a very simple object, but it already shows the essence of the packer object: ability to easily control stretching and placement of components in rescaling. The top board needs a couple of finishing touches before it is ready for use. First of all, let's hide the packer box to make the wireframe simpler and cleaner. In the property window, go to the Wire tab and activate the Hide Geometry option.

Next we will assign a material to the board. Unselect all objects by clicking empty area on select window. Take a side view. Switch to the material tab of the select window. Select wood from the library, and then activate Map/Parallel from the popup menu. Press the Shift key down and draw a symmetric cube in the view window. Shift key helps to keep all edges of the map cube equally long, which is important for the procedural wood material. Draw the cube about 0.05 meters wide and high. The second click finishes the parallel map tool.

Take a front view and hit the render button to make sure that the board is made of pine wood. You will probably see some unexpected seams in the wood pattern. This is because the procedural wood is configured for parquet floors and other similar wide surfaces made of several parts. To fix the problem, use the property window and change the Block size of the wood material to ' 0 0 0 '. The board will become like cut out of from a single trunk.





Continuous wood

Hint: Rotating the parallel map slightly from the top view may improve the look of the pine wood pattern, because the board is no longer cut perfectly along the wood grains. Real planks are cut that way because of the conical tapering of trunks.

Switch back to the leftmost Geometry tab of the select window. Drag the parallel mapping object into the hierarchy of the top_board packer. Select the parallel map and apply WF-inv tool, which is in the MiscTools bar below the select window. This hides the wireframe of the material map object. We do not want it to clutter the clean wireframe of the finished object. The board element is now ready.



The hierarchy of the finished top board

We will need three more boards for the lower part of the frame. This time we will not make any corner cuts or other decorations, so the modeling process is a just a simpler version of the steps above. Take the front view, activate the Analytic/Cube tool and draw a cube, which is 0.1 meters wide and 0.4 meters high. Remember to set its depth to 0.02 meters.

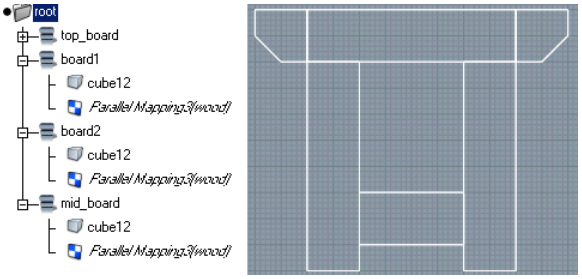
The new cube selected, activate the packer tool. Click Accept to finish the tool with the default options. Rename the new packer as 'board1'. Open the property window and enable Packing Options/Fill Y and Expand for the new cube. Set Wire tab's Hide Geometry option for the new packer.

Go to the select window and select the parallel map object under the top_board packer. Press Ctrl key down and drag the parallel map into the board1 packer. Ctrl dragging creates a copy of the object. Find and select the created copy from the hierarchy of the board1 packer. Apply MiscTools/WF-Inv button to make it visible again. Use the circular transformation handles to rotate the parallel map 90 degrees, so that wood grains match the direction of the board. Move the mapping on top of the new board. Make the map WF-invisible again.

Duplicate the board1 by Ctrl-dragging its red translation handle in the view window. Place the copy to the opposite side of the frame and rename it as 'board2'.

Hit Ctrl-d hotkey to duplicate board2. Press shift key down and rotate the new board 90 degrees using the circular rotate handle (Shift locks rotation to 45 degree steps). Rename the new horizontal board as mid_board and place it between the vertical boards.

The horizontal board is most likely too long to match the gap. Try entering the correct measure to the Height control in the tool bar (Height controls the length of the rotated board). The board does not become shorter. This is because the packer defines a minimal size, to prevent accidental overlapping of packed parts. To redefine the size, open the Geometry Managing frame in the property window and set the Min. Height value of mid_board to 0.2 meters. Then select the child cube of the mid_board packer and change its Height to 0.2 meters. Now the middle part matches the gap.

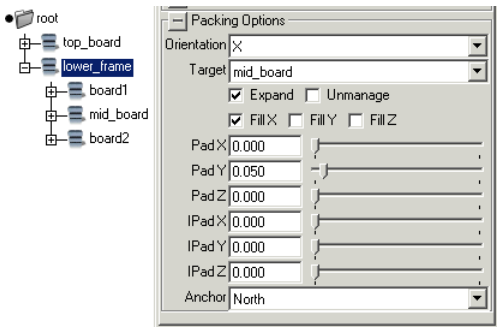


Three new boards added

Select board1, which has a proper orientation, then multi select board2 and mid_board, too. Activate the Packer tool and Accept it. Note how the new packer moves the middle board upward, because the default alignment option is Center.

Rename the new packer as lower_frame. In the property window's Spec tab, set Target=board1. The leftmost board should not expand in the packing direction. Therefore, make sure that the Expand option is not set. Set Fill Y option, because we want the board to fill the packer vertically. Then set Target=board2 and apply the same options: Expand off, Fill Y enabled.

Select mid_board from the Target gadget. The middle part will be the expanding component, so enable Expand. This part is supposed to stretch horizontally, so enable the Fill X option. Select Anchor=North from the bottom of the property window. The mid_board moves to the bottom of the packer. To add a suitable offset, set Pad Y = 0.05 meters.



Structure of the lower frame

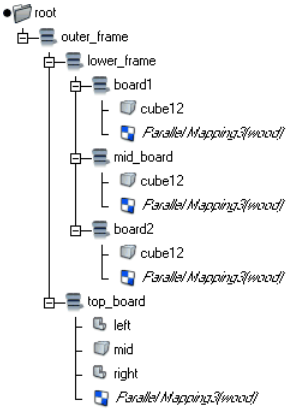
Try changing width and height of the lower_frame packer from 0.4 meters to say 0.6 meters, just to test that rescaling works as expected. Do not worry if lower frame expand over the top board; the next packer level will fix that. If scaling does not work as expected, recheck all packing options of sub objects and make sure they match the instructions above.

Restore the dimensions back to the original size 0.4 * 0.4 meters. Use the wire tab of the property window to hide the wire frame box of the packer.

The next step is to combine the lower frame and the board on the top. Make sure that the pivot points of these two objects are properly centered. In other words, pivots should be approximately on the same vertical line, so that the packer tool will understand that the parts should be packed vertically. Drag the pivots to the center points if necessary, or use the Center Pivot tool.

Select both objects, activate the Packer tool and click Accept. Rename the new packer as 'outer_frame'. Set the packing options of its child objects as follows:

- Set Target=top_board, clear Expand and enable Fill X
- Set Target=lower_frame, enable Expand, set Pad X=0.1 and then enable Fill X and Fill Y
- As usual, hide the geometry of the new packer



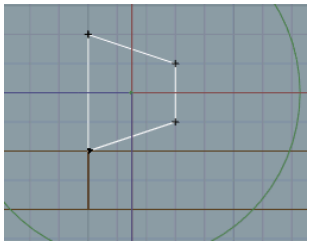


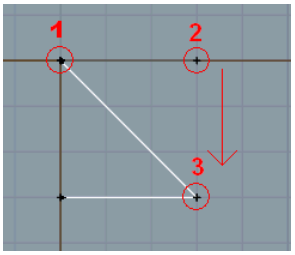
Outer frame of the window finished

The outer frame is now ready. Change the width and height dimensions to w=0.8 m, h=0.65 m. Check that the frame scales properly. Save the project to backup your work.

Window Panes

The frames of the window panes can be modeled easiest using NURBS modeling tools. Take a top view and zoom in to see the left edge of the exterior frame better. Activate the NURBS/Curve tool. On the second row of the toolbar, set Order=Polygonal and enable the Closed option. Draw the profile of the window frame for example as shown in the picture below. After the 4th point, click Accept. You can naturally define a more sophisticated profile, if you like. Our example profile is 3 cm wide and 4 cm deep.





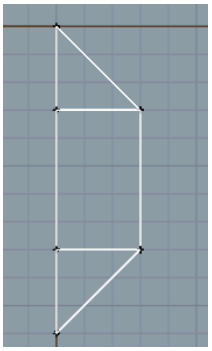
The upper profile curve is skewed 45 degrees

Take a front view. Move the created profile curve to the window corner. We will join frames using a diagonal beveling. Packer-managed assembly needs a corner piece, which is modeled as follows. Duplicate the created profile 2 times (Ctrl-d hotkey). Move the first copy 3 cm (=width of the profile) down from the window edge. Select the second copy, activate the Transformation/Skew tool, click on the left end of the profile and then on the right end. Move the mouse down until the profile is skewed 45 degrees. Click third time to finish the tool.

Select the two duplicates of the profile curve. The right end of the second toolbar row shows X-Sect tool. Activate it and click Accept. A new NURBS mesh appears to the select window. Rename it as 'corner'.

Duplicate the original profile again. Move the original profile below the corner piece, and the new copy about 5 cm below the original profile. Select both curves, and use the X-Sect tool to create a new surface. Rename it as 'mid_part'.

Duplicate the corner object. Use the tool Transformation/Mirror to flip the copy vertically. Place the mirrored copy exactly below the other two parts, as in the illustration.



Three NURBS meshes

Select all three NURBS meshes. The beveled corner pieces may have strange object space orientation. Go to the MiscTools bar, open the side menu of the Norm tool and select Rotate to fix this. Then activate the packer tool and click Accept. Rename the new packer as 'vertical_unit'. Set the Hide Geometry option in the property window's wire tab.

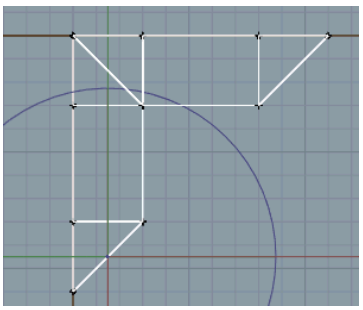
Select the new vertical_unit packer, open the property window and go to the Spec tab's Packing Options frame. The child objects should be packed as follows:

- Set Target=corner and clear the Expand option.
- Set Target=copy of corner and clear the Expand option
- Set Target=mid_part and enable the Fill Y option. Expand should also be enabled.

As usual, test the packer by increasing the height. Only the middle unit should stretch. Reset height back to the original value (0.11 m).

Deselect all objects, take a top view and switch to the material tab. Select wood material and start the Map/Parallel tool from the popup menu. Draw a 5 cm * 5cm box around the vertical_unit. Remember to hold Shift key down so that parallel map becomes symmetric in all dimensions. When the parallel map is drawn, switch to the geometry tab of the select window and drag and drop the parallel map object into the hierarchy of vertical_unit. Finally make the parallel map WF-Invisible from the MicsTools bar.

Go back to the front view. Duplicate vertical_unit, rotate the copy 90 degrees and move it so that the units form one half of the frame, as shown in the picture below. Rename the copy as 'horizontal_unit'.



Joining diagonally beveled pieces using a packer is a bit problematic. The control boxes of the two packers actually overlap in their current position. Normally, packer places the child objects side by side. We can pack the overlapping parts together permanently as follows.

Select the vertical_unit packer. Start the Packer tool and Accept. Rename the new packer as vertical1. Open the property window and select vertical_unit as Target and set the Packing Options as follows:

- Expand enabled
- Set Pad Y = -0.03 meters (=width of the profile)
- Enable the Fill Y option

Thanks to the negative Y padding, the corner pieces stick out from the control box of the packer. Hide the geometry of the new packer in the Wire tab of the property window. The height of the vertical1 increased when Y padding was changed, so move the pieces to match them seamlessly again. Then duplicate the vertical1 packer, rotate or mirror the copy 180 degrees and place it to the other side of the pane. Rename it as vertical2. Duplicate the horizontal_unit as well, rotate the copy 180 degrees and move it to the bottom to complete the frame. Rename the last item as horizontal_unit2.

Activate Analytic/Cube tool and draw a cube, which fills the hole inside the frame. Use the Depth control of the second row of the tool bar to make the cube thinner, for example to 0.005 meters (=5 mm). While the cube is selected, go to the material tab of the select window, drag the material glass and drop it into the view window. This assigns the glass to the cube. Switch back to the geometry tab of the select window. Take a side view and move the cube in the middle of the frame. Rename the level, which contains the cube and the glass material map as 'glass'.

Select both vertical pieces and the glass level. Apply MiscTools/Norm/Rotate to make sure that object spaces are properly aligned. Then start the Packer tool and click Accept. Rename the new packer as mid_frame. Hide its control box geometry in the Wire tab of the property window. The packing options of the children are:

- vertical1: Expand disabled, Fill Y enabled
- glass: Expand, Fill X and Fill Y enabled
- vertical2: Expand disabled, Fill Y enabled

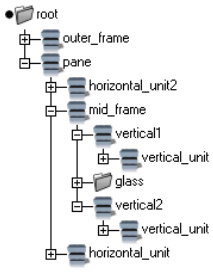


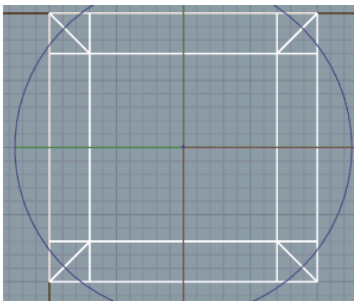
Note

If the frame profile includes a groove for the glass, wrap the glass object into a packer and apply negative X and Y padding to it. Then the glass extends out from the packer into the groove of the surrounding frame.

Finally, select horizontal_unit, mid_frame and horizontal_unit2. Apply MiscTools/Norm/Rotate. Create a new packer with the default options. The packing options are:

- horizontal_unit: Expand disabled, Fill X enabled
- mid_frame: Expand, Fill X and Fill Y enabled
- horizontal_unit2: Expand disabled, Fill X enabled





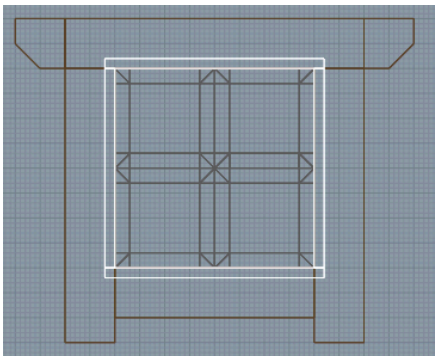
One window pane is now ready. Rename it as 'pane' and adjust its Width and Height to fill one quarter of the whole window (0.2 * 0.2 meters in our example). Set Hide Geometry for the pane packer to keep the wireframe clean and simple.

Duplicate the pane and move the copy to the right side of the first pane. Select both panes, and create a packer called top_row. Using the property window, enable Expand, Fill X and Fill Y for both child objects of the packer. Duplicate the top_row, move it below the top panes, then rename the copy as bottom_row. Select both rows and create a new packer. Rename it as 'panes'. Enable Expand, Fill X and Fill Y for its child objects (top and bottom rows). As before, hide the geometry of the created packers. Test the quad pane system by adjusting its width and height to, say, 0.6*0.8 meters. Then reset the size back to width=0.4 m, height=0.4 m.

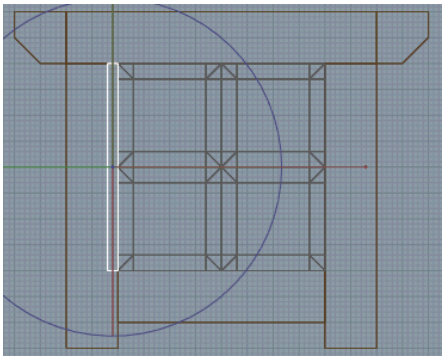
Window Edges

The next parts are simple to create. We will add 4 cubes (wooden boards) around the panes. The purpose of these cubes is to cover the edges of the window hole. Therefore, the board depth must be adjustable so that it can be matched with the thickness of the wall. In order to achieve desired z-direction packing, the board layer is split into two parts, as explained below.

The first part consisting of 4 packed cubes surrounds the panes, as shown in the picture below.



Take a front view and create a 2 cm wide cube. The height should match the height of the window panes (40 cm), and depth the depth of the pane profile (4 cm). Take a side view and move the cube so that it matches the side of the panes. The cube selected, activate the packer tool and accept it. Rename the packer as board_left. Deselect the board and switch to the top view. Go to the materials tab, make sure wood is still selected and activate the Parallel Map tool from the popup menu. Draw the mapping cube (5cm*5cm) around the board. Switch back to the geometry tab and drag and drop the parallel map into the new board_left packer. Make the parallel map WF-invisible. Select the board_left packer, open the property window, set Target=cubeX and set the Fill Y and Expand options.



First board added

Duplicate the board_left packer and move it to the opposite side of the panes. Rename it as board_right. Make two copies of the board_left, rotate them 90 degrees and move them above and below the panes. Rename the copies as board_top and board_bottom. Note that the top and bottom boards are not long enough yet, but the parent packers will later fix this.

Select board_left, panes and board_right. Activate the packer tool and accept. Set the packing options as follows:

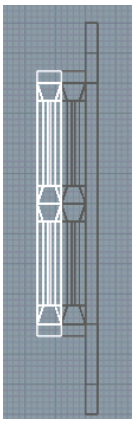
- Target=board_left: Expand=Off, Fill Y enabled
- Target=board_right: Expand=Off, Fill Y enabled
- Target=panes: Expand=On, Fill X and Fill Y enabled

Rename the new packer as mid_part.

Then select the board_top, mid_part and board_bottom packers. Apply MiscTools/Norm/Rotate to fix the object space orientation. Then create a new packer, which packs the three selected components. Rename it as wrapped_panes and set the packing options for its sub objects as follows:

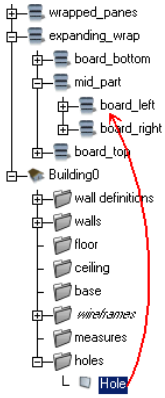
- Target=board_top: Expand=Off, Fill X enabled
- Target=board_bottom: Expand=Off, Fill X enabled
- Target=mid_part: Expand=On, Fill X and Fill Y enabled

Next we create the z-expandable layer of the wrapping boards. To make things simpler, just duplicate the wrapped_panes object and rename it as expanding_wrap. Move it behind the panes from the side view, so that it matches the other parts seamlessly.



wrapped_panes copied and moved

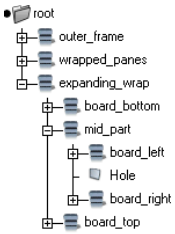
Open the hierarchy of the new duplicated packer expanding_wrap. Find the packer called panes and delete it. Removing sub objects of a packer usually changes the way how children are packed, so select the parent packer (called as mid_part) of the deleted panes and make sure that the children have correct packing options. Both boards should have Fill Y enabled, and Fill X and Expand disabled. Do not worry if the vertical boards move when you deleted the panes. We will next add a new component, which restores the shape.



If the window will be used with the building construction tools, it needs a hole cutting feature. Take a top view, activate the Building tool and create a straight wall behind the window. The direction should match the window. Click Accept to finish the building tool after the second point. Take a front view and activate the hole tool. Draw a hole, which matches the panes (40 cm * 40 cm). When the hole is added, explore the hierarchy of the building object to find the hole item (it is placed in a sub folder called holes). Drag and drop the hole to the hierarchy of the mid_part packer, between the board_left and board_right objects (that's the hierarchy position where panes were before we deleted it). You can also make the hole WF-invisible; it's wireframe does not show any information that is not already available from other parts of the window.

Select the new parent of the hole (packer mid_part), set Target=hole and enable Expand, Fill X and Fill Y. In other words, the hole fills the middle area of the expanding_wrap packer.

You can now delete the building object; it was just used for obtaining the hole. To save some work, you can naturally save a hole object, reload and scale it whenever you create new hole-cutting library objects.

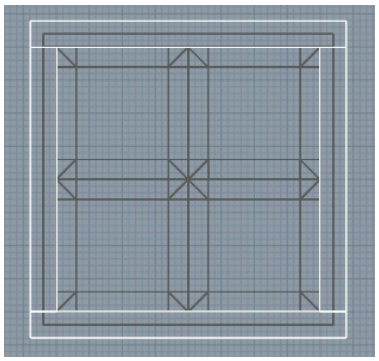


expanding_wrap packer hierarchy

Select the top level packer 'expanding_wrap'. On the property window, set Target=[All] and activate Fill Z option. Z filling is now used first time. After that, select each child packer from the select window and set the Fill Z option to all their sub objects. You should go trough all nested levels of packers and activate Fill Z for all children of all packers. Then try changing the depth of expanding_wrap packer from the current value 4 cm to say 10 cm. Make sure that all boards expand properly. If not, recheck that all Fill Z options are enabled. Finally, reset depth back to 4 cm.

Interior Mouldings

The last part of the window system is interior mouldings. First create four wooden boards, approximately 4 cm wide and 1 cm thick, which surround the window as in the example picture. Each board is a packer, which includes the geometry and the wood map. The geometry is packed using Fill X or Fill Y and Expand options. We do not repeat the creation steps here; they are identical with the steps for creating the edge wrapping boards above. In the example picture, the mouldings are joined in 90 degree angle. You can also use 45 degree beveled cuts between the mouldings, just like for the window panes. They are a bit more laborious to set up, though.



For the sake of clarity, rename the interior moulding items as moulding_left, moulding_right, moulding_top and moulding_bottom. Select moulding_left and moulding_right. Make sure the red object axis is horizontal when seen from the front view; if not, apply MiscTools/Norm/Rotate. Then create a new packer, which includes the two vertical mouldings. Rename the new packer as vertical_mouldings.

After the creation, the vertical mouldings jump slightly from their original positions. Open the property window and change the packing options as follows:

- Target=moulding_left: Expand and Fill Y enabled, Anchor=West
- Target=moulding_right: Expand and Fill Y enabled, Anchor=East

Now the mouldings return back to their correct positions.

Next select moulding_top, vertical_mouldings and moulding_bottom. Apply MiscTools/Norm/Rotate to fix the object spaces. Then create a new packer including these 3 components and rename it as mouldings. The packing options of the children are:

- Target=moulding_top: Expand off, Fill X on
- Target=moulding_bottom: Expand off, Fill X on
- Target=vertical_mouldings: Expand, Fill X and Fill Y enabled

Component Assembly

The window components are ready. We need to assemble them into a single packer. This phase requires some planning, because the components are not all symmetric. Simple centering is not enough - the outer frame must have certain offset values with respect to the other window.

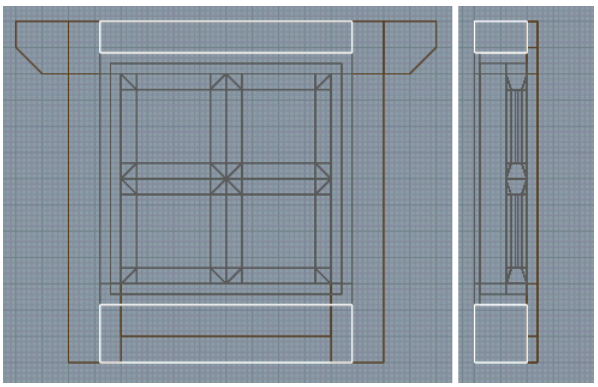
First we pack the panes, the edge boards and the interior mouldings. These are symmetric and can be managed in a simple manner. Take a front view and select the three above-mentioned objects one by one. Object handles should be centered in the middle of the window (exactly in the middle crossing of the 4 panes). If not, drag the handle center (pivot point) into the center of the window. Take a side view and ensure that each component has its handles inside the geometry. The packer tool uses the pivot points to detect the direction of packing. Move the components so that they fit together seamlessly. Select all 3 parts (wrapped_panes, expanding_wrap, mouldings) and create a new packer. Rename it as window.

Set the following packing options for the children of the new window packer:

- Target=mouldings: Expand off, Fill X and Fill Y on
- Target=wrapped_panes: Expand off, Pad X = Pad Y = 0.02 meters, Fill X and Fill Y on
- Target=expanding_wrap: Pad X = Pad Y = 0.02 meters, Expand on, Fill X, Fill Y, Fill Z enabled

Next we pack the outer frame and the window together. To solve the asymmetric vertical positioning problem, take a front view and create two cubes. The first cube (rename it as 'fill cube1') should fill the gap between the top edge of the interior mouldings and the top edge of the outer frame. The second cube ('fill cube2') fills the respective area below the window; see the attached picture. Take a side view and make sure the cubes are aligned with the window packer. Decrease the depth of cubes so that they match the depth of the window packer (9 cm).

Select both cubes and make them WF- and RT-invisible using the MiscTools tab. The only purpose of the cubes is to reserve some space in packing.



Two highlighted 'place holder' cubes define asymmetric packing offsets from edges

Select the cubes and the window packer. Create a new packer and rename it as window. Set the packing options:

- Target = fill cube1: Expand off, Fill X and Fill Z may be enabled (filling is not relevant for invisible cubes)
- Target = fill cube2: Expand off, Fill X and Fill Z on
- Target = window: Expand on, Fill X, Fill Y, Fill Z on

The final packing in depth direction needs some special attention. So far, we have created packers using the automatic Detect option. The packer tool has figured out the packing arrangement and orientation itself. Next, we will guide the highest packing orientation manually, because we want to define, which point is the fixed anchor when dimensions change. For a door, the anchor point should be in the bottom front edge of the door. Then the door

elevation will not change when the door height changes. For a window, the anchor point should probably be at the top, so that top edges of several windows remain aligned regardless of applied height changes.

Take a front view. Select the outer_frame first, then multi select the window. This time the selection order is important! Start the packer tool. Do not click Accept right away. Instead, change Orientation control in the toolbar to Z (we did not bother lining up the pivot points above). Click first time on the top left corner of the whole window object. Then move the mouse down and to the right, until the packer box matches exactly the width and height of the window. Click second time to finish the packer tool. The new packer has its anchor set to the top left corner of the view, as the XY arrows indicate. Hide the control box of the packer from the Wire tab's Hide Geometry option of the property window.

Rename the new packer as window; this will be the final window object. Pack the children objects as follows:

- Target = outer_frame: Expand off, Fill X and Fill Y on
- Target = window: X Pad = 0.16 m, Expand on, Fill X, Fill Y and Fill Z on

The parametric structure of the window is now ready.

Drag and Drop Settings

The final step of the tutorial is to set convenient properties for the window object considering its later use in house construction projects.

Select the complete window object (highest packer level). Make sure that its sub hierarchy folder is closed in the select window. This ensures that you can select the complete window object simply by clicking its wireframe on a view window. If the sub hierarchy is opened, view selections would activate sub objects, which is inconvenient considering possible transformations to new positions.

Open the property window. Go to its leftmost Gen tab. Set Hit-Invisible option for the window. This will make snap operations in complex models - with tens or even hundreds of windows - faster.



Note

Objects, which snap together (for example kitchen cabinets) should not be totally Hit invisible. However, their internal parts, such as shelves and minor details (handles, decorations) can be made Hit invisible to improve performance.

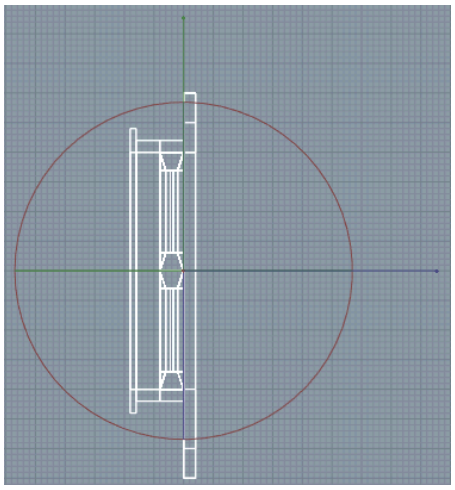
Switch to the Spec tab in the property window. Open the Drag and Drop Snapping frame. Here you can define what happens when the window object is dropped from the object browser to a view. The following options are suitable, when the window object handles are aligned with the world space. You can check the alignment by taking a front view while the window is selected. Red axis should point to the right, green axis up and blue axis towards the viewer. If the space is not like this, apply the MiscTools/Norm/Rotate to fix it.

- Space = Object. Snap targets are searched in directions defined by the window geometry.
- Dir1 = XZ. Window snaps to the nearest wall in XZ direction (horizontally). It does not snap to roof or floor.
- Dir2, Dir3 = None. Secondary snap directions can be inactive. These options are usually activated for corner searching. Our general purpose window object is not corner specific.
- Rotate=H,P. Object is first rotated around its y axis (Heading angle) to match the snap surface. Then it is rotated by its x axis (Pitching angle). Walls are usually vertical, so pure H (Heading) rotation would be appropriate for most cases, too.
- Axis=Z. The positive Z axis of the object will be rotated to match the surface normal of the wall. In our window, positive Z axis corresponds to the outer side of the window. Therefore, the outer window frame is rotated to the drop side of the wall. With these settings, the window should always be dropped outside the wall.

- Fit=Pivot. The object is moved so that its pivot point matches the snap position.
- Range=2 meters. This is a suitable finite snapping range. You can temporarily disable snapping by dropping objects farther away from the walls than this limit. On the other hand, it is easy to drop the window quickly within this snap range from a desired target wall.

Take a side view and make sure that pivot point is placed between the outer frame and the rest of the window - see the picture below. If not, drag the pivot point there. From the front view, drag the pivot point to the center of the window. Move the window object up/down so that the window is placed vertically to a suitable default height. For example, the pivot point can be 1.5 meters above the ground level. The elevation of windows in each building project will naturally depend on the wall and base heights, but this kind of default elevation ensures that wall-window snapping will work. Windows, which have been quickly drop-snapped to walls, can later be moved up or down.

Set the default size of the window using the dimension controls (Width/Height/Depth) of the toolbar. The window was modeled above to a minimal size w=0.8m h=0.65m d=0.11 m. A more common size might be 1.2*1.2*0.23 meters.



Place the pivot behind the outer frame



Note

The dimensions include the mouldings. If you want to control the size using the actual hole dimensions, drop the window into a new packer level and add some negative X and Y padding, which compensates the mouldings.

We can also optimize the real-time drawing of the window. When shaded OpenGL mode is enabled, the program computes up to several hundreds of pixels wide preview texture for each component of the window. The window object we created is actually not that simple; it includes almost 100 sub objects! Accurate preview computation will take long time when several tens of windows have been added to the scene.

We can conveniently change all objects using the powerful advanced selection tool. The tool is available below the select window; click the toolbar to open it. First clear all selection by hitting the Clear button. Switch to the Class tab of the Advanced Selection window. It shows all object types included in the current scene. Find Polyhedron from the tree and click it to activate the class. Then click the Select button to select all polyhedrons of the window. Next activate Rectangle from the class tree and click Select. Then activate Cube class and click Select. Finally activate Nurbs Mesh class and click Select. Now all geometric surface items included in the window object have been selected. Click the Advanced Selector bar to shrink the window back to its original size.

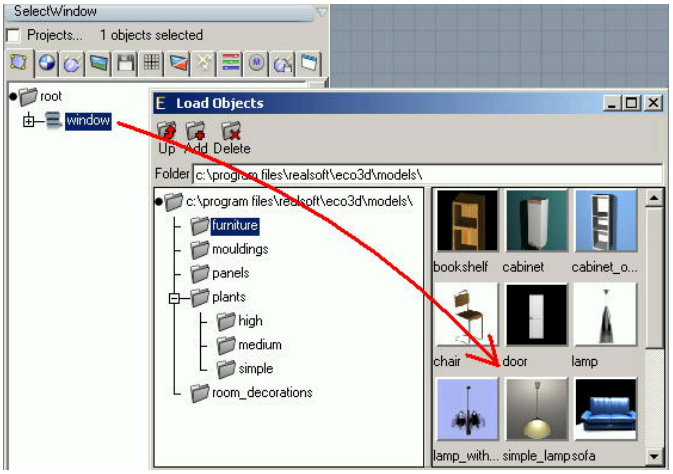
Go to the property window's Wire tab. Change the texture Quality to a low value, for example 8. You can also reduce Shaded Quality to 1 or 2 to achieve even faster OpenGL drawing.

Saving with a Thumbnail and Materials

Let's create a nice thumbnail image for the window. Take a side view and zoom out. Click the root level and create a point light source above and in front of the window. Navigate the view to find a nice viewing position, which shows the shape of the window well (maybe a direct front view is OK). Zoom in so that window fills most of the view. Hit zero hotkey '0' to render the view. Open File/Save As window and click the Thumbnail image square to record the image from the view. You can also fill in the author and the comment information.

Close the Save Dialog without saving. Instead, open the File Browser window. Select a suitable target folder for saving in the browser window. Then drag the window object from the select window and drop it to the browser's object thumbnail area at the right side of the window. Drag and drop to the browser window saves the dropped object with the current project thumbnail and all materials that were used in the object. It is a quick way to save all relevant object information.

You can now find the created object from the browser window and start using it in your house construction projects.

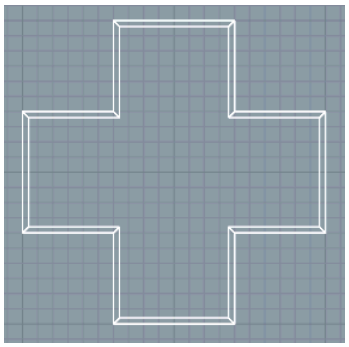


Modeling Roofs

Multiple Roof Lines

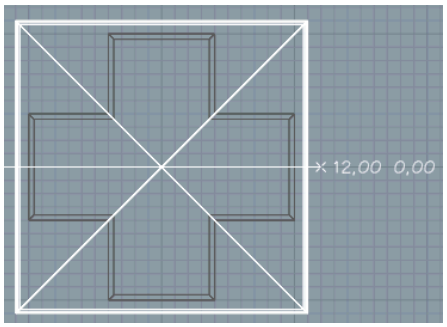
The roof object can be controlled using several roof lines. To try this, take a top view, activate the building tool and draw the walls as shown in the picture.

After drawing the walls, hit the roof tool button. It appears to the second row of the toolbar automatically when a building object is selected. The roof tool adds a default roof to the building. In this case, the default roof is a pyramid-like symmetric hipped roof.



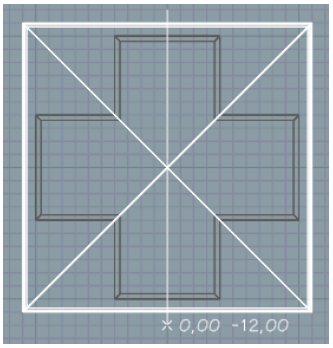
Draw cross-like walls

The roof becomes automatically selected, and its context tool Roof Line appears to the tool bar. Click the Roof Line tool. From the top view, draw a straight line across the building as shown in the picture. Click Accept or hit the Enter key after clicking the second point.



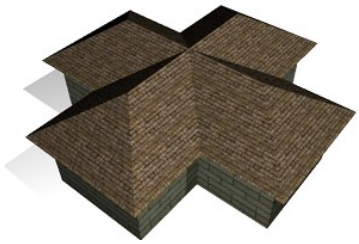
The first roof line

Note that the first roof line does not change the roof at all. It is still a pyramid like hipped roof. Activate the roof line tool again and draw a second line, which is perpendicular to the first roof line, across the building.



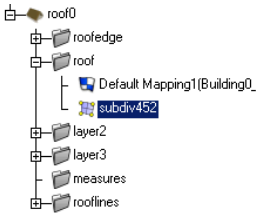
The second roof line

The two guide lines together define a cross-like roof.



Editing the Roof Manually

Automatic roof builder provides a quick way to add the roof. However, it is quite limited. The automatic roof object can be used as a starting point for modeling unusual roof shapes.

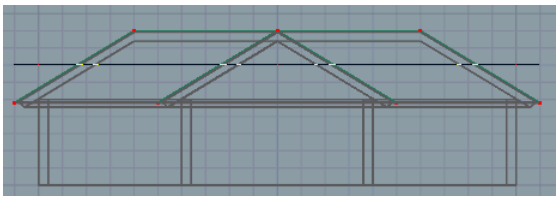


The roof surface selected

Select the roof object created in the example above. Open the property window (using 'p' hotkey or from the popup menu). Deactivate the Build option. Automatic roof construction is now disabled, and you can start editing the shape.

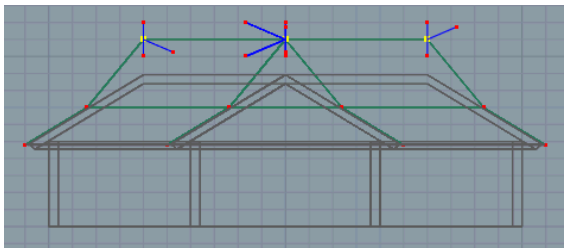
Select the object called 'subdivXX' in the sub folder called 'roof' of the roof creator object.

The second toolbar row shows an Edit mode control; set it to the Point mode. Then scroll the second row of the toolbar until you find the Knife tool. Click the tool to activate it. Take a front view and draw a horizontal line across the roof as in the picture below. Hit the Enter key after clicking the second point.



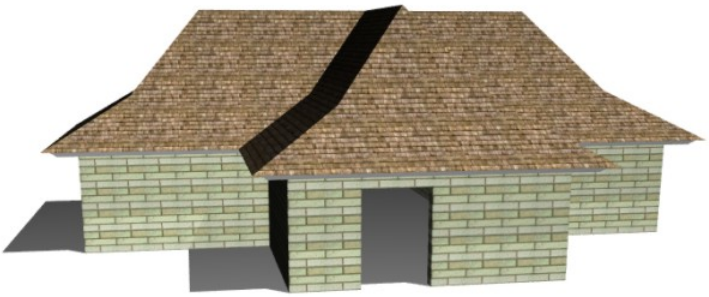
Cutting the roof top with the knife tool

Select the topmost points of the roof by LMB-dragging a selection box around the roof top. Move the pointer exactly on top of a selected point handle, press the left mouse button down and drag the points upwards about one meter. Be careful not to drag the blue handle lines - it will break the roof!



Move the roof top upwards

The modified roof is now ready. To improve rendering quality, open the UV editor from the Windows pull down menu and move the selected points further away from other points to compensate texture stretching, which appeared when the roof surface was stretched upwards.



Oriental roof style



Note

Enabling the roof's Build option removes all manually edited changes from the roof.

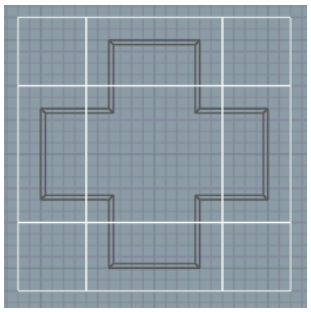
Custom Roof Shapes

The whole roof surface can be modeled manually. Placing the custom roof surface into the hierarchy of the roof builder object allows some automatic parametric features such as wall fitting with the roof.

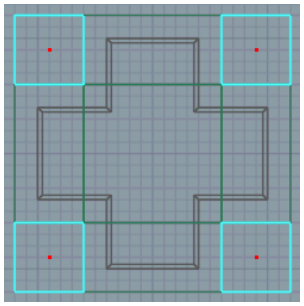
We will continue from the tutorial above. If you experiment with another building project, just make sure that Build option of the roof object is turned off. Select the sub folders roof, layer2 and layer3. Hit the del key to delete them. This removes all roof surfaces.

Take a top view. Activate the SDS/Rectangle tool from the toolbar. Draw a rectangle that is about 1 meter wider than the walls. From the front view, move the rectangle above the building.

Go back to the top view. Apply the SDS Knife tool 4 times to cut new sections through the SDS rectangle, as shown in the example picture.



SDS rectangle forms the roof



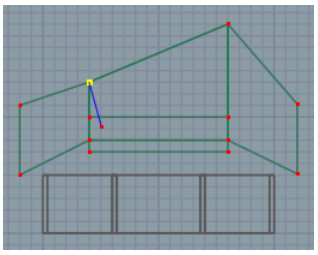
Corner faces selected

Set the Edit mode selector in the toolbar to Face. Click a corner face and then shift click the three other corner faces to multi select them.

Hit the Destroy tool button in the toolbar to remove the corner faces.

Set the edit mode selector to Point. Select the SDS points one by one from the top view, and apply the move tool (m hotkey) from the side and front views to move the points up and down, so that the SDS surface gets an irregular bumpy shape. Note that holding the y modifier key down while moving keeps the roof points on their original vertical axis.

Set the Edit mode back to Face. Apply the Dupl tool (Duplicate Faces). Hit m hotkey, press y modifier down and move the duplicated faces slightly upwards. The roof gets some thickness.



Roof points edited vertically

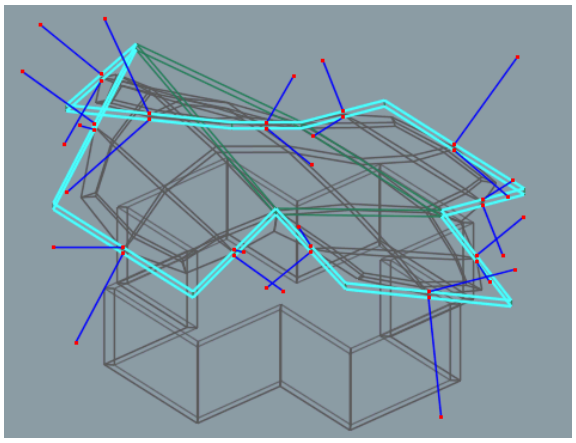
The top layer of the roof is now suitably selected. Take a top view and zoom in heavily so that you see only a small part of the roof (one texture tile). Switch to the materials tab and drag and drop the material buildingX_roof to the

view window. This assigns it to the selected faces. Open the property window and select a suitable texture map for the buildingX_roof material.

Switch back to the geometry tab of the select window. Go to the spec tab f the property window and change the Rendering/Type of the SDS object to Smoothen to NURBS. The roof gets an organic rounded shape.

Set the SDS Edit mode to Edge. Multi select outermost edges of the top and bottom surfaces of the roof - see the picture below. You can do this easily from the top view. Just keep the Shift key down and drag small selection boxes which hit the desired edges. Do not select the short vertical edges at roof corners.

Set the rounding selector at the right end of the second row of the toolbar from Round to Sharp. You can make the edges sharp also using the controls available in the Spec tab of the SDS property window.



The roof edge loops selected

The roof is now ready. Set the Edit mode to Object and drag and drop the SDS object from the root level into the hierarchy of the roof object. It can be placed anywhere in the roof's sub hierarchy.

Select the Wall0 object in the wall definitions folder of the Building object. Activate Extend to Roof option from the toolbar. Walls expand up to the roof. To ensure accurate match, change the fitting density of the wall object to 0.1 meters from the toolbar.



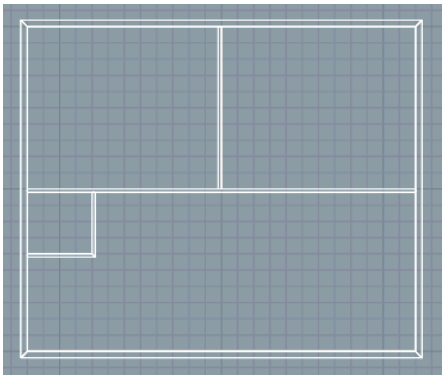
Gaudian roof style?

Multiple Floors

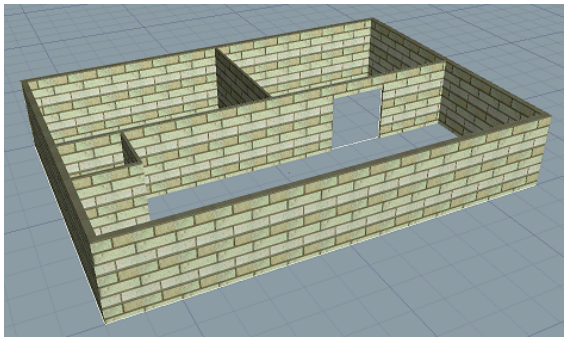
This tutorial shows how to create buildings, which have multiple floors.

Ground Floor

Start the building tool, take a top view and draw a building shown in the example picture. The dimensions of the rectangular building are 12*8 meters. When the outer wall is done, split the interior first with a horizontal sub wall, and then using a vertical sub wall. The third sub wall separates a 2*2 meter corner from the largest interior space. Cut a hole between the room at the top left corner and the largest room using the Hole tool. Rename the building object as 'ground floor'.



The walls of the ground floor



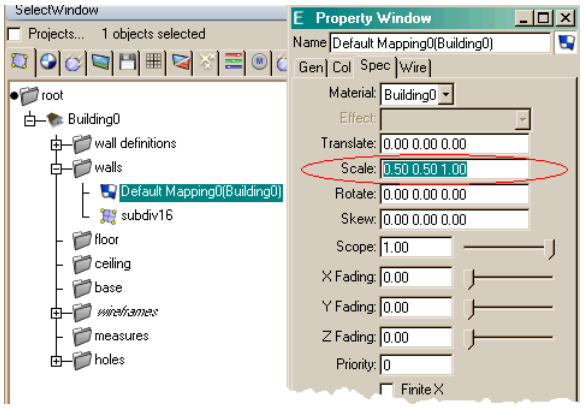
Too large tiles

Go to the second materials tab of the select window. Select the material building0 and open the property window. Select textures/tiles to the FileName field using the file browser.

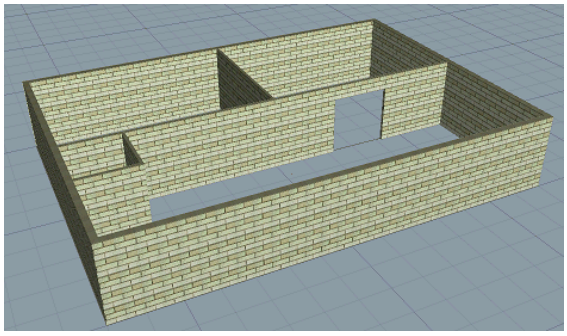
Go to the rightmost view models tab of the select window. Drop OpenGL shaded to the view window to see the building in shaded mode. The brick pattern is too large, because the selected texture includes only a few tiles.

The size is easy to fix. Go to the geometry tab of the select window. Select the item ground floor/walls/default mapping(Building0). Open the property window and go to the spec tab. Change the Scale value from (1 1 1) to (0.5 0.5 1).

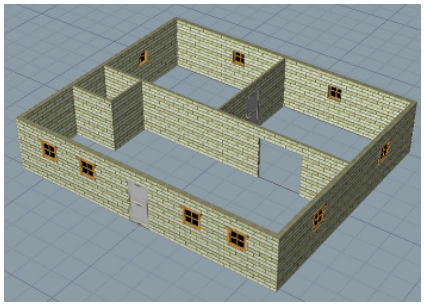
The tile size is reduced to 50 % from the original size, and now it is quite realistic. Close the property window.



Adjusting the texture mapping scale



Tile size fixed

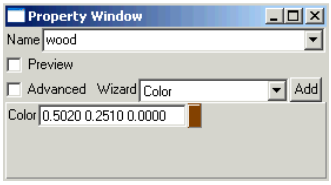


Doors and windows added for the ground floor

Select the ground floor (=the building object) and make it the current level using the Make Current popup menu of the select window. Open the Browse Object Library window from the popup menu. Drag and drop doors and windows for the ground floor. Restore the current level back to the root level.

Add a light source to the scene, navigate the view to see the building from a suitable angle and hit the zero key to render the view. Pinewood windows may not look that good in a brick wall. Go to the second tab of the select window to examine the material library. Select the material called wood. Then hit the del key to delete it.

Select 'New/VSL Material' from the popup menu. Open the property window and rename the material as wood. Set the Wizard selector of the property window to Color and click the Add button. A new color control appears to the property window; change it to a dark brown tone (0.5 0.25 0). The new kind of painted brown wood is now applied to all windows.

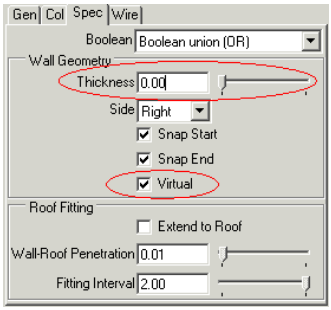


Pine wood replaced with brown painted wood

Select again the complete building (ground floor). From the top view, start the room tool. In the toolbar, define the name 'livingroom' and then click in the middle of the largest room. When the room decoration appears, turn Ceiling temporarily off to see into the room. Go to the materias tab and define the texture file names for livingroom_walls and livingroom_floor, for example 'wood' and 'plates'.

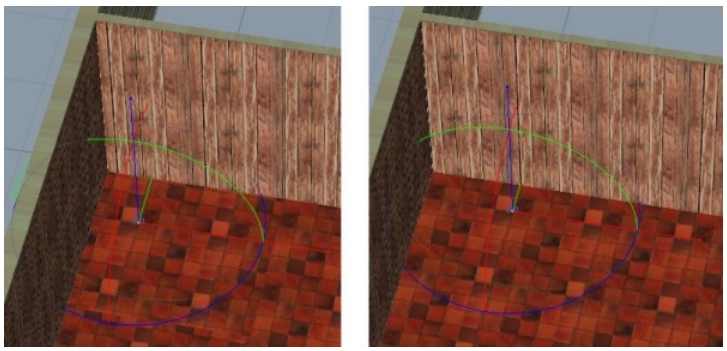
Select the livingroom object. Hit Ctrl-D to duplicate it. In the view window, move its position indicator into the small corner room. The corner room gets the same material attributes. Rename the copy of the room object as 'staircase'.

In an ordinary house, the staircase is not enclosed into a room. To remove the wall, select the last sub wall object from the wall definitions folder of the ground floor building. Open the property window, and enable the Virtual option. Then set the Thickness of the wall to zero.

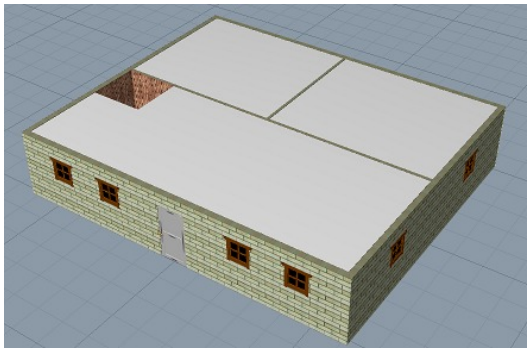


Properties of a virtual wall

The room tool sets the texture pattern relative to the position of the room indicator. Take a top view and make sure shaded OpenGL drawing is enabled. At this point, the texture pattern of the floor may not run continuously from the staircase area to the living room. If you see a visible seam, select the staircase room object. Then turn the grid snapping off and activate the move tool. Move the room indicator to find a position, which makes the pattern to run seamlessly. Turn grid snapping back on.



Move the room indicator to fix the texture seam



The ground floor finished

The corner room has apparently disappeared. However, it still divides the space. We can, for example, turn off the ceiling option from the staircase to cut a hole from the ground floor to the upper floor. Such a hole is a parametric feature and its shape and size can be edited at any stage of the building construction.

Apply the room tool two times more, to texture the remaining rooms. The rooms can be called, for example, as bedroom and kitchen. After texturing all rooms, enable the ceiling option for all rooms except the staircase. The ground floor is now ready.

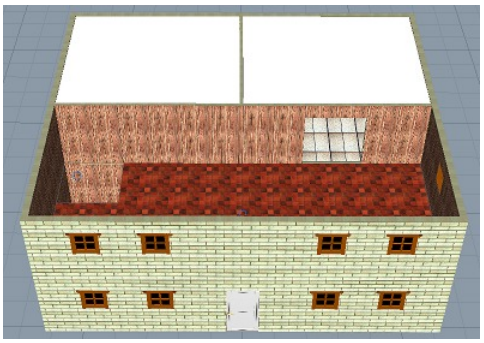
Upper Floor

Select Edit/Duplicate from the pull down menu. Rename the duplicate as 'upper floor'. Move the upper floor upwards until it is placed exactly on top of the ground floor.

The upper floor obviously needs some editing. First of all, find the door object, which cuts the outer wall, and delete it from the hierarchy of the upper floor. You may also rearrange the rooms if you like, or add or delete some windows, or change some textures. Do not change or remove the staircase object.

Select the staircase object of the upper floor and open the property window. Staircase downstairs had ceiling disabled and floor enabled. To finish the hole between the two floors, disable the floor option for the upper staircase room. You can now see all the way downstairs through the hole.

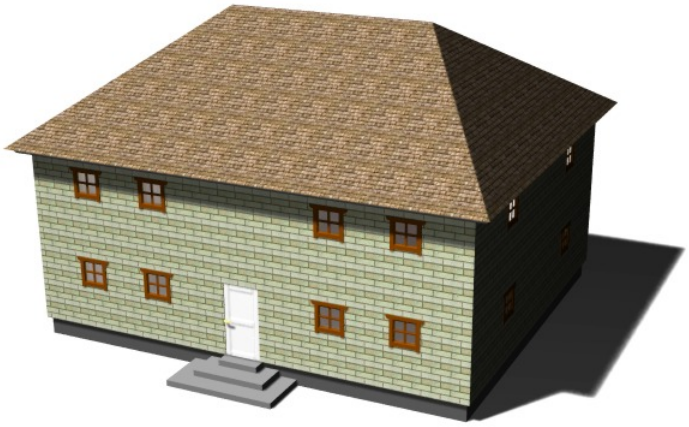
If you zoom in and look closely in shaded OpenGL mode, you can see a narrow gap between the wall textures of the upper and lower staircase rooms. To fix it, select the staircase object in the upper floor and using the property window, set its floor offset to zero. Then select the staircase object under the hierarchy of the ground floor and set its ceiling offset to zero. The gap disappears.



A hole between two floors

The actual stairs can be modeled using the general modeling toolset of the software. With the macro feature, it is easy to build repeating forms. The steps can be cut to match the staircase area using the Boolean tools. For details, see other modeling related tutorials of this manual.

To complete the building, enable the ceiling option for all rooms of the upper floor. Select the upper floor object and apply the roof tool to it. Set desired roof options using the property window. Go to the material tab of the select window, select the upperfloor_roof material and set a suitable roof texture for it using the property window. Select the ground floor. Set its Base type to Fitted to Walls using the selector in the tool bar. Model the stairs at the front door for example using analytic cubes.



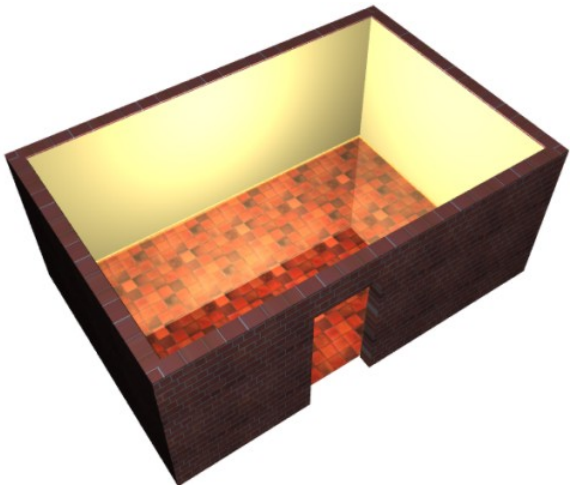
The finished building

Designing Kitchens

The packer object is a very useful tool for managing furniture groups. The next tutorial about kitchen design is a good example.

Take a top view. Start the Building tool and draw a simple rectangular building. Then start the room tool. Name the room as kitchen and click in the middle of the building to add the room. Switch to the materials tab and assign suitable texture maps to the kitchen floor, wall and the ceiling materials using the property window. Create a point light in the middle of the kitchen, 0.3 meters below the ceiling.

Go to the grids tab of the select window. Drag and drop a 0.1-meter grid to the view window. It is a suitable grid for kitchen design.



Empty kitchen

Switch back to the geometry tab of the select window. Take a top view. Open the Browse Object Library window from the popup menu of the select window. Drag and drop the object 'Cabinet' from the furniture folder into the view window, close to the top left corner of the back wall. Drop the cabinet closer to the back wall than the side wall, so that it snaps primarily to the back wall. If the drop position was properly chosen, the cabinet ends up in the corner as in the example picture.



Drop a cabinet to the corner

Click the Packer tool of the Creation tab of the toolbar. Click Accept to finish the tool with the default options. The new packer takes its orientation from the selected cabinet. Rename the packer as 'wallcabinets'.

While the packer is selected, hit the Tab key or apply Make Current from the popup menu of the select window. Then new cabinets will be inserted automatically to the packer.

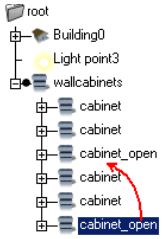
Now you do not need to aim at all when inserting new cabinets. Just double click the item Cabinet again in the object browser window. The new cabinet goes automatically to the end of the row.

The loaded cabinet is automatically selected. Change its width to 0.6 meters using the tool bar controls.

Double click the icon 'Cabinet_open' on the object browser window. Change the width of the new cabinet to 0.8 meters.

Double click the Cabinet icon 2 times more, to add two new cabinets to the row.

Select the second cabinet. Change its width back to 0.4 meters. Cabinets move to fill the gap.



Changing the order of cabinets

Double click the 'Cabinet_open' icon to load it. As expected, the new cabinet goes to the end of the row. In the select window, drag the last cabinet in the list (=the new Cabinet_open) and drop it just below the other open cabinet in hierarchy. The open cabinets will be placed side by side.

Select the new open cabinet. Hit del key to delete it. Cabinets move together to fill the gap.



Wall cabinet row finished



Chapter 6. Materials

Introduction

When creating photorealistic images and animations, it is necessary not only to create complex shapes, but also to simulate different real world material properties. Realsoft 3D includes powerful tools for this purpose.

Mapping materials to objects using drag and drop methods

The fastest and easiest way to assign a material to a geometric object is to drag and drop. Select the target object and drag a material from the material library, and drop it on the view window. The next example demonstrates this straightforward technique.

1. Select Sphere from the Analytic toolbar and create a sphere by clicking twice on the view window.

2. The sphere tool leaves the newly created object selected. If you were assigning a material to a previously created object, you should select it now. The selected object will be the target of the dropped material.

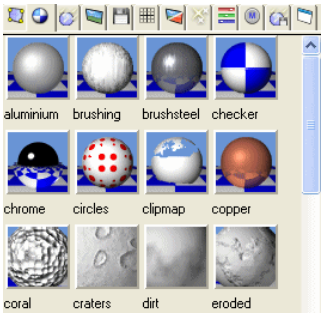


Sphere selected

3. Go to the second tab of the select window, called Materials. This will display the contents of the current material library.

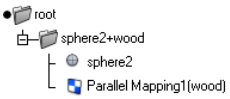
4. Move the mouse on top of the material 'wood', press the left mouse button down, then while holding the button down, move the mouse over the view window. Release the left mouse button there. In other words, drag&drop the wood material onto the view window.

That's it. Note that the position on the view window where you release the left mouse button does not matter. It does not select the target object, because this would be too inaccurate.



The material tab of the select window

Switch back to the object hierarchy section of the select window. It should now show you an object named 'Sphere+wood'. Open the sub hierarchy of that level by clicking the small '+' box at the left side of the level 'Sphere+wood'. The level object consists of two sub objects: the original sphere and wood. In other words, you have a wooden sphere. Realsoft 3D combines shapes and materials using the powerful and intuitive concept of hierarchy. Thanks to this, you can add new parts to the geometry later and they automatically get the right kind of (hierarchically defined) material properties.



The structure of a wooden sphere in the select window



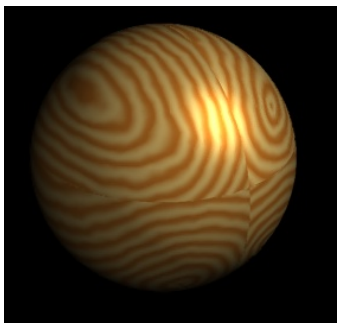
View control bar

The new material link object 'Parallel mapping(wood)' is called the *material mapping object*. The material to which the mapping refers affects all the objects in the same hierarchy level with the mapping object.

Let's render the sphere to see if the material works:

5. Use the Point light tool of the Light Sources section of the toolbar to insert a light source above the sphere.

6. Click the Render button in the View Control Bar to render the sphere (or use Render/Ray trace from the view's popup menu).



The wooden sphere rendered

Mapping multiple materials to a single object

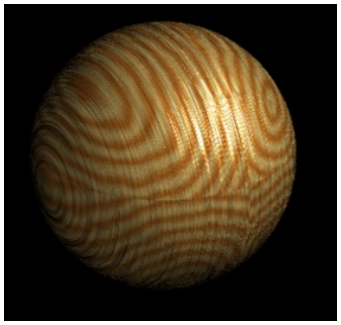
Attaching any number of materials to a single object is just as easy. For example, to turn the wooden sphere to a shiny grooved wooden sphere:

1. Make sure the 'sphere+wood' level is still selected in the select window's geometry tab. Note that the previous material dropping automatically leaves the hierarchy to the correct state.

2. Drag & drop the 'plastic' material into the view window.

3. Drag & drop the 'brushing' material into the view window.

4. Render the view window.



Grooved shiny wood

The plastic property makes the surface shiny, and brushing adds small irregular grooves to it.

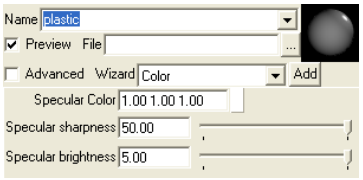
Modifying material properties

Most materials of Realsoft 3D, including wood, plastic and brushing, are not fixed but allow various adjustments. Let's change the shininess of the modelled sphere.

1. Go to the Materials tab of the select window. Click the material plastic with the left mouse button to select it.

2. Select Properties from the select window's popup menu. You can also combine steps 1 and 2 by double clicking the 'plastic' material icon.

3. The property window opens showing the material's properties. There are only two sliders!



Properties of plastic material

4. Decrease the Specular brightness value from 5 to 2 and Specular sharpness from 50 to 10. Close the Property window.

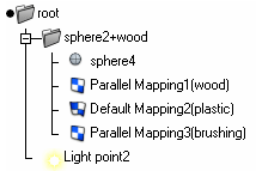
5. Render the view window. The shiny area is now less bright but wider, because the sharpness was reduced. The material combination looks more like varnished wood now.

One might think that the material editor of Realsoft 3D is poor, because there are only two sliders available! The truth is quite the opposite. What we see here is a tailored interface for the plastic material, which is so simple that anybody can change the material right away. Instead of displaying hundreds of confusing buttons and sliders, the program only shows the characteristic properties of each material. Later we will see that there are other, more advanced and powerful levels of control available, which can be adopted step by step as you become familiar with the program.

Modifying mapping objects

The hierarchy of the model is now like this:

As we have demonstrated in this tutorial, Realsoft 3D uses geometric mapping objects in the scene hierarchy to assign materials. One great advantage of this approach is that modifying a material's geometric properties such as position and density becomes very easy and intuitive. Also operations such as duplicating, cutting and pasting materials or combinations of materials to affect new objects can be done efficiently.



Many materials assigned to a sphere

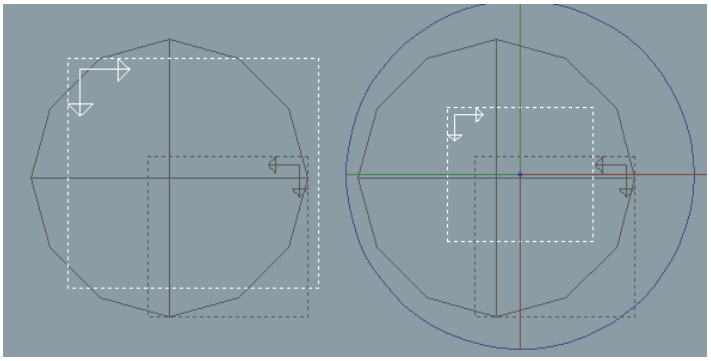
For example, let's try to make the grooving of the wooden surface denser:

1. Select the object 'Parallel Mapping(brushing)' from the Select window.

2. Go to the Transformation tab of the toolbar and activate the Scale tool. This tool is for resizing objects.

3. Press the Shift key (Shift modified produces uniform scaling so that the proportions of the mapping will not change), and click in the middle of the selected parallel mapping object. Then click on the corner of the cube (while still holding shift down) and move the mouse towards where you first clicked. The parallel map shrinks accordingly. When the size has halved, click a third time to finish shrinking it.

4. Render. The grooves are twice as dense.



Shrinking the groove map

Similarly, you can use the rotate tool (or object handles) to change the orientation of the grooves. The same tools, which you use for building your models, can be used for manipulating material patterns and bump maps. Further-

more, you can see the result in a visual form in the view window, without any need to look for material specific 'density' sliders, etc. somewhere. This saves you from having to learn specific material positioning tools.

The drag & drop method presented above provides a very fast way to assign materials to objects. For the materials above, it is the most suitable way. However, sometimes drag&drop does not provide enough control of how the material assignment should take place. For example, if we now want to paint a photograph of a logotype onto the wooden sphere, the program can't know if the logotype should cover the whole surface (wrapped around the sphere), or if it should occupy just a small rectangular area on top of the sphere. In such cases, it is best to use the actual mapping tools included in Realsoft 3D. However, before presenting these tools, let's examine how to create a new material.

Creating a new material

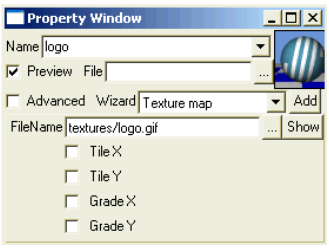
The next task is to add a small logotype on the north pole of the sphere. You can draw the logotype with a paintbrush program, or simply use any example image in the included textures folder. Using a bitmap image for painting surfaces is a very common procedure in 3D graphics. It's usually called 'texture mapping'.

1. Click the right mouse button in the Material tab of the Select window and select New/From Template from the opened pop-up menu. A file browser appears, showing the contents of the Materials folder of the program installation. Go to the Image Maps sub folder and select the item called Color Map. Click Open. A new material is inserted at the bottom of the Select window.

2. If the property window is closed, double click on the newly created material icon to open it. Change the material name from 'Color Map' to 'logo' using the uppermost Name field.

3. The property window shows some texture map specific controls. Click the browse button labeled as '...' beside the File name gadget and find the image (for example, pick the image 'logo' from the textures folder).

4. Set the Preview check box on the material editor. A small view window for previewing the current material appears. You see the result of rendering immediately. In addition, the preview image in the select window is updated. The texture map material is now ready for use, and you can close the property window.



A texture map material

The material collection, which you can see in the Select window, contains frequently needed materials. Some of the materials are ready to use ('Brushsteel' is such a material). Some of the materials define just one basic property, like a color pattern (wood, stripes, circles, checker, marble, granite, etc.), or bumps for the surface (wrinkles, waves, reptile, coral, craters, brushing), or optical properties (plastic, mirror, glass). You can combine these materials to get more interesting ones, just as we did earlier in this example.

In addition to materials of the default library, the materials folder contains many more useful materials, and you can insert these to the current library using the New/From Template tool. Some of the templates are empty skeletons like the color texture above, and you have to fill in the missing information. Others are ready to use.

Finally, the user can create almost any imaginable shading or material effect using VSL, *Visual Shading Language*. This powerful tool will be explained in detail in more advanced tutorials.

Using the mapping tools

Now we will insert the created texture map material to the scene using the accurate mapping tools:

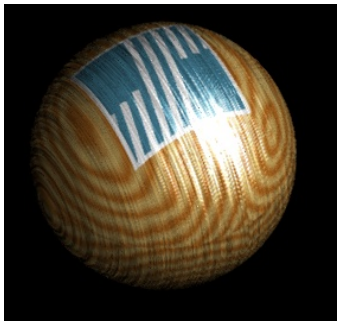
1. Go to the object hierarchy section of the select window and activate the level 'sphere+wood'

2. Take the top view (e.g. View popup menu/Camera/Top) so that the top half of the sphere is well visible.

3. Go to the material tab of the Select window and click the right mouse button over the logo material - a pop-up menu opens. Select the Map/Parallel menu item. This activates the parallel map tool.

4. Enter two points through the view window to define a rectangle. The logo texture will be placed into this rectangle.

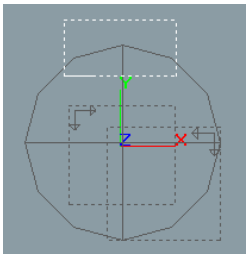
The mapping operation is now ready. Render the view to check the result.



The new texture map inserted

Rotate the camera downward or take the bottom view. Then render again. You should see the same logo image at the bottom of the sphere, but as a mirror image! This is because the parallel mapping works at an infinitely long distance by default. To change this:

1. Go to the Select window's Geometric Objects tab and select the 'Parallel map(logo)' mapping object.



Logo map encloses the top of the sphere

2. Take the front view and move the mapping object upwards using the object handles or the move tool. Move it so that the topmost part of the sphere is inside the mapping box.

3. Open the property window of the mapping object and go to the Spec tab. The property window shows you parallel mapping specific options, like the name of the mapped material.

4. Set the Finite Z option. This instructs the program to project the material only to the depth indicated by the mapping object's shape. Finite X and Finite Y do the same in the width and height directions, but they are not needed this time because the logo texture has a finite size and we did not activate the tiling option.

Now the logo map affects only the upper side of the sphere.

The last steps of this tutorial show you how to get rid of a material mapped to an object. To remove the brushed grooves from the sphere:

5. Go to the Geometric Objects tab of the Select window and select the 'Parallel map(brushing)' object.

6. Press the right mouse button and select Delete from the Select window's popup menu.



Note

The mapping object contains references to the materials, not the materials themselves. This approach has several advantages. Often you have to adjust the material properties afterwards. It is sufficient to modify the material just once to get the desired change in all objects which are made of the material. Secondly, a material is a large collection of information and therefore this method saves a lot of memory. The third advantage is that the same material can be used in many ways and with different kinds of objects. For example, a material can be wrapped nicely around a sphere using spherical mapping and spread over a flat plane using parallel mapping, without need to modify the material itself.

Blending Materials

This tutorial chapter introduces new ways of combining materials. You will learn how to mix materials:

- 50 % of wood

- 20 % of steel and
- 30 % of stone.

The tutorial also explains how to limit the effect of a material to an arbitrary area.

Level: Beginner/Medium

The chapter "Materials" explains how to assign multiple materials to a single object. Putting several materials into the same hierarchy level automatically combines those materials. This tutorial will explain how the materials are combined and how you can control that blending.

The first three materials *wood*, *plastic* and *brushing* do not share a common property. *Wood* defines the color pattern, *plastic* the shininess and *brushing* the bumpiness of the surface. Only the fourth material *logo* includes another color pattern, which replaces the wood beneath it. This happens because *logo* was mapped to the object after *wood* and therefore is further down in the hierarchy.



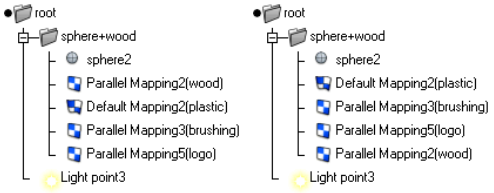
Note

Materials are evaluated in the same order in which their mapping appears in the hierarchy (from top down). A mapping applied later will be evaluated later and therefore it might overwrite all previously applied properties.

Now let us begin: Load the object, which you created in the previous chapter or the file "tutorprojects/material/mapping/multimap.r3d".Then do the following:

1. Render the view window to make sure that the logo is visible on the top of the sphere.

2. Place the mouse pointer above of the topmost mapping object Parallel Mapping7(wood); drag it over the icon of the lowest mapping object Parallel Mapping10(logo) and drop it when the cursor changes to an down arrow. This moves the wood mapping to the bottom of the list.



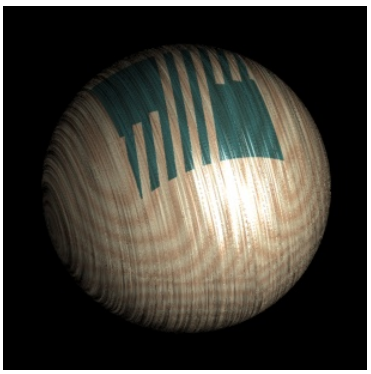
Logo is mapped after wood in the original hierarchy and replaces the wood in the area where it is defined. The hierarchy on the right side shows a situation where wood overwrites logo.

3. Render the view. Now wood grains are clearly visible also at the logo area.

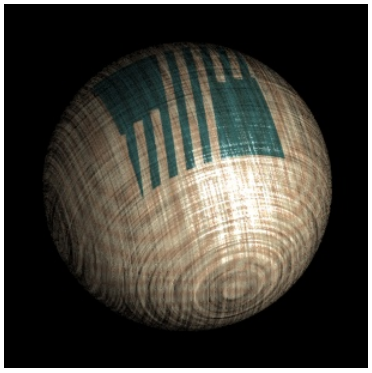
Now continue as follows:

4. Double click the wood mapping object to open the property window. Click the Spec tab to see the mapping specific options.

5. Set Scope to 0.5, close the property window and render the view. The wood material is only partly visible now and the other materials (white default material and logo on the top part) are visible through it.



50 % scope makes wood transparent and it is blended with previous materials



Combined brushing patterns

What happens if we add another bumpy material to the sphere?

6. Activate the object Parallel Mapping9(brushing) from the select window. In the front view, press the 'Ctrl' key, and simultaneously rotate the object - using the blue rotation handle - about 90 degrees. Now release the 'Ctrl' key. This is a quick way to create a rotated copy of the mapping. You should now see two copies of Parallel Mapping(brushing) in the select window.

Based on the previous experiments, you might now think that the new copy of brushing overrules the original one, because it comes later in the object hierarchy. Test this by rendering the view window. You will see two orthogonal brushing patterns, which is not what you might have expected.

This highlights that this bump map behaves differently from the color patterns. These bump maps are combined (added) whereas color maps overrule each other unless scope values are used to blend the materials in a desired fashion. The materials in the default library are deliberately constructed to follow this (useful and intuitive) convention. But please note! Realsoft 3D has an open architecture and you can easily implement any other kind of behavior using the material system. Bump maps can replace each other if required.

Local materials

Most materials in the previous example affect the target object all over. You may enlarge the sphere as much as you like, but you will not "run out" of wood. Only the logo material has a local influence at the top of the sphere. The influence area is defined by the cube-like geometry of the mapping object and is locally limited because we activated the Finite setting for the parallel logo mapping.

It is often necessary to limit the influence of a material to a more arbitrary area than a simple cube. We will practice this by making a marble plate with some golden letters on it.

You might want to load the sample project: tutorprojects/material/mapping/goldtext.r3d

Tutorial level: Medium

1. Select Environments/Default Startup from the main menu or reload startup.r3d from the main Realsoft 3D folder to initialize your working environment and get the full material library back after the previous tutorial.

2. While in front view, create the plate. For example, use an analytic cube and rename it to 'plate'.



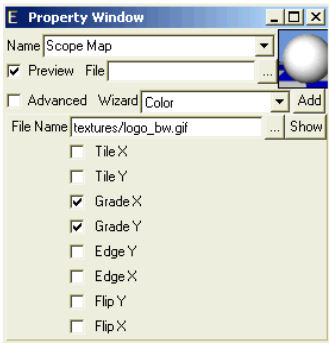
White areas will be made of gold

3. Start a paint program such as Paint, which comes with the MS Windows Operating System. Make a black image with some white text on it and save it with a suitable name (e.g. 'textures/my-textimage').

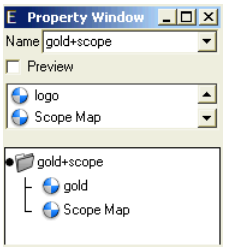
4. Make sure that the plate object is selected in Realsoft 3D. Go to the Materials tab of the Select window and drag&drop the marble material to the view window. This assigns the marble to the selected object.

5. Select New/From template from the material's select window's pop-up menu. Select the template Image Maps/Scope Map using the file browser and click Open.

6. Double click the icon of the new Scope Map material to open the property window. Set File Name to the text image which you created. Activate Grade X and Grade Y options. Grade options will blend the gold text smoothly to the background, thus reducing the jaggedness of pixel edges of letters.



Properties of the scope map material



The level material includes two sub materials

7. Go back to the select window and select New/Level Material from the popup menu. A new material appears in the select window. Using the property window, rename it to 'gold +scope'. The upper area of the property window shows the contents of the material library and the lower area the contents of the created level material. Drag and drop both Gold and Scope map to the node on the lower part. Close the property window.

8. Make sure that the level containing the plate and the marble mapping is selected. Switch to the Materials tab of the select window. Move the mouse pointer over the icon of gold+scope material and choose Map/Parallel from the popup menu. Then create a mapping rectangle in the view window. The mapping rectangle should fit inside the plate; its size defines the size of the text.



Gold text mapped to the plate



Golden text in a frame

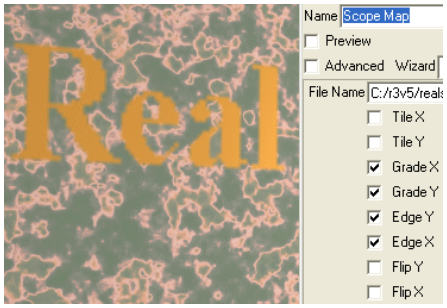
Render the view window. The result is indeed golden text on marble background, but there is a golden frame around the text as well.

The explanation for this is that the scope map cuts the influence of the gold only inside the black and white texture. Nothing limits the influence of gold outside the scope map. This is easy to fix:

9. Go to the Geometric Objects tab in the Select window and select Parallel mapping(gold+scope). Open the property window, switch to the Spec tab and set Finite X, Finite Y and Finite Z. You may recall that this limits the material to have influence only inside the mapping geometry. Close the property window.

10. Activate the side view and make sure that the selected gold+scope mapping object actually intersects the front side of the plate. Otherwise, because of the Finite options, the golden text would not appear on the plate.

11. Switch back to the front view and render. There is no gold at the edges of the plate, but if you look very carefully, you may still see a thin golden line following the edge of the parallel map. Why? Activate the Materials library on the select window. Double click the Scope Map material to open the property window. The Grade X&Y check boxes are set in order to smooth the scope image. These gradient options work also at the edges of the texture map, and the 0 level (no effect) scope of the background is blended to the surrounding level 1 (full) scope. Set Edge X and Edge Y options to disable the edge blending. Render the view to check the final result of the tutorial.



Set Edge options to remove thin gradient at the edges of the scope map

Instead of using Paintbrush for drawing the texture image, you can also use the 3D painting tools of Realsoft 3D to paint the scope mask image. For more information, read the tutorials about 3D painting.

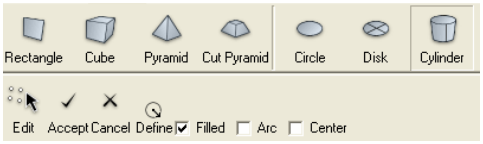
Texture Mapping

Attaching painted, scanned, etc. images to objects is one of the most commonly used techniques in texturing. This chapter examines the basic tools and controls for texture mapping.

Level: Beginner/Medium

Example file: 'tutorprojects\material\mapping\can'

In this tutorial we will create a soft drink can with some labels painted on it. For the sake of simplicity, the shape of the can may be just a simple cylinder.



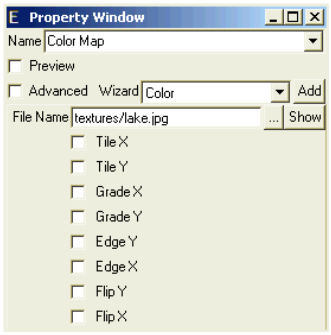
Modeling a can using the Analytic cylinder tool

1. Activate a top view and switch to the Analytic tab of the toolbar. Select Cylinder from the tool's menu. Then create the cylinder with two mouse clicks on the view window.

2. Activate a front view and stretch the length of the cylinder until the shape is suitable for a soft drink can. To do this you can use:

- the scale handle at the end of the cylinder's handle axis
- the numerical axis control in the property window's Spec tab
- or the extend or scale tools

3. Go to the Materials tab of the Select window. Choose New/From Template from the popup menu. Using the file browser, select the template Image Maps/ Color Map. Click Open.



The color texture map material

4. Double click the icon of new Color Map material to open the property window. Using the browse button at the right side of the File Name gadget, select the sample texture 'lake' from the textures sub-folder of the Realsoft 3D installation directory. You can also use any other image if you like.



Cylinder mapping

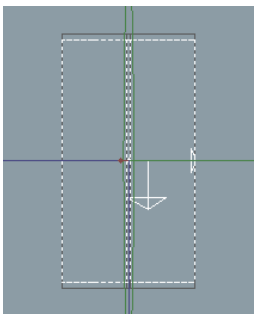
5. Close the property window. The cylinder should be selected at this point; if not, click its wireframe to select it. Then drag & drop the Color map material from the select window to the view window.

6. This drag & drop action automatically creates a cylinder mapping that matches the can. This happens because the Color Map template is defined to behave this way (find out more about preferred mappings from the VSL tutorials). The cylinder mapping wraps the selected image nicely around the can and is a good starting point for further editing.

7. Slightly rotate the view downward from the front view and render it.

As you see in the image, there are some problems. The bottom of the shows strange artifacts

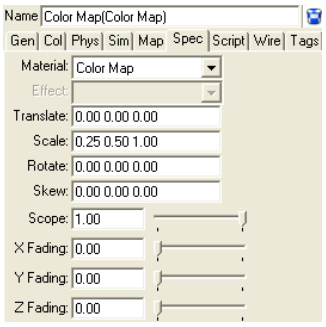
8. The artifacts at the bottom indicate accuracy problems. The cylinder map and the can have *exactly* the same length. It is impossible to say which one ends first, the texture mapping or the can (computers have only finite accuracy). Therefore, the rendering computations produce random results when trying to decide which one ends first. This is easy to fix. Scale the cylinder mapping to make it shorter using the scale handle and move it so that the mapping is in the middle of the can (see the image below). Render to verify that the mapping works well this time.





Cylinder mapping shortened and centered

9. The lake texture goes all the way around the can which makes visualizing the image hard. So let us make it smaller. Open the property window of the mapping object and go to the Spec tab (make sure that the mapping object is selected and the geometry objects tab is active on the select window). Change the Scale gadget value to '0.25 0.5 1'. Then render again. The width of the lake image has become one quarter and the height one half of the original size.



Adjusting the mapping scale



10. Keep the property window open but switch to the Materials tab on the Select window. The property window shows now the controls of the Color Map material. Activate Tile X and render again. The image pattern now is repeated and travels around the can. You may close the property window now.



Tile X activated

11. The stripe of shrunk logo labels appears in the top half of the can, but how do you put it at the center of the can? You might have noticed that the mapping specific properties not only include Scale but also Translate', Rotate', etc controls. You can use the Translate gadget for accurate numeric positioning. However, the mapping geometry itself provides a more straightforward way to place the logo. Simply drag the cylinder mapping downward using its blue translate handle.

Note that using the unified tools, you can easily create a mapping object shaped like a *cylinder sector*:

- Take a top view
- Activate the unified circular objects tool from the pull down menu: Tools/Unified/Circular Objects
- In the tool control bar, set Class to Cylinder Map and select a material from the drop down list
- Activate the Arc option
- Draw the sector

Such a sector map allows intuitive control over the mapped area without the need to use numerical scale attributes.

Texture anti-aliasing

Let's experiment a bit more with the logo material created above.

Level: Beginner/Medium

Example file: tutorprojects\material\mapping\antialiasing

Select the Cylinder+Color Map from the object hierarchy and delete it.

Go to the top view and zoom out very far. Select Rectangle from the Analytic tools menu and draw a rectangle which covers the whole view.

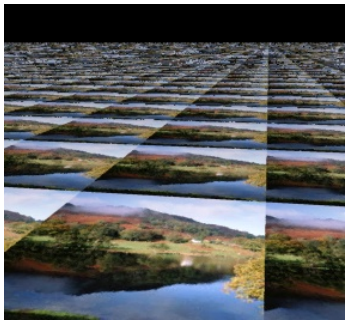
Zoom back in to a normal scale. Activate the Materials tab of the toolbar and choose the Parallel mapping tool. Select the Color Map material and draw a parallel map in the view window. The size of the map should be much smaller than the size of the rectangle.

Set the view back to the front view. Go to the Light Sources tab and activate the Ambient light tool. Click Accept. Now the simple test scene is ready.

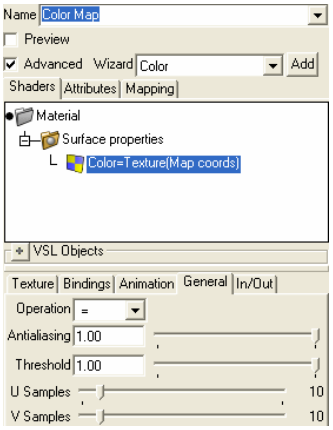
Go to the Materials section of the select window, activate the Color Map material created in the previous tutorial and open its property window. Tile X is already set but set Tile Y as well.

If the view window is in parallel projection, select Camera/Perspective from the view's popup menu. Rotate the view window upwards and render. The result should look like in the example image.

The lake image can be seen well in the front area, but the quality gets worse as the distance grows. Improving rendering quality helps a bit, but any reasonable quality cannot completely remove problems appearing far away from the camera. It is better to modify the material itself to fix this.



Bad anti-aliasing



Adjusting anti-aliasing

In the material's property window check the Advanced option to see the internal structure of the material. The tree-like VSL diagram shows the structure of the material. You can see an item 'Color=Texture(Map coords)'. Select it by clicking it. The lower part of the property window now shows several tabs. Select the General tab.

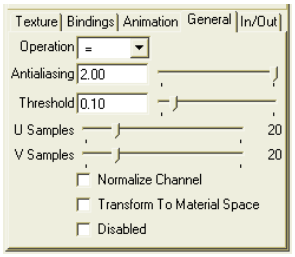
Set the Anti-aliasing value to 1. It is a good default value to start with. The resolution of the lake image is some hundred pixels in both directions. When such a texture covers only few pixels in the final rendering many samples are needed to compute it. Optimally, the amount of samples should be the amount of details, i.e. the resolution of the image. Usually this would be very slow to compute. Setting the U samples and V samples to 10 tells the renderer to compute 100 (10*10) samples in the densest areas.

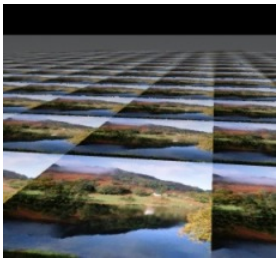


Note

The quality of the texture map image also plays a role. Poor quality image maps can’t be fixed by high anti-aliasing settings.

Render the view to test the result. If the quality is not good enough at the horizon, increase the anti-aliasing value by one. If the densest areas are well rendered, but there are problematic areas closer to the camera, halve the Threshold value from 1.0 to 0.5. The Threshold defines the minimal texture density where the texture anti-aliasing starts. If these adjustments are not enough to give a good quality everywhere, continue to increase the U and V samples.





Good anti-aliasing achieved

Material Classes

Realsoft 3D has an open architecture, which means that a material object can have any kind of property or interface. This chapter gives an overview of the material types included in the program. You can find the collection of installed material classes by opening the New sub menu from the popup menu of the select window's material section.

VSL Material

This is the most powerful and commonly used material type of the program. All materials, which you can find in the default material library, are of this type. The reason why VSL materials are so powerful is that the low level interface

of the VSL materials is a programming language. Advanced users can therefore implement unusual and customized material effects. Don't be alarmed by the term 'programming language', you do not have to write complex formulas and algorithms, because VSL materials can be built using wizards and an easy drag and drop interface. Consequently, it is easy to get started. Nevertheless, in order to fully utilize the possibilities of VSL, some programming experience is required.

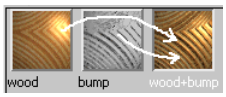
Using VSL materials is very easy. VSL makes it possible to build an optimal interface for each material type. You can see this by examining the materials of the default library in the property window.

You can find more information about this material type from the VSL specific section of this chapter.

Level Material

The Level material allows you to combine existing materials to construct more complex materials. Thanks to this, you can map several materials to an object using a single mapping object. The tutorial Blending Materials already showed one typical use of the level material: combining a material and a scope mask, which defines where the material has influence.

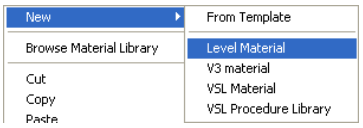
Creating level materials



Wood and bump are dragged and dropped into a level material. The result is a bumpy wood material

Level materials can be created from the select window's New menu (when the Materials tab is selected).

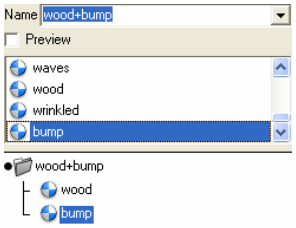
This material can refer to any number of other materials. The sub materials of a level material can be selected simply by dragging the desired materials into the level material in question.



Create a Level Material

Modifying level materials

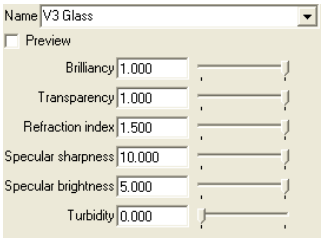
Level materials can be edited as usual: just double click the desired level material in the select window and the property window is opened allowing you to edit the material. Also the property window supports drag&drop. Drag the desired materials from the upper material list into the level material.



A level material consists of two sub materials: wood and bump

V3 Material

This material type implements the same kind of controls for optical material properties as Real3D. Brilliancy property defines the amount of mirror like reflection. The amount of diffuse shading is automatically computed from brilliancy: if brilliancy is zero, diffuse shading is at its maximum and vice versa. The material can be used for basic mirror, glass, etc. and it renders quickly.





Note

V3 material does not provide advanced editing or material wizards. The more general VSL material allows all effects that V3 material can define, but V3 material is slightly faster to render.

3D Painting

This tutorial chapter introduces the 3D painting tools of Realsoft 3D.

Level: Beginner/Medium

The 3D painting system of Realsoft 3D allows you to paint the surface of any geometric object using various painting tools, such as freeform curves, rectangles and circles. The program includes apowerful material system for defining optical and other properties of objects. Furthermore, objects can be textured using various texture projection methods, such as volumetric mapping and UV mapping. In addition to standard mapping methods such as parallel and spherical mapping, the VSL language can be used to define customized and unusual mapping transformations. Realsoft 3D also allows any material property to be defined using any number of textures per surface. The system is very open and the possibilities are endless.

The 3D painting system of Realsoft 3D is designed to support all these features.

It allows you to paint any surface property (including user defined channels), such as color, transparency, refraction or UV coordinates. The painting target may have multiple textures, and regardless of the mapping method, they all are taken into account.

Creating a football

A well-known problem in texturing spherical objects is that it is very difficult to draw textures for the poles of the sphere due to the fact that the UV space in the poles is collapsed to a single point. 3D painting solves this problem.

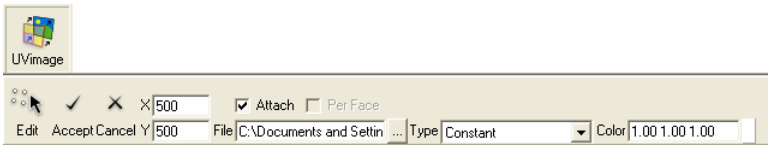
1. Create an analytic sphere using the Analytic/Sphere tool.

2. Activate the UVimage tool. You can find it from the Materials section of the toolbar.

3. Specify the desired texture resolution (for example 500*500) and set Type to Constant. Set the Color field to white (1,1,1), which is a suitable base color for the football.

4. Define a suitable File name for the image, for example 'myfootball.bmp'. Click Accept.

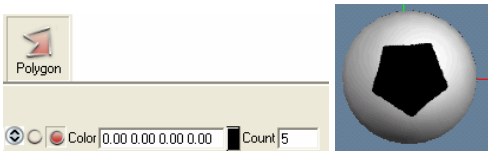
This creates and attaches a new blank white image to the selected objects (=sphere) using direct UV mapping. You can verify this at the sphere's Property window Col tab.



Activate the UVimage tool and initialize a texture for the football

5. Switch to the top view. If shaded OpenGL is not yet on, activate it from the view control bar. This way you will see in real time how the shaded surface gets painted.

6. In the select window make sure the sphere is selected. Activate the Polygon painting tool. If the sphere is not selected, the OGL display is not immediately updated after the painting action.Select Filled polygon and set Count to five. The Color field should display black color, because we are going to paint the black areas of the football. Now, enter two points through the view window to specify center and radius for the polygon to be painted.



Activate the Polygon painting tool, turn Filled on and set Count to five. Draw the first polygon onto the north pole.

7. Select Camera/Front from the view's popup menu. Then select Again from the popup menu to draw the second polygon on a new side of the sphere. The Again menu activates the polygon paint tool with exactly the same options as before, so you can just draw the polygon.

Repeat step 7, each time selecting a different view direction from the Camera menu, until you have painted all sides of the sphere.



3D Painted sphere



Note

You can use the Shift key, combined with the ViewControls, to get opposite viewing angles. For example, Shift + Top view shows the bottom view.

8. Go to the Select window's Images tab. Double click the name of the image you created with the UVimage tool to view the generated texture.

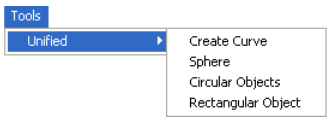
You can save the generated image using the File/Save As function. If you save the entire project, the generated image (myfoolball.bmp) will be saved also. If you only want to save the painted image, set Sections to Selected and set only the Images check box.



Image generated by 3D Painting

Using curve, rectangle and circle tools for 3D painting

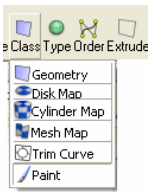
The 3D painting tool set, included in the Materials section of the toolbar, contains specialized tools for most of the common geometric shapes, like circles, rectangles, freeform curves, and polygon lines.



These four basic creation tools support the 3D painting option

In addition to this, you can also use the four tools included in the Unified menu for 3D painting. Setting the Class option to Paint activates 3D painting mode. Instead of creating new objects, the tools paint the specified geometry to those underlying objects, which have bitmap textures.

The advantage of this painting method is that modeling oriented tools have more options for defining the shape. For example, if you want to paint a circle arc by defining three points through which the circle arc should pass:

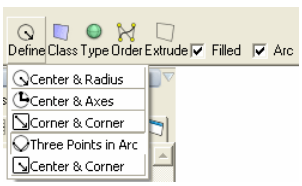


Set Class option to Paint

1. Create a sphere and attach a texture to it using the UVimage tool the same way as in the football example above.

2. Make sure the sphere is selected. From the top menu, select Tools/Unified/Circular Objects. Set Class option to Paint.

3. Set the Arc check box of the tool control bar. Set the Define method to Three Points in Arc and enter three points on the sphere through the view window.





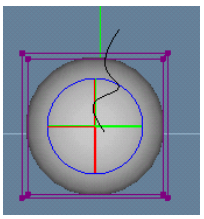
Check the Arc option and set the Define method to Three Points in Arc. Painted arc shown on the right.

Using selected objects for painting

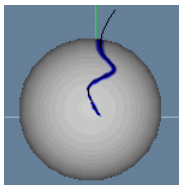
You can use any geometric object as a pen for 3D painting.

1. Create an analytic sphere and use the UVimage tool to create a blank white texture for the sphere.

2. Use the Curve tool (Note: Curve tool of the Unified menu, or the NURBS/Curve tool, but not the 3D painting tool) to create a curve over the sphere.



A sphere and a curve selected



Curve painted to sphere



The Paint with Selected tool

3. Make sure the curve is selected. Go to the Materials section of the toolbar. Activate the Selected tool and specify desired color and line width. If the curve is open like in the image above,select the unfilled drawing option. Then click Accept.

One advantage of using selected objects for painting is that you can use all geometry manipulation tools to control 3D painting.

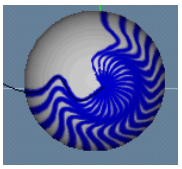
For example, you can use the Macro system to paint a certain pattern over the object:

4. Activate Macro/Record from the main menu.

5. Rotate the pen curve a bit.

6. Apply the Selected paint tool.

7. Reset Macro/Record from the main menu and hit the 'Ctrl+g' keys a couple of times to execute the current macro.



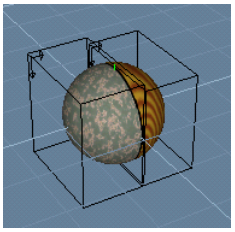
Repeat 'rotate + paint' macro to get the entire sphere painted

Converting materials to direct UV mapped textures

The UVimage tool is very powerful and allows you to do much more than just creating new blank textures for objects. This example demonstrates how you can do 3D painting by mapping multiple materials to objects using standard mapping methods, such as parallel, cylinder and spherical maps.

Let's assume that you want to create an object with a marble left side and a wooden right side.

1. Create an analytic sphere.



A sphere with two parallel maps: wood and marble.

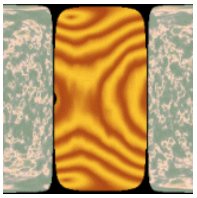
2. Go to the select window's material tab and map the two materials to the sphere by dropping the materials onto the view window. Move and scale the created parallel maps so that they split the sphere into two halves like in the image below. Make sure that the two parallel maps together completely enclose the sphere - any part of the sphere outside the mapping volume will render white (no texture).

3. Go to the object hierarchy tab of the select window. Select both parallel mapping objects, choose Properties from the popup menu. Go to the Spec tab of the property window and set all the Finite options. These options limit the mapped material inside the parallel map's cube like volume.



Select both mapping objects and set the Finite options through the Property Window/Spec tab

Rendering the view window shows the desired division between materials. However, the materials are procedural (defined by mathematical formulas). This means that if you want to export the scene, for example, to a polygon based game engine, the materials must be converted to bitmap images which then can be used as usual texture maps. Continue as follows:



Volumetric parallel mapped materials converted to a surface texture

4. Select the sphere and activate the UVimage tool. Set Type to Evaluate. Specify the desired resolution (e.g. 512*512) and file name, and click Accept.

This creates a new image with the specified name and attaches it to the sphere as a direct UV map. You may delete the two parallel mapping objects now.

The generated image can be loaded to any 2D painting program for further editing, etc.

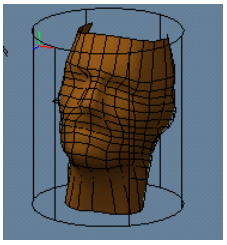
Painting SDS objects

A problem with subdivision objects is that they usually don't have a well defined default UV space which could be used to bind textures on them. For example, when modeling a head, the user starts from a basic shape like a cube or a sphere and uses extrude, subdivide and other SDS tools to add new faces and control points to the object. The tools do their best to guess sensible UV coordinates for the added points, but if the construction process is long and complicated, the UV space of the result object will not be well defined.

The following kind of problems may occur:

- UV coordinates of points may not be evenly distributed. A part of model that is big in 3D space may occupy only a little part of UV space. Consequently, it is not possible to paint small details to such parts of the model. Even the smallest point (one pixel in texture image) covers a wide area.
- Many parts of the model may have the same UV values. When painting to such an UV folded' part of the surface, other parts will get some paint as well.

Map2Obj tool allows you to redefine the UV space for objects. The general-purpose tool actually can define any surface property of SDS objects, but we will consider only UV coordinates in this tutorial.



The SDS face and a cylinder mapping object

The task is to paint a modeled SDS head. The first step is to make sure that the model has a decent UV space.

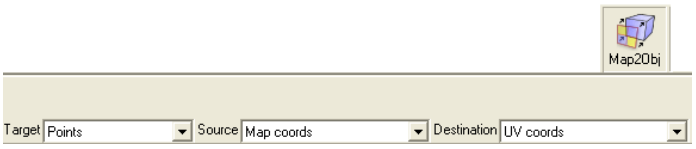
1. Model a head or load in 'models\sds\head'

2. Create a cylinder mapping object around the head from the top view (activate the Unified/Circular objects tool from the menu and set the Class option to Cylinder map). The material used is irrelevant.

3. Multi select the head and the cylinder mapping object (the order does not matter, because the tool recognizes mapping objects automatically) and activate the Map2Obj tool included in the Materials tab of the toolbar. Set the following options if not already set:

- Target: Points
- Source: Map Coords
- Destination: UV coords

Then click Accept.

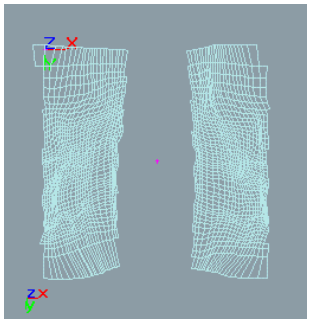


Define pointwise UV coords

The above mentioned settings convert the material mapping space defined by the cylinder mapping object to pointwise UV coordinates of the SDS head. You can verify this through the UV Editor:

4. Select the head and open the Windows/UV Editor window from the main menu.

You can now fine-tune the UV coordinates. Set the head object to point edit mode from the tool control bar. Open the popup menu of the UV editor and make sure that Current UV is selected (it selects the pointwise UV editing instead of face specific coordinate editing). Also, make sure that the defined UV channel 'UV coords' is selected in the popup. SDS objects can namely have multiple UV channels per vertex. Then edit the control points in the UV editor.



UV Space generated by the cylinder map

The property window allows you to modify UV coordinates numerically:

5. Select the SDS control point(s) by clicking/dragging them through the View window of from the UV editor.

6. Go to the property window's Spec tab.

7. The UV field shows you the current uv value of the selected point(s). Enter a new value to this field.

A note for advanced users: Map2Obj can also be applied to selected points only. This option may be necessary in defining UV spaces for complex shapes. The target material of mapping object may also contain suitable Map coords transformations. For example, the simple transformation 'Map coords = Linear(Map coords)', 'multiply' = '0.5, 0.5, 0.5', 'add' = '0.5 0.5 0.5' in the Surface properties shader maps all the points to one quarter of the UV space. Similar transformations can also be defined using the transformation controls of mapping objects (included in the Property window's Spec tab).

Use your favorite painting program in 3D painting

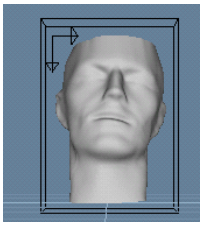
This is another example demonstrating the UV-image tool.

The exercise this time is adding a texture to a face. Painting a human face is not a trivial job, so you might want to use your favorite 2D painting program to do the actual painting.

1. Load in a model of a face (for example the included sample object 'models\SDS\head'). If the object is a subdivision object, make sure that UV coordinates are properly set (see the previous example).

2. Go to the Materials section of the select window. Create a new VSL material called 'face' using the New/From Template tool; the appropriate template is Image Maps/Color Map. Leave the texture file name empty at this point.

3. Map the face material to the face model using parallel mapping as shown in the adjacent image.



A face and a parallel mapping object

4. Zoom in so that the parallel mapping and the face fill the view window. Set view's projection to Parallel and turn shaded OpenGL on.

5. Now, copy and paste the image in the view window to your favorite 2D painting program (such as PhotoShop) by holding down the Alt key and the Print Screen key on your keyboard. Create a new file in Photoshop (File size is automatically the size of the clipboard) and Paste in the image. You can also use Render/Save to File menu of the view's popup to save the contents of the view window and then load it into the paint program.

6. Paint over the shaded face image. When done, use the Crop tool to create a new image, which contains only the pixels inside the parallel mapping.

7. Save the image with a suitable name.

8. Go back to Realsoft 3D. Select the face material and open the property window. Set File name of the material to the image you painted and saved. Close the property window.

9. Ray trace the view window. The face image you painted in your favorite 2D painting program now matches exactly the face model. However, it is mapped using parallel mapping and cannot follow possible deformations of the face. Therefore we have to convert the parallel map to an UV map.

10. Activate the UVimage tool: Set Type to Evaluate. Specify a suitable resolution (can be the same as when you painted the image) and a file name and click Accept.

You have now converted the parallel map material to a direct UV texture. Delete the parallel map object and the face material. Another possible workflow for doing the actual painting in a 2D painting program is the following:

1. Use the UVimage tool to initialize a direct UV texture for the model the same way as in the first Football example.

2. Use the 3D painting tools (the Curve painting tool is quite suitable) to draw rough outlines of important areas like eyes, mouth, eyebrows, etc.

3. Save the project. Start the paint program you like most and load in the UV texture, which already has good drawing guides for the face. Finish the face drawing.

Paste colors on objects

Consider the following problem: an image of a face should be painted on another, completely different face. The image cannot be simply projected onto the face with a parallel mapping, because the size of the nose may be different, etc.

The Paste Colors on Objects tool provides a solution to this problem. This tool allows you to use any 3D object as a projector for projecting the image on the target surface. The image is projected using the current view projection: the target object will get the color of that point of the projector object that appears in the same location in the view window.

1. Load/Create a model of a face. If the face is a subdivision object, set up a proper UV space as instructed in the example 'Painting SDS objects' above. You may use one of the provided face objects, for example 'models/head.r3d'. Turn the view window to show the face from the front.

2. Create an empty UV map for the face using the UVimage tool (see the Football example above). Name the texture file as 'matchedface'.

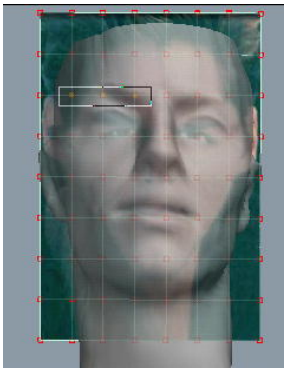
2. Create a rectangular Nurbs mesh of say, 10 times 10 points. The denser the mesh, the more accurately you can define the projection. Move the mesh behind the face, but in front of the back of the head.

3. Open the property window, go to the Col tab and enter the name of the face image to the Texture field.

4. Select the face object and using the property window's Wire tab, set the No Shading option for it. Alternatively, you may set the 'Forced visibility' wire setting of the projector mesh to, say 0.5. This forces the texture mapped projector surface to show through the shaded face. In both cases the purposes is to see the shape of the face object and the face image at the same time.

5. Go back to view the face from the front view and turn shaded OpenGL on. You will now see the face image through the wire frame of the face object.

6. Select the rectangular mesh. Then use deformation tools and point editing to move the points of the mesh to match the image to the face object.



Fitting a face texture to a model of another face by deforming the projector mesh

7. When done, make sure the nurbs mesh is selected. Go to Materials toolbar section and click the Paste Colors tool. Specify a desired sampling quality (e.g. 50*50) and accept the tool. The texture map on the nurbs mesh is projected onto the surface of the face.

You can now delete the projector mesh. You can find the matched face image in the Images tab of the Select window and view it using the property window.

Because the target of painting (or its UV space) can be very curved, the tool cannot guess a good sampling quality. Therefore, the tool asks the user to specify the number of samples to be taken. High sampling values may take a long time to compute. If the suggested quality 50*50 wasn't good enough, you can activate the Paste Colors tool again using higher Resolution values.

Note that you can paint any number of images to the head this way. Just rotate the view and repeat the steps 2-7 to paint ears, wrinkles, hair etc. over the head. New paste operations will overwrite earlier painting on overlapping areas.

Preparing an object for 3D painting

In the previous examples, we used the UVimage tool to create and attach a new texture to an object. You can also attach an existing image to an object and paint to it with the 3D painting tools.

There are two ways to connect the target image of 3D painting to the target object of painting:

- Direct UV map
- Material system

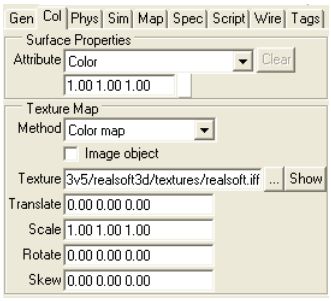
Direct UV Map

The UVimage tool used in the football example above assigned the created image to the sphere by using the direct uv map method. If you want to use an existing image, you can do this manually as follows:

1. Select the target object and open the Property window.

2. Go to the color tab and specify the desired image in the Texture field.

The specified image is now used as a direct UV mapped texture for the object and all 3D painting tools can be used to paint the object.



Use myimage.jpg as a direct uv mapped texture

Material System

Also bitmap textures included in materials and mapped to objects using the mapping tools (parallel, cylinder, spherical, etc.) can be painted with the 3D painting system. The advantage is that you can paint simultaneously to multiple images. Furthermore, the texture map objects can define not only color, but also any other information. You can paint transparency, bumps, and even visibility of materials to surfaces this way, as shown in the advanced tutorials.

For example, to paint bumps to a sphere (sample project: tutorprojects/material/3dpaint/paintbumps)

1. Select the sphere.

2. Activate the UVimage tool. Set Type to Constant color, Color to black and clear the Attach option. Resolution can be e.g. 256*256. Set an appropriate file name, for example 'textures\testbumps'. This creates an empty black texture map, but does not attach it to the sphere as a color map.

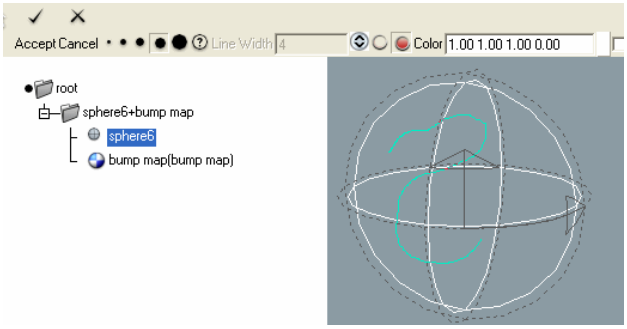


Initializing a bumpmap

3. Go to the Materials section of the select window and select New/From Template from the popup menu. When the file browser appears, select Image Maps/Bump Map and click Open. Open the Property window and set File name to 'textures\testbumps' or whichever name you used in the previous step. Close the Property window.

4. Drag and drop the newbump material to the view window. This creates a spherical mapping that assigns the bumpmap to the sphere.

5. Activate, for example, the Curve painting tool from the Materials section of the toolbar. Select a medium size brush, set non-filled line type and set the color to white. The brighter the color, the higher bumps will be drawn. Then press the left mouse button down over the sphere and draw a curve by dragging the mouse (LMB down). Release the mouse button when the curve shape is ready. Render the view window to verify the result. Use 'Again' from the view popup to draw new curves.





Drawing bumps

You can also initialize an empty black UV texture map, 3D paint to it and afterwards change the color map to a bumpmap. The advantage of this approach is that you can see the result of painting in real time using shaded OpenGL.



Note

When painting an object that has multiple textures using 3D painting tools, it may be necessary to protect all but one texture at a time from being modified. For example, an object may already have a color map, which must not change during bump painting. This can be done simply by selecting all mapping objects that you do not wish to paint to, and set Hit Invisible from the property window's Gen tab. If the object has a direct UV map (i.e. Texture field set in the object property window's Col tab) and you wish not to paint on it, the name string must be temporarily cleared. Cut the name, then past it back when painting is done. The trick works because the render engine that handles 3D painting is the same as the ray tracing engine.

Visual Shading Language

Introduction to Visual Shading Language

Welcome to *Visual Shading Language (VSL)*

VSL provides you with power equal to programming languages. It allows you to define very advanced material effects, such as procedural bump maps, anisotropic lighting effects or light radiation that makes the target object transparent!

Despite the fact that VSL is actually a programming language, you don't have to be a programmer in order to use it. Thanks to an easy-to-use drag&drop interface, most of the power of VSL is accessible to users with no programming background.

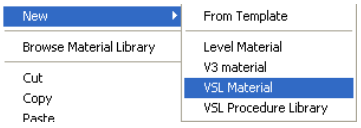
This tutorial shows you how to get started in creating fancy material effects with VSL.

Tutorial level: *Beginner*

Constant red

In this example we will create a material which turns the surface red.

1. First we need a new VSL material. To create one, switch to the Materials tab in the select window, open the pop-up menu by clicking the right mouse button, and pick New/VSL material from the list.

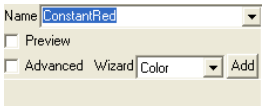


Create a new VSL material

A new black icon is created into the material tab with the default 'MaterialX' name. If Show Previews option is off (you can check it from the select window's popup menu), you will find the new material at the beginning of the material name list. Slowly double click the new material icon to activate the rename tool. Enter the new name 'ConstantRed' and press Enter.



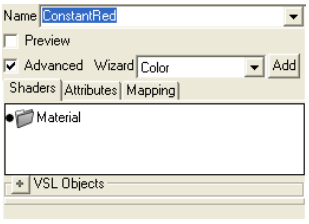
A new empty material is inserted into the material tab of the select window.



A blank material

2. Double click the new black material icon and the property window is opened. The created material does not define any material properties. Therefore, its property window is empty.

3. Check the Advanced check box in the top left edge of the material property window. This opens the actual VSL editor. You can also check the Preview option, to see the effects of editing actions.



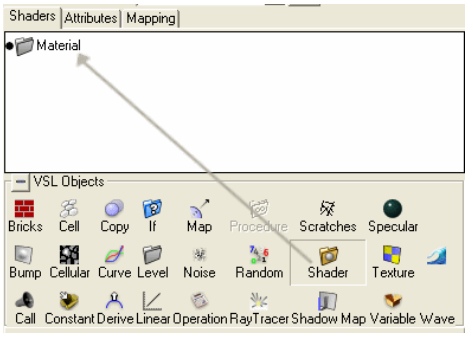
The VSL editor

4. Click the VSL Objects button (the small + sign). This shows you the VSL objects, which you can use to define material properties.



Available VSL objects

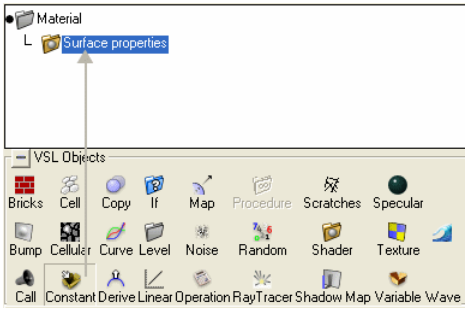
5. Our goal was to create a material, which simply turns surfaces red. This means that we want to define the *surface properties* of objects. Drag the Shader icon from the Available Objects and drop it over the 'Material' node appearing at the top part of the window. Note how the mouse pointer changes to indicate when dropping is allowed. This creates a new node to the VSL tree. The new node has the name 'Surface properties' and is automatically selected, as the highlighting of the name indicates.



Drag and drop the Shader icon into the Material node

Look at the lower part of the VSL editor. It always shows the properties of the selected VSL object. The current selected object, 'Shader', has only one property, 'Type', which is by default 'Surface properties'. Click the right side of the 'Type' gadget to open the whole list of available types; it includes names such as 'Light properties', 'Volume illumination', and so on.

Many of the shader types are easy to understand from the short but descriptive name. If you want to make an object red, you need a 'Surface properties' shader, because 'red color ' is a property of the surface (red paint, etc.). If you want to define how fog reacts in light, you need a 'Volume illumination' shader. Makes sense, doesn't it?



Drop a Constant object to Surface properties

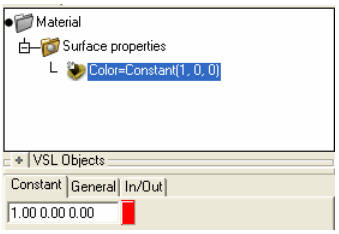
The default type 'Surface properties' is exactly what we need, so do not change it, just close the type list by clicking the right side of the gadget again. Surface properties is the most frequently used shader, and that is why it is the default type for all newly created shaders.

5. Drag & drop the Constant icon from the Available Objects box inside the Surface properties node. Again, a new node is inserted into the VSL tree.

6. The new Constant node is again selected, and the lower part of the window shows you its properties. The default color value is black. Click the small sample color square and change the color to red.

7. Check the Preview button to update the icon in the material select tab and to view the effect of the material.

Your first VSL material is now ready.



Surface color set to red

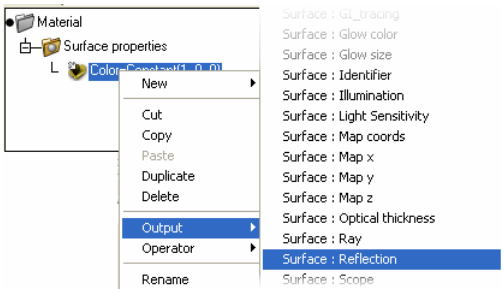
Red reflections

Next we modify the ConstantRed material slightly.

In the example above, we used a VSL object 'Color=Constant(1,0,0)' to achieve a red surface. To be exact, we defined such a property that makes an object appear red when ordinary white light illuminates it. This property is called the *diffuse color* of an object, and it has a very strong impact for the appearance of most real world objects and materials (cloth, paper, matte-like surfaces in general). Theoretically, diffuse color describes how much and in which color tiny irregularities and bumps of a surface scatter light around.

Diffuse color alone is not sufficient when we try to simulate real world materials and surfaces. Another well-known property, which describes what happens when light hits a surface, is the amount of mirror-like reflection. We will experiment with this next;

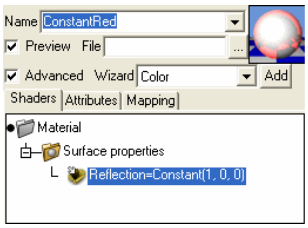
Using the previous material, make sure that the Constant VSL object in the material hierarchy is selected (its name is highlighted). Move the mouse pointer over the area, which shows the VSL structure of the material, and open the popup menu using the right mouse button. Open the Output sub menu. It includes a long list of items: Surface:Alpha, Surface:Color, and so on. Select Surface:Reflection from the list.



Selecting the output channel for a Constant VSL object

The long menu list shows the properties, which describe the type of the surface: the amount of diffuse color, reflection, transparency, etc. Such properties are called *Channels* in Realsoft 3D. You can control the appearance of a material by writing desired values to suitable channels, just like we did above (first diffuse color and now mirror reflection).

Some of the channels have names that may look hard to understand: UV coords, Map Z, for example. These channels contain information computed by the rendering engine and they are made available for the user so that he/she can find out what is going on during the rendering and ultimately, customize all the rendering actions.



Red reflections and white diffuse color

Let's stop the theoretical considerations and continue the tutorial. The Constant VSL object now defines the amount and color of mirror reflection. Consequently, the preview window shows the material again white, except that there may be some red reflections visible. All objects have white diffuse color by default. Thanks to this, we can see newly created objects in rendering without a need to define colors for them. Nothing in the VSL structure constructed so far changes this default setting, and therefore the diffuse color is white again.

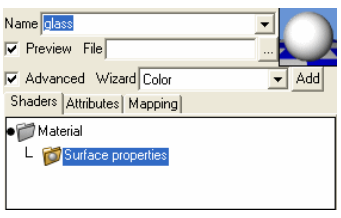
The reflections look strange, because the floor below is white and blue, but reflections show it red. Anyway, this is what we defined: the color value of the constant object is still red from the first VSL tutorial. As a matter of fact, although this is an unusual effect, even stranger phenomena occur in real world materials. Therefore, it is important that we have the freedom to implement the wildest imaginable shading effects using VSL.

Glass

Next we build a bit more complex material: glass. There is a glass like material available in the material library, but it is useful to learn how such a material is defined step by step.

Select New\VSL material from the select window's popup menu and rename the created blank material as Glass.

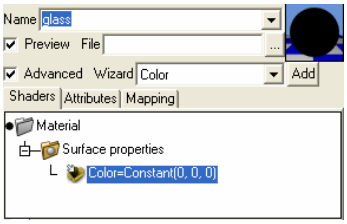
Open the property window unless already opened. The newly created material is automatically selected. The Advanced and Preview options should also be set from the previous example; if not, activate them.



The starting point

The first step is the same as in ConstantRed: drop a Shader from available VSL objects into the root level material node. Most glass properties can be defined in Surface properties, and therefore we do not have to change the default type of the shader.

So far, the material has only the default properties: white diffuse color, thanks to which we can see the preview sphere. In the example image above, the simple preview sphere has a blue and white checker floor below it. This makes it easier to see how the material will interact with the environment. The preview project is fully customizable and can be changed (see the material preview tutorial for details).

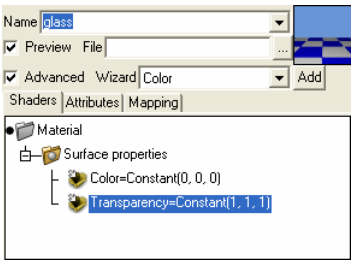


Black diffuse color

Drop a Constant object from the available objects into Surface properties. The constant object is inserted, assigning the constant black value to the (diffuse) color channel. Consequently, the preview turns black. This is what we want, because the surface of glass is so smooth that it does not scatter light randomly around but lets it through and reflects it in certain exact angles. Keep in mind that random scattering is the source of matte-like diffuse shading.

Drop another Constant object into the Surface properties shader. Open the popup, and change the Output channel to Surface:Transparency. Changing the output channel was explained in the Red reflections example above. Nothing happens yet, because the constant transparency value is still black. Change it to white (1 1 1) using the gadget on the bottom of the VSL editor. The preview sphere disappears.

This kind of material is very boring, because applying it to an object is almost the same as deleting the object - nothing remains visible. The material lacks an important property of real glass, namely the ability to refract light. Let's fix this.



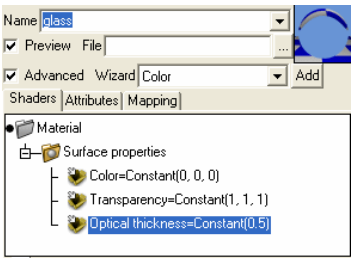
Fully transparent sphere

Drop a new constant object to surface properties and change the output to Surface:Optical thickness. The default thickness is zero - the thickness of empty space. Change the value of constant object to 0.5 using the gadget at the bottom of the property window. Note that optical thickness is defined by a single value, whereas color has three components: red, green and blue. The constant object automatically changes its value gadget to match the sub channel dimension of the output channel.

Note that you can compute the refraction factor from the optical thickness value using the simple formula

```
Refraction index = 1/(1-thickness)
```

Assigning the value 0.5 gives refraction index = 1/(1-0.5) = 2. The material mimics real glass in this respect. Note that as you become familiar with VSL, you can easily build a material, which uses the preferred way of measuring the refraction.

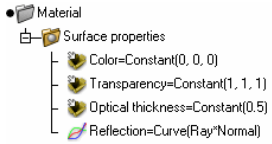


Refracting glass

The preview window changes, but you may not see any refraction yet. The preview window is actually a small view window and you can rotate, pan and zoom around just like in normal view windows. Move the mouse over the preview window, press Alt key down and drag with the right mouse button to rotate the view upwards. At certain angles, the refraction of the blue floor becomes visible. It is quite extreme, because the glass lens here is fully spherical. Think about eyeglasses with spherical lenses - certainly hard to wear and they turn the world upside down.

The material now looks like glass, but we can still improve it. You have probably noticed that all real world glass objects not only refract light but also reflect it. The mirror reflection is sometimes weak, but it exists. The previous example already showed how to make an object reflective. This time we will define the amount of reflection in an advanced manner: reflection should depend on the viewing angle towards the surface. The smaller the angle becomes, the more reflection appears (which happens in the real world as well).

Drop a Curve object from Available Objects into Surface properties. The preview turns funny because the default settings are all wrong.

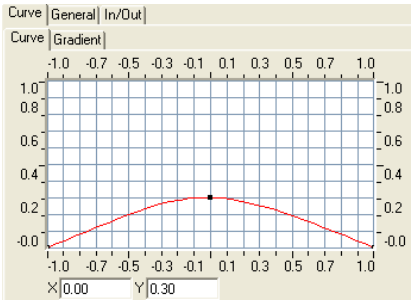


Reflection defined by viewing angle

Open the popup menu and change Output to Surface:Reflection. Open the popup again. There is one new entry, Input 0. The Curve VSL object computes values to the output channel by applying a user drawn curve function to values read from an input channel. Set Input 0 to Surface: Ray*Normal. This channel is maintained by the ray tracer and measures exactly the property we want: how perpendicular the ray from the camera is to the surface. The range is from -1 to 1, the middle value zero being the case when rays travel along the surface (perpendicular to the surface normal).

The curve gadget at the bottom of the property window shows a straight line ascending from zero to one. Move the mouse pointer over the curve and open the popup menu. Select Set Minimum and Maximum values from it. Change the MinX from zero to -1, hit Enter and close the window. Now the curve covers the whole range of possible Surface:Ray*Normal values.

Edit the curve shape so that the maximum height is in the middle (see the picture below). You can select existing points by clicking them or with a drag box the same way as when editing geometric objects. To add a new point, simply drag the curve from a point holding the left mouse button down. You can delete the unnecessary points using the Delete function from the popup menu. Note that there are actually three curves, but edit them all to a same shape, because we do not want color distortions this time - white should look white in reflections. The the start and end values of the curve can be slightly above zero, which makes the surface reflective everywhere.



The curve that defines reflection

Now the reflection is ready, but you may not see much change in the simple preview scene in the small window. Nevertheless, this kind of subtle detail is important when building truly photorealistic scenes.

The final step is adding some shine to our glass.

Drop a Shader object to the Material node. Change the Type from Surface properties to Surface illumination.



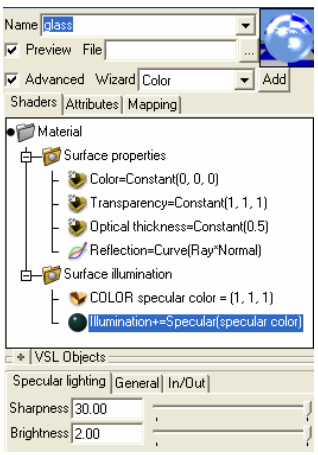
Surface illumination shader added

Drop a Specular object to the Surface illumination shader. Take a look at the inserted Specular object: It uses the Color channel as an input the same way as the curve object above uses Ray*Normal. Usually this works fine, but now the color is set to black (the first Constant assignment in surface properties) and therefore the default input does not work in this case.

Drop a Variable object on top of the Specular node in the VSL structure. Wait until the mouse pointer shows an up arrow indicating that the new object will be inserted before the specular object. If the Variable object was added after Specular, you can drag and drop it into a new position in the VSL hierarchy; just watch the mouse cursor when dropping to find out the correct end position. Actually variables can be inserted anywhere in the hierarchy and they are equally visible within the parent level, but declaring variables in the beginning of the level makes VSL code more readable.

Using the property controls of Variable, change its name to SpecularColor, set the Initialize checkbox and define a value 1,1,1 to the Initial Value gadget.

Click the Specular object in the VSL tree to activate it. Open the popup menu and change Input 0 from Color to SpecularColor. Now the specular object uses white color defined by the SpecularColor variable to compute the highlights. You can see this in the preview window. Adjust the sharpness and brightness values of the Specular object until the highlights are suitable. The glass material is now ready.

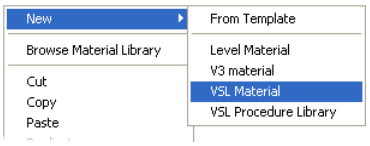


Glass with specular highlights

VSL Wizards

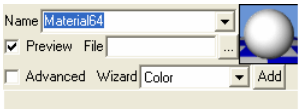
The easiest way to create new kinds of VSL materials is to use VSL wizards. These wizards are available in the VSL editor. Wizards construct the necessary material structure automatically, and insert suitable controls for easy use.

For example, to create a marble like material using wizards:



Create a new material from the select window's material tab.

1. Switch to the material tab of the select window and create a new VSL material using the select window's popup menu.



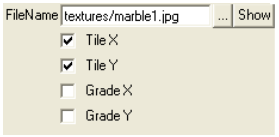
Property window with Preview enabled

2. Double click the created blank material icon to open the property window. Check the Preview option so that you can preview the material while the construction proceeds.



Add a texture map to the material

3. Select Texture map from the wizard list and press the Add button. New controls appear below the wizard list.

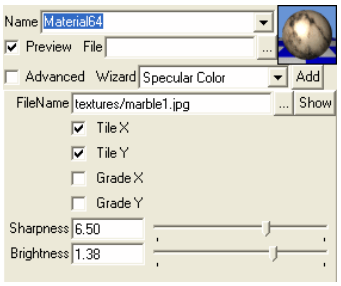


Texture map properties

4. Check the TileX and TileY options, and click browse to find a suitable texture map file. For example, textures/wood.

5. Select Specular Color from the wizard list, and press Add. New controls appear.

6. Adjust specular sharpness and brightness until the material looks suitable on the preview window.



A marble material created using the texture and the specular color wizards



Note

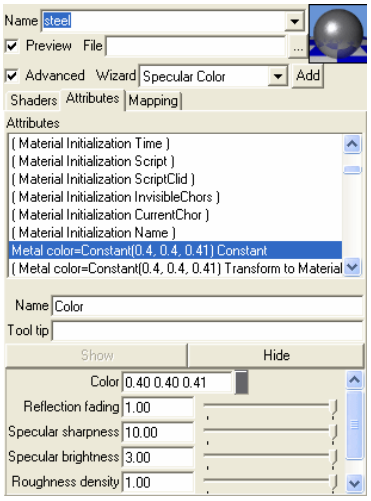
All wizards are editable - to change a color texture wizard is simply a case of changing the outputs; To change the texture wizard for example, to a fade map, change the output from Color to Fade. The wizards are supplied to make creating materials easier and as a base to work upon.

Building an Interface for a VSL Material

Consider the following situation: You have written a very powerful and nice material using VSL and you want to let other users to play with it. The VSL tree consists of 50 objects doing all kinds of fancy things. By changing certain attributes of certain VSL objects, you can control the appearance of the material.

So, how to expose only the key attributes (say irregularity and thickness of wood grains) to the end user and hide the unnecessary complexity from the user?

The VSL editor includes tools for solving this problem. It allows you to expose only selected attributes of a VSL material. Other users can adjust the properties of a VSL material without knowing anything about its internal structure.



Double click the desired property to show it in the easy-to-use interface

Setting up the interface

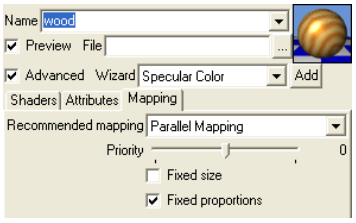
The Attributes tab in the VSL editor lists all the attributes of the VSL objects that has been inserted to the VSL tree of the currently selected material. The attribute names are the same or similar to the labels of gadgets, which are visible when you edit VSL objects. You can pick the suitable attributes and add them to the tailored interface.

You can rename the exposed attributes using the Name field, in case the default label is not descriptive enough. For example, if you expose the same attribute of two different VSL objects, say the file name of a color texture map and the file name of a reflection map, renaming is necessary.

Selecting the preferred mapping method

The Mapping tab of the Advanced material properties contains settings, which define how the constructed material should be mapped to target objects. It is quite a common situation that a material should be always used together with a particular kind of mapping. A good example is wood: real world wood material consists of circles, which are parallel mapped along the axis of a tree. Therefore, only parallel mapping works well with wood.

The mapping specifications are used by the drag & drop system. The user can overrule them by selecting another mapping type from the mapping creation tools.



Selecting mapping options for drag & drop

The following controls are available:

- **Recommended Mapping**: select the preferred mapping type from this gadget.
- **Fixed size**: If set, the size of the mapping geometry does not automatically resize to match the size of the mapping target object.
- **Fixed proportions**: If set, the proportions (width, height and depth) of the mapping geometry do not automatically stretch to match the size of the target object. For example, parallel map always becomes a symmetric cube if this option is set.

If the recommended mapping is set to Any Mapping, drag & drop system creates a mapping that best matches the geometry of the target object. Spheres will get a spherical mapping, cylinders gets cylindrical mapping, etc.

Bump Mapping

Bump mapping is one of the key techniques when building realistic materials. It provides a quick and memory efficient way of adding fine detail to surfaces.

A traditional way to define a bump map is using a bitmap image to define the bumps: the brighter the color, the higher the bump. The major advantage of this approach is that the user can define the shape of bumps by drawing, which is intuitive. Disadvantages include high memory usage, finite resolution and the regular grid-like nature of the bumps.

Bumps can also be defined procedurally using VSL objects such as Noise or Wave. This method is suitable for representing various semi-regular real world surfaces, such as skin of a reptile, surface of a rock, waves on water,

etc. Procedural bumps can have very high accuracy (= level of detail), and they do not require much computer memory.

Realsoft 3D provides two different solutions for bump mapping:

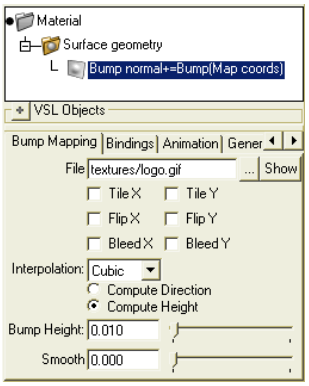
- Modifications of the Bump height channel in the Surface geometry shader
- Direct modifications of the Bump normal channel in the Surface properties shader

The first alternative is a general method and suitable for all kinds of bump mapping situations. Direct Bump normal modifications are more limited but are usually faster to compute. Both approaches are considered below in more detail.

Using the Bump height channel in a Surface geometry shader

This is the recommended bump mapping method because of its generality. The examples below show the structure of image based and procedural bump maps.

Image bump map



The Bump VSL object can be used for computing bump height values from a bitmap. The Texture object is not continuous enough for smooth bumps.

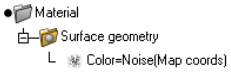
- Select Cubic interpolation for smooth bumps, or apply Smooth filter for even greater smoothing.
- The Compute height option should be set.
- Use the += operation to add new bumps to previous ones (to change this go to the General tab).
- Do not General/Normalize the bump height channel (see the General tab).

Example file: Materials/Image Maps/Bump Map

Procedural bump map

The image on the right shows a simple procedural bump map. The height of the bumps can be adjusted using the Amplitude control of the Noise object.



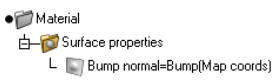


- Bump height units are measured in meters. Therefore, height values are typically very small, from 0.001 to 0.01 (1 mm - 1 cm).
- The computed Bump height value should be added to the previous value (+= operation). This ensures that several bump maps can be combined without problems.

Example: Default material library/Wrinkled

Using the Bump normal channel in a Surface properties shader

Image bump map



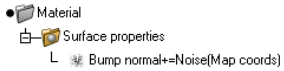
The VSL structure of a typical image bump map is shown in the image.

- The mapping geometry should match the shape of the target object: spherical mapping is suitable for a sphere, cylinder mapping for a cylinder, parallel mapping for a planar surface and default UV mapping for a freeform Nurbs surface. This method cannot be used for complex SDS shapes.

- General/Normalize option of the Bump operator should be set because the surface normal is needed in a unit form.
- General/Transform to material space option of the Bump object should be set so that the normal is modified in a mapping space relative way

Example file: tutorprojects/material/vsl/pixelbump

Procedural bump map



The VSL structure of a typical mathematical bump map is this:

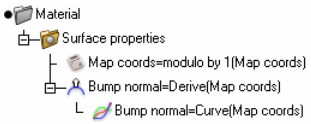
- General/Normalize should be set.
- The noise should be centered on zero: set Noise/Base to -h and Noise/Amplitude to 2*h, where h is the magnitude of noise (for example 0.1).
- General/Transform to material space option is not necessary, because noise is similar in all directions. However, if the bump procedure defines regular shaped bumps, which should follow the target surface, the transform option should be set and the mapping geometry and surface geometry should match.
- The Noise operator works best with a parallel mapping, because the computed coordinates are continuous. Spherical mapping renders with a visible seam.

Example file: tutorprojects/material/vsl/mathbump

Curve object bumps

The Derive object can be used to transform an arbitrary VSL object to a bump map suitable form. For example, one can draw the shape of bumps using the curve object.

The first VSL object in Surface properties shader is Operation/Modulo by 1. This object computes repeating sequence of 0..1 coordinates and hence creates tiling of the bump pattern. The Curve object needs not have Normalize or Transform to mat space set, because the Derive object does this. Bump height can be adjusted by modifying the vertical scale of the curve.



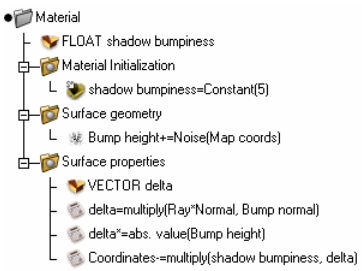
Bump map operates on two-dimensional surfaces, and therefore the third curve is unused. The first (red) curve controls bump height in the horizontal direction, the second (green) curve controls bumps vertically.

Objects such as Noise or Random do not require use of a Derive object, because Derive processed noise is just another noise field and the same applies to Random as well.

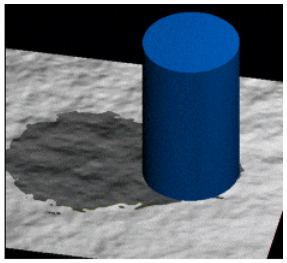
Example file: tutorprojects/material/vsl/curvebump

Bump map and shadows

An additional level of realism can be obtained by distorting the shadows on a bumped surface. The material presented here shows how to achieve this:



- The Material initialization shader is used to initialize the variable defining the amount of shadow distortion.
- Surface geometry computes the bump map in the usual manner.



A shadow on a bumpy surface

- In the Surface properties shader, the surface point is moved the amount defined by Bump height channel to the direction of the surface normal. This affects the illumination and shadow computations, which will take place later.

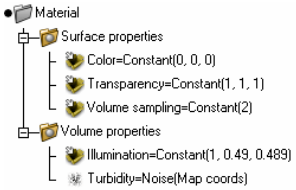
Some extra care is taken to move the point to the camera side of the surface, not behind it. Moving the point behind the original surface would make the surface cast a shadow onto itself.

Example file: tutorprojects/material/vsl/bumpyshadows

Fog Examples

Plasma fog

Plasma like fog, that is, fog which is self illuminating, can be simulated by the following VSL structure:



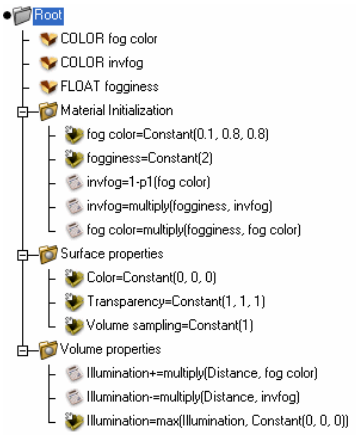
- The first two VSL objects in the Surface properties shader make the fog surface invisible. Assigning the value of 2 to the Volume sampling channel activates the volume effects: the interior of the fog cloud is studied by taking two samples at suitable intervals. For a simple fog like this, even a smaller value of 1 would suffice.
- In the Volume properties shader, a constant value is assigned to the illumination channel. The Illumination channel represents the total result of shading. This material assigns an illumination value directly. This means that light sources or shadows will not affect the fog, it appears self-illuminating.
- A Noise object defines the Turbidity property. Use a parallel mapping to map this fog material, then noise uses continuous coordinates without seams. Noise is not centered at zero because negative values would result in empty patches in the fog.
- Turbidity, the density of fog, is the strongest fog attribute in the sense that other illumination properties define the properties of one single fog particle, and the final result (color in the rendered image) is after all defined by turbidity. This makes sense: one tiny fog particle does not change the shading much, no matter how bright it is. There must be many such particles (=some amount of turbidity) before the contribution becomes noticeable.
- Remember to check Volumetric effects in the Render settings/Ray tracing options, to get the effect visible.

Example file: tutorprojects/material/vsl/plasma

Linear fog

The rendering system handles the blending of fog effects to other illumination automatically. The blending ratio is defined by the Turbidity property, which defines the density of the fog. The blending process is not linear: The longer the distance, which the light travels inside the fog, the stronger the influence of the fog becomes, but the fog never stops the ray completely.

The example image shows the structure of a material, which overrules this default fog shading. The fog effect defined by it is linear: if a layer of say 1 meter changes the illumination traveling through it by 50 %, then two meters of fog hides the background completely. The density of the fog defines the distance, which is required for full impenetrability.

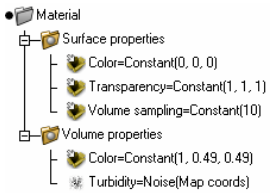


- Turbidity channel is not used at all. The material computes and assigns the fog effect to the illumination channel. Leaving turbidity to zero means that no default shading will take place.
- The fog is similar to the previous example in that it is self-illuminating.

Example file: tutorprojects/material/vsl/linearfog

Shadows in fog

The two fog types presented above are self-illuminating: light sources do not affect them at all. A fog that reacts with light can be obtained by the following VSL code:



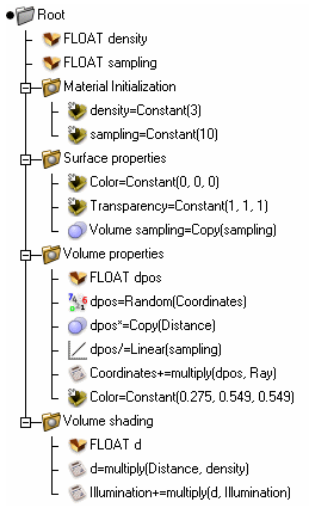
- The interaction with light sources is obtained by assigning a non-zero value to the Volume:Color channel in the Volume properties shader. The color channel defines the diffuse shading properties of the fog particles.
- The Volume sampling channel has now a much higher value than in the previous example. The higher the value, the more accurately the shape and details of the shadows inside the fog are rendered.
- Ensure that Volumetric effects and Lighting in volume are set in the Render settings/Ray tracing options.
- Only the light sources having the Volumetric option set (set by default) create volumetric effects.

Example file: tutorprojects/material/vsl/shadowfog

Additive fog

The Linear fog example above explained how to define custom fog shading for a self-illuminating plasma fog using the Volume properties shader. It is possible to customize the default fog shading of fog that reacts with light in the Volumetric shading shader, which is evaluated after fog illumination computations.

The material below shows one such example, additive fog. The fog is diffusely illuminated. It has the special property that it never darkens the illumination behind it. A normal fog does this: black smoke can hide flames of fire.



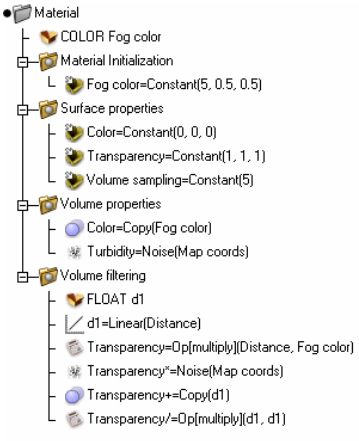
- Density is a parameter that the controls the thickness of the fog.
- Sampling defines how accurately the shadows in the fog are computed.
- The main parameters are initialized in the Material initialization shader.
- The Surface properties shader makes the fog surface transparent, removes diffuse shading from it (Color=black) so that the surface becomes clear, and launches volumetric sampling by copying the sampling parameter to the Volume sampling channel.
- The Volume properties shader does not set the Turbidity channel, which controls default shading of the fog. Instead, it moves the sample position randomly along the traced ray. The sampling density and distance control the amount of random movement. Therefore, the random variation keeps the samples inside the fog. The random factor improves quality and helps to detect small shadows.
- Volume shading computes fog illumination, which is proportional to density and distance. This illumination is simply added to any previous illumination. This makes the fog additive.

Example file: tutorprojects/material/vsl/additivefog

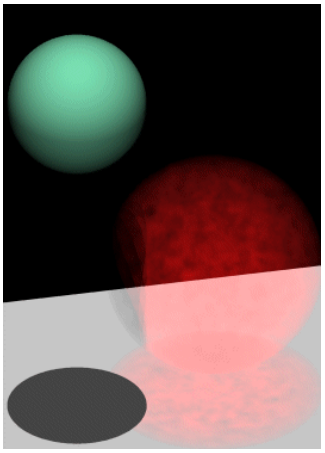
Shadows by fog

This example demonstrates how to use the Volume filtering shader. In this shader, it is possible to define how the interior of a material creates shadows. The shadowing property is defined by assigning suitable values to the VFilter:Transparency channel in the Volume filtering shader.

The VSL hierarchy is:



- The rather complex computations in the Volume filtering shader take care of several details: they produce a shadow, whose intensity is proportional to the square of the distance which the light travels in the fog. The intensity is modified by some Noise and the shadow color is smoothly faded to white at edges.
- The rather complex computations in the Volume filtering shader take care of several details: they produce a shadow, whose intensity is proportional to the square of the distance which the light travels in the fog. The intensity is modified by some Noise and the shadow color is smoothly faded to white at edges.



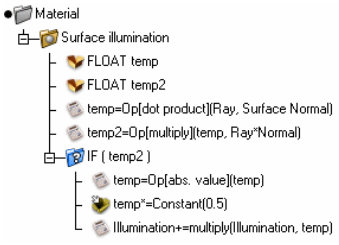
- The shadow in the example project is quite subtle and it is hard to tell if it is really the interior and not the surface, which creates the shadow. The difference becomes important in animations where objects and/or light sources are inside the fog. In other situations, it may be a better solution to create the shadow on the surface, because volumetric filtering is quite a time consuming feature. Especially, when both volumetric illumination and volumetric filtering are activated (like in the example), lots of computations are required, because the interior of the fog creates shadows in the interior of the fog. Therefore, the rendering time required by volumetric effects becomes proportional to the square of the sampling density - a 10 times higher sampling rate takes 100 times longer to render.
- Volumetric effects, Lighting in volume and Volume shadows in the Render settings/Ray tracing options must be enabled when rendering this example.

Example file: tutorprojects/material/vsl/shadowbyfog

Illumination Examples

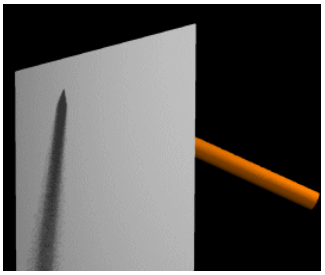
Paper

It is possible to define custom surface illumination effects by using a Surface illumination shader. The following material; paper, has the property that lighting (including shadows) is visible through the paper (to a certain extent).



The illumination shader computes the product of the dot products <ray*normal> and <lightray*normal>. This product is negative only when the surface is viewed from the opposite side of lighting. The If object notices this and computes illumination in those cases where default shading would leave surface without illumination.

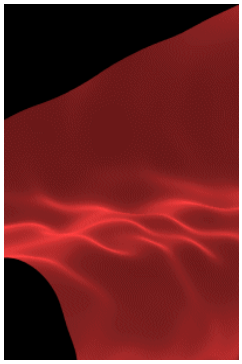
The alternate shading actions are very simple: the lighting intensity is modified by the lighting angle and then reduced by 50 %, because all the light cannot penetrate the paper.

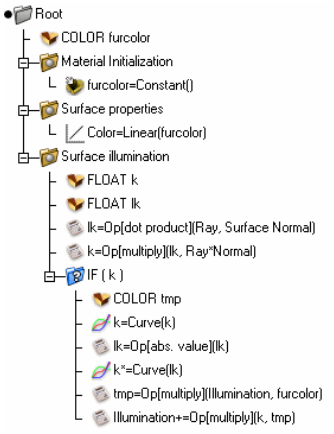


Example file: tutorprojects/material/vsl/paper

Fur

This example material simulates a surface covered by dense, short fur that shines in light.





- The Surface properties shader includes only one object, which adds a low amount of diffuse color by scaling down the furcolor value defined in the Material Initialization shader.
- The actual trick is made in the Surface illumination shader: illumination is defined so that it reaches the maximum value when the small hairs are viewed from the side. This behavior is quite different from the usual diffuse shading.
- The If object tests, that light does not get through the surface.
- The two curve objects define the amount of illumination; by modifying these curves, surface appearance can be adjusted.

Example file: tutorprojects/material/vsl/shinyfur*Example file:* tutorprojects/material/vsl/silk

Matte shadows

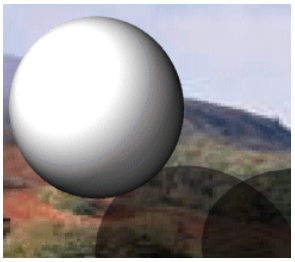
In the Surface illumination shader, both the original light properties of a light source and the amount of light reaching the examined surface point are available. It is possible to compare these two channels for creating advanced matte effects. In the example below, a photographed backdrop is not illuminated by the light sources (the photograph already includes illumination), but 3D objects still cast shadows on it.

1. Create the following VSL material:



2. Create a camera object using the camera tool.

3. Create a backdrop object using the backdrop tool. Use the material defined above.



4. Put some objects in front of the backdrop, and some light sources behind objects, so that objects cast shadows to the backdrop.

5. Take the camera view (View's popup Camera/Object View) and render.

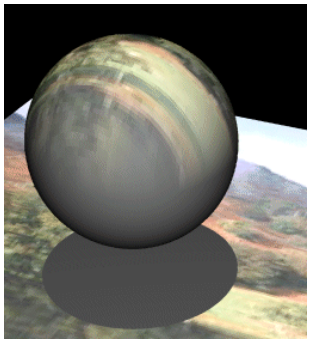
Some notes:

- In the Surface properties shader, the color of the backdrop is assigned directly to the illumination channel. The result will show the color in its unmodified state. Diffuse color is set to black to eliminate other shading effects.
- The Surface illumination shader first computes the amount of shadow. Negative values are truncated to zero, and the Surface:Illumination channel is scaled by the resulting factor.
- This example shows a situation where the backdrop image is in a digital form. We could also store the amount of shadow into the alpha channel. This would allow generation of shadows later to any background video sequence.

It is also possible to map reflecting color to the areas where the backdrop image is supposed to reflect the environment and so on - the possibilities are endless.

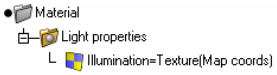
Example file: tutorprojects/material/vsl/matteshadows
Example file: tutorprojects/material/vsl/alphashadows

Slide projector



1. Create a light source.

2. Then apply the following material to it (for example using a parallel mapping):



Example file: tutorprojects/material/vsl/slideprojector



Note

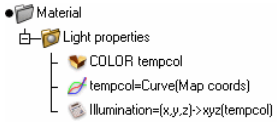
The mapped material defines light properties in the examined illuminated surface point, not on the surface of the mapping object. This means that the parallel projection used in the example above works best with a distant light source, because both project their influence in a similar way. A pyramid mapping matches with a point light source.

Custom light falloff rate

This example is recommended for those users who are familiar with polar space coordinates.

1. Create a point light source.

2. Then apply the following material to it using a spherical mapping. The origin point of the mapping should match the position of the light source.



- The three curves of the curve shader control the light falloff with respect to polar coordinates. In other words, it is possible to draw the shape of the light beam created by the light source:
- The red curve controls the first polar parameter, heading. This may be constant in which case the light distribution is symmetric. By making peaks to it, light source may send more light up, to the left, etc.
- The green curve controls the pitching parameter. Zero pitching corresponds the direction of the spot axis, a pitching of 0.5 is 180 degrees from the axis. Curve values greater than 0.5 control directions behind the light source. For example: if the green curve has a high value in the beginning, the spotlight will have high intensity in the middle. By changing this parameter, it is possible to make the light source cast a light ring, for example.
- The blue curve is perhaps the most useful one: it defines the falloff by distance. Usually this curve decreases down to zero. The size of the sphere mapping defines the corresponding area in space. By resizing it, it is possible to adjust the area of influence of the light source.
- A more intuitive way to control the shape of a light beam is based on a slide projector idea. Draw two gray scale images, which show the shape and falloff of the light beam. Then map them to the light source from the side and from the top using a parallel mapping.

Example file: tutorprojects/material/vsl/curvelight

Gamma ray light sources

This example shows how to create a light source that casts special kind of light that illuminates only certain kind of objects.



If object tests if the gamma channel value is inside 0.01 and 1

1. Create a new channel of type Float using the New popup menu of the Channels tab of the select window. Rename it as gamma.

2. Create a light source and map the material of the upper image to it.

3. Create an object and map the material shown in the lower image to it (any mapping type is suitable):



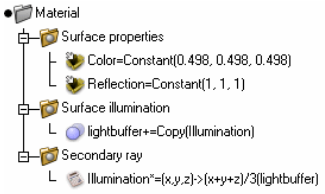
The Constant object assigns a value between 0.01 and 1 to the gamma channel

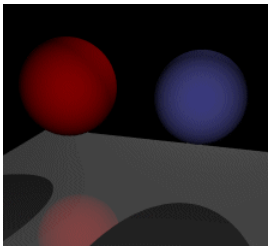
Only objects with gamma material become illuminated by gamma light.

Example file: tutorprojects/material/vsl/gammalight

Reflective in light

Light can define surface properties. For example, a light source can act as an X-ray radiator, which makes objects transparent. Or, like the example below shows, lighting can control the reflectivity of a surface.





Lightbuffer is a user defined color channel. The amount of light arriving from light sources is summed into it.

In the Secondary ray shader, the illumination of the traced reflection ray is scaled by the total sum of light.

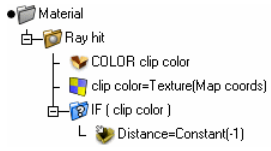
Example file: tutorprojects/material/vsl/lightmakesreflective

Miscellaneous Examples

Clip mapping

Clip mapping is the effect where a material removes the underlying surface. The following example demonstrates this:

The If object test can be, for example, a test if the red signal is dark enough:




```
X inside (0, 0.5)
Y always
Z always
```

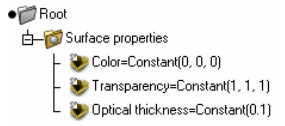
Setting the Distance channel to a negative value removes the surface. There is a VSL wizard available for clip mapping, as well as a template in the Image Maps folder.

Example file: tutorprojects/material/vsl/clipmap

Simple glass

This is the structure of a very simple glass-like material:

- Color is set to black. This removes diffuse illumination, which would otherwise practically hide the transparency of the material.
- Transparency is set to white (1,1,1). All light gets through the surface.
- Refraction is defined by optical thickness channel, 0.1 is suitable for glass.

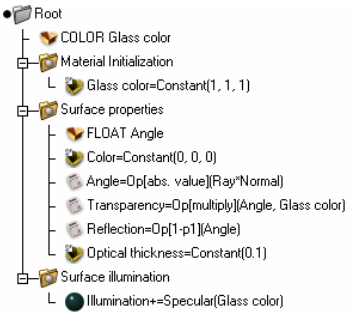


Example file: tutorprojects/material/vsl/simpleglass

Realistic glass

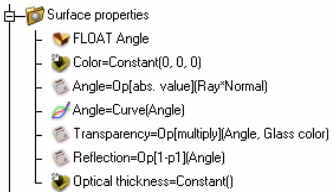
The glass material of the image is created by the Glass VSL wizard. It produces realistic looking glass.

- Glass color is defined as a constant, which is initialized in the Material initialization shader. Other needed colors are consistently computed from this variable.
- In the Surface properties shader, transparency is defined as a function of the angle in which the ray hits the surface. Perpendicular rays penetrate the surface well, whereas rays that have the direction of the surface are reflected. The channel ray*normal measures surface-ray angle: the channel value is +1 or -1 for orthogonal rays and approaches zero as the angle gets smaller.



Possible improvements:

If accurate reflectivity control is required, a curve object can be added to surface properties. By adjusting the curve, surface reflectivity can be easily modified as a function of angle.

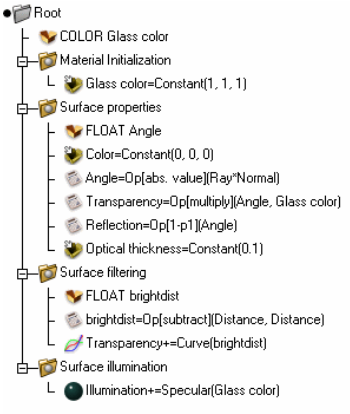


The Surface filtering shader can be used to control the glass shadow creation. The adjacent image shows an example. Here the difference

```
Op[subtract](light:distance, filter:distance)
```

measures the distance behind the glass. The curve object then adds some brightness according to this distance, so that there is actually a focal distance where the effect appears brightest. Finally, if the 3 curves that modify transparency are not identical, a colorful spectrum shadow is created!

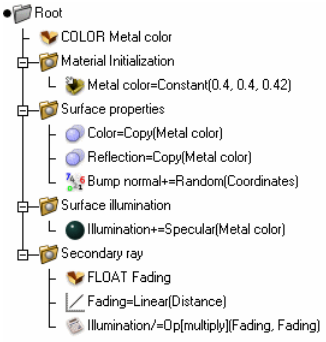
Example file: tutorprojects/material/vsl/spectralglass



Steel

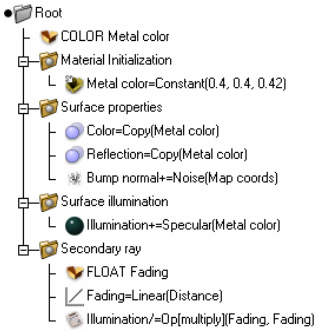
In this example, the VSL structure of a metal created by the steel wizard is examined;

- The Random object that modifies the bump normal channel makes the surface appear rough. General/Normalize option should be set. The Random object's distribution should be zero-centered: base = -0.5*amplitude.
- Reflections on the metal surface are faded by distance using a Secondary ray shader. First, the distance of the reflection is modified by a simple linear transformation, then this value is squared to obtain faster, non-linear falloff rate for fading, and finally the color of reflection (found in traced ray:illumination) is divided by the value.



Example file: tutorprojects/material/vsl/steel

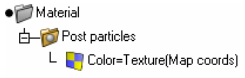
By replacing the Random object with a Noise object, the rough appearance of the previous material can be changed to a brushed metal appearance. The brush effect requires that the mapping used defines one dimension that is strongly compressed. The cylinder mapping axis is made very short for this purpose in the example project.



Example file: tutorprojects/material/vsl/brushedsteel

Texture mapped post processing particles

The color of particles can be defined using a VSL material:



1. Create a set of 1D particles.
2. Map the particle disks post effect to the particles with a default map.
3. Map the material shown on the right to them with a parallel map

A Post particles shader defines properties for particles that are rendered using a post effect such as Lens flare or Particle disks. Properties for Scan line type particles are defined using a Surface properties shader.

Example file: tutorprojects/material/vsl/texturedparticles

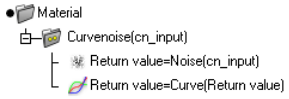
User defined VSL procedures

The program includes tools for creating new user defined VSL objects. Frequently needed algorithms can be collected into a VSL procedure library. The procedures can be called from VSL materials. For example:

1. Create a new procedure library by using the menu New/VSL Procedure Library of the material tab of the select window. Open the Properties window. The created procedure library contents are displayed.
2. Drop a Procedure object to the Root node of the VSL tree.
3. Select the created Procedure() item. Rename it as Curvenoise using the Name field on the bottom of the VSL window.
4. Change to the Parameters tab and press Add. A new parameter appears to the previously empty parameter list. Rename the parameter, for example, as cn_input using the Parameter name field.

5. Drop a Noise object to the Curvenoise procedure. Change the input parameter of Noise to cn_input. Output should be 'Return value'.

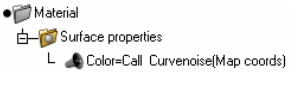
6. Drop a Curve object to the Curvenoise object. Both output and input parameter should be 'Return value'. Modify the curve a bit. The procedure is now ready. Its structure is shown in the image.



7. Activate the select window and create a new VSL material. Set Advanced option to see its contents on the property window.

8. Drop a shader to the VSL tree of the new material. The default, Surface properties, is suitable.

   Drop a Call object to the surface properties shader and select it. Pick Curvenoise from the displayed list of available procedures. The default input and output parameters are suitable. They can be changed using the VSL editor's popup menu.

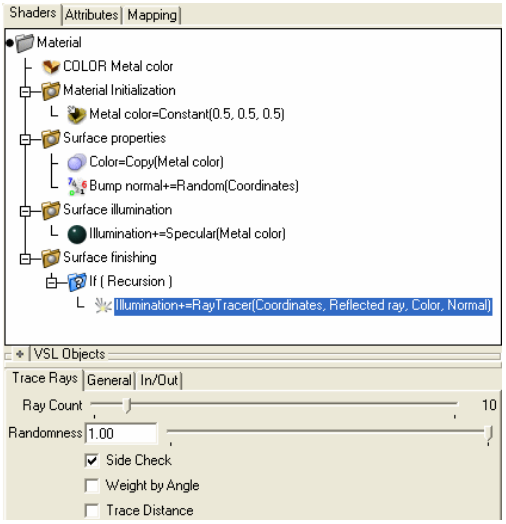


9. Map the material to a test object using a parallel mapping and render.

Example file: tutorprojects/material/vsl/vslprocedure

Aluminium with blurred reflections

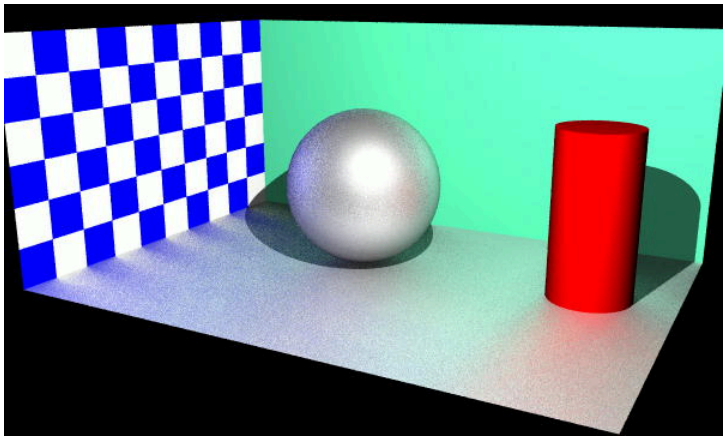
Customized ray tracing effects can be generated using the Raytrace VSL object. The aluminum like material, whose tructure is shown in the image, generates strongly blurred reflections.



Blurring is created by a Raytrace object in a Surface finishing shader. The Raytrace object traces 5 random rays around the reflection direction. Each sample direction is strongly randomized. The rendered image shows how reflections remain recognizeable only at a very close range from the aluminium surface.

The image quality depends heavily on the amount of traced samples. It is easy to adjust the amount of traced rays, but high values should be used with care. If the scene contains aluminium surfaces that reflect each other, computation task (and hence rendering time) increases very rapidly when sampling rate grows. Already at a recursion depth 3 and sampling rate 10 one single aluminium surface evaluation can generate 10*10*10 = 1000 ray trace operations!

Both recursion depth and recursion threshold should then be adjusted to keep the rendering time tolerable. The material itself can also define a local recursion limit: the Raytracer object can be placed inside a If level which tests that recursion depth is less than 1.01.



It is also possible to define a post processing configuration which filters away the noise of a shading component before adding it to the image. For details, see the example about post processed illumination in the Rendering / Advanced Illumination Examples chapter of the manual.

Example file: tutorprojects/material/vsl/blurredaluminium

Type Casting

There are three kinds of channels and variables available: color, float and vector. Most of the VSL objects repeat the same calculation to all sub channels independently. The user does not have to worry about the data type differences when building VSL code. VSL automatically changes data types to a similar form so that the computation can be performed.

For example, in a VSL expression 'color = wave(coordinates)' both input and output channel are 3 dimensional. Therefore, the computation goes as:

color.red = sin(coordinates.x)

color.green = sin(coordinates.y)

color.blue = sin(coordinates.z)

The expression 'color = wave(distance)' (distance is one dimensional value) is computed as:

color.red = sin(distance)

color.green = sin(distance)

color.blue = sin(distance)

The expression 'alpha = wave(distance)' (both alpha and distance are one dimensional) is simply:

alpha = sin(distance)

and 'alpha = wave(coordinates)' goes as:

alpha = sin(coordinates.x)

Similarly, if the object, by its nature, always outputs a single floating point value, the result is copied to all output sub channels. So, 'ray = length(normal)' computes:

ray.x = length(normal)

ray.y = length(normal)

ray.z = length(normal)

Parameters

The number of input parameters of VSL objects varies. 'Constant' has none, whereas 'Linear interpolation' of 'Operation' object has three. The parameters can be set either by selecting the object and using the popup menu, or from the in/out tab of the VSL editor.

An object can have only one output parameter. Nevertheless, VSL objects can also change values of their input parameters. Because of automatic type conversions, the original address of a parameter is not always passed in a VSL object call. Instead, the system copies the (possibly modified) converted value back after the execution to allow multiple output channels.

This principle is important to know when passing a parameter several times to a shader that also modifiers several parameters. The final value is not defined by the order the object uses them internally, but by the order they are given in the parameter list. For example:

```
color = SpecialShaderThatChanges4Channels(color, color, color);
```

The actual output channel value assignment is made last and is most significant. After that, the assignment of the last (3rd) input parameter has second highest significance.

This should not be a problem because the order would be unknown anyway without explicit instructions; developers of new VSL objects should assume and follow the above defined convention.

Assigning Multiple Materials to SDS Objects

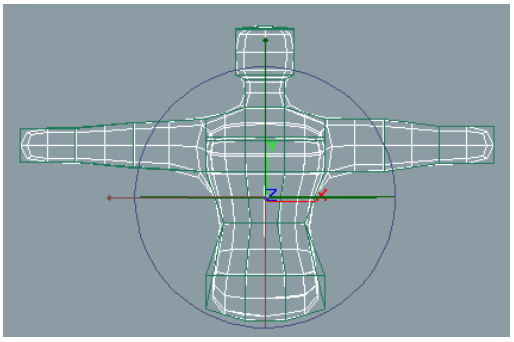
This example explains how to map materials to a subdivision object. Mapping is defined so that material patterns follow surface deformations (bending of skeletally controlled human figure, for example). You will also learn how to restrict each material to affect only desired parts of the surface.

Level: Medium/Advanced

Sample files: 'tutorprojects/material/sdsmaterial/torso' and 'tutorprojects/material/sdsmaterial/sdsmat'

First, model a simple human figure, like the upper body shape of the example image, which was created by extruding the head and two arms out from a 4*4*4 subdivision cube. Rename the figure as 'Torso'.

We will assign two materials, wood and brushed steel (both include color and bump patterns) to the torso. Therefore, we have to attach suitable mapping coordinates to the object (the patterns will follow the assigned coordinates). Both the wood and steel material are designed to work best with parallel mapped coordinates. Therefore, make sure that the torso is selected, go to the materials tab of the select window, drag and drop the wood material of the default material library into the view window. This creates a suitable parallel mapping. Ray trace the view and you will see a wooden figure.

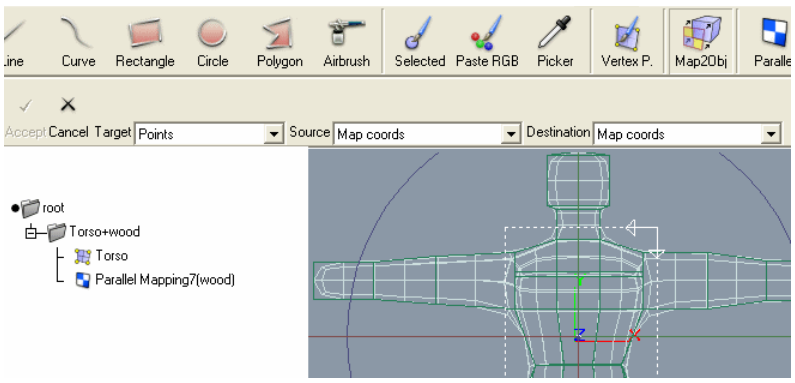


Subdivision modeled torso

Go back to the geometric objects tab of the select window. Open the hierarchy of the created torso+wood level. You can now adjust the density of the wood pattern by selecting the parallel mapping object and by applying the usual Transformation/Scale tool. The smaller the mapping object, the denser the wooden pattern.

This kind of wood mapping works fine for rigid objects. However, the strictly geometric parallel mapping cannot follow arbitrary flexible deformations. For example, if the arm of the torso were now bent, the arm would move across wood patterns, which looks strange in animations. Therefore, we have to convert geometrically defined parallel coordinates to pointwise defined parallel coordinates.

To do this, select the torso object and the parallel mapping from the select window. Go to the Materials tab of the toolbar and click the Map2Obj tool icon. The tool controls display various options: Target=points and Source=Map coords should already be selected by default, but change Destination to Map coords. Then click the Accept button.



Map2Obj tool stores mapping coordinates to surface points

You may now delete the parallel mapping object from the scene. Then select the parent level of the torso. Click the Default mapping tool icon on the toolbar (it is included in the same Materials section of the toolbar as the Map2Obj tool explained above). Make sure that the tool controls display Wood as the selected material and change Mapping type to Null. Then click Accept.

It you render the scene, the wooden torso should look pretty much the same as before. However, now the mapping coordinates follow nicely with all kinds of deformations. We used Null mapping above, because the surface itself computes mapping coordinates from the stored pointwise information and the default mapping should not overwrite this by some other coordinates.

Defining the influence area of a material using 3D painting

The next task is to limit the area of the wooden surface to some sub area of the torso. For example, let's try to write the text 'Wood' to the chest of the torso with wooden letters, the rest being the usual default material.

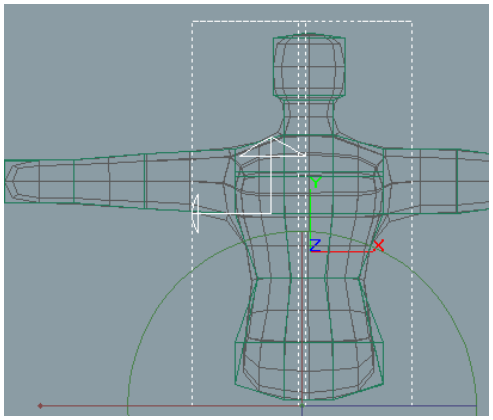
For this purpose, we need another set of pointwise coordinates. A nice way to define the text area is to use 3D painting. This method requires two-dimensional coordinates, which run along the surface. We can define these coordinates using the Map2Obj tool with a cylinder mapping.

Therefore, select the parent level of the torso, take a top view and pick Cylinder mapping from the toolbar's Materials section. Click in the middle of the torso, and shape the cylinder around the torso. The diameter is not important, but the seam direction sometimes is, so click a second time behind the torso to finish the cylinder mapping. The used material is not important, because this is just a temporary step.

Take a front view. Select the created cylinder mapping, extend and move it so that it covers the torso in the height direction (use Transformation/Move and Transformation/Extend tools or object handles to do this).

Then select the torso and the cylinder mapping and activate the Map2Obj tool. Target=Points and Source=Map coords are again suitable, but set Destination to UV coords. Then click the Accept button. Now the torso has parallel mapped pointwise mapping coordinates and cylinder mapped pointwise UV coordinates. Delete the cylinder map object.

The cylinder map does not always unwrap the surface of a target mesh properly. For example, if the arms of the torso were bent, some surface areas might get badly folded over other areas in unwrapped UV space. If this happens, the UV editor can be used to fix problematic areas.



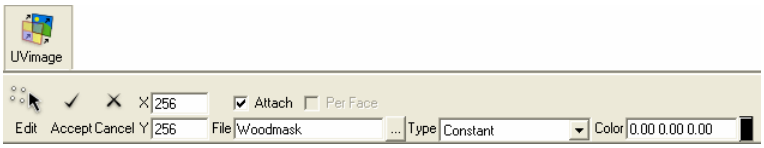
Cylinder mapping defines uv coordinates for surface points

You can find the UV editor from the Windows menu. By comparing point selection on a normal view window and on the UV editor, it is easy to figure out how the cylinder map unwrapped the surface. Note that points can be selected on the normal view and moved on the UV editor.

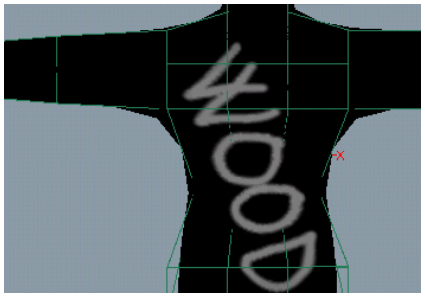
We will assign materials to the middle chest of the torso, which is a neatly unwrapped area, so UV point editing is not necessary this time. If the purpose was to paint the arms, then some editing would be necessary, because the arms cover a very small area of the UV space and 3D painting is therefore inaccurate there. In a worst case, only a couple of pixels of the used texture map might cover the arms.

Next we define the wooden area. Select the torso and activate the UVimage tool (it is located in the Materials tab of the toolbar). This tool is for creating and assigning a bitmap texture to an object. The activated tool displays several new controls. The X and Y fields display the resolution of the image to be created. Default x=128, y=128 is a somewhat low resolution, so change it to x=256, y=256. Type=Constant is suitable; this means that the image will be initialized to a given color. Set the constant color to black by typing 0 0 0 to the Color field or by using the color selector which opens by clicking the small sample color square. The next option, Attach should also be set, so

that we will see the image wrapped on the object as a color texture. Finally, enter a suitable File name, for example 'woodmask'. Then press the Accept button.



Turn shaded OpenGL on if not yet selected (you can find this option for example, from the view compass). The created texture map makes the object appear totally black. Activate Airbrush tool from the toolbar's Materials tab. Pick a medium size brush from the airbrush controls (for example 4 units wide brush) and set pure white color 1,1,1,1 into the color field. Then take a suitable look at the chest of the torso and draw the letters 'Wood' over the body. The white text becomes visible on the chest. Then press Accept to finish using the tool.



3D painted mask image

Now the mask image for the wooden area is ready and we will convert it from a color texture map to another form. Make sure that the torso is the selected object and select Object properties from the view window's popup menu. Select Col tab from the property window and empty the Texture field displaying the name 'woodmask.bmp'. Close the property window.

Building the materials

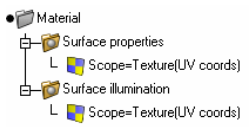
The next step is to create some materials. We could use easy to use templates for this purpose, but let's practice using the Visual Shading language a bit and build the materials step by step.

Go to the select window, switch to the material tab and pick New/VSL material from the select window's popup menu. A blank material appears in the material library. Double click the icon of the new material to open the property window. When the property window appears, first change the name appearing at the top to 'woodmask'. Enable the Advanced option and also open the Available objects frame, in case it was closed.

The wood material defines both surface properties and some illumination properties. Therefore we need to define the same properties for the mask material controlling the strength of wood. Drag and drop the Shader object twice onto the top most Material level node of the VSL hierarchy. Then activate the lower shader and change the Type field from Surface properties to Surface illumination.

Drag&drop a Texture object from available objects onto the Surface properties node. Open the popup menu on top of the created texture node and set Output to Surface:Scope and Input0 to Surface:UV coords. The scope channel is a special channel that is used to control the effect of other channels. The selected input channel UV channel, is the same point assigned channel that we used to bind the wood pattern to the torso. Now we use the same channel to bind the mask pattern to the model. Define the texture name woodmask.bmp to the File gadget at the lowest part of the property window (use the browse button labeled as '...' if you are not sure about the file name or path). Then enable the Tile X and Tile Y options. The mask image is wrapped only once over the torso (one single tile), but the UV coordinates must usually run continuously over the seams of a cylindrical surface like the torso. Tiling improves the continuity.

Also activate the Gradient X and Gradient Y options. The airbrush tool already created smooth edges to the painted area, but the gradient option improves the quality even more, so that the wooden area becomes smoothly blended to the background.



The material defining the influence of wood

Finally, select the texture object, then select Copy from the popup menu. Select the shader node Surface illumination and select Paste from the popup. The mask material is now ready and looks like in the adjacent image.

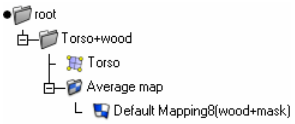
Go to the select window and select New/Level material from the material tab's popup menu. Rename the new material as 'Wood+mask' using the property window. The property window displays the available materials on the upper area and the contents of the level material on the lower area. Drag and drop the Wood and Woodmask materials from the upper material list onto the 'Wood +mask' node on the lower area. The result is shown in the exampel image.



A level material consisting of two sub materials

Average mapping

The next step is mapping the 'wood+mask' material to the torso. Change the select window to show the object hierarchy. Select the Wood default map object. Select property window's Spec tab. The top most gadget, Material, displays the target material of the mapping. Change it from Wood to Wood+mask level material. Then activate the Average map tool from the right end of the toolbar's Materials section and click Accept. This creates a new hierarchy level containing the default map object.



The hierarchy of the average map

The new Average map level is actually a special mapping object that averages the effect of materials mapped by its child mapping objects. Each material has a weight defined by the scope channel. If only one material is applied, an averaging object is not necessary, but we will also add some steel areas and the average map object makes sure that steel and wood are treated similarly. The following sample equation shows the difference between using and not using the average map, in a point where both wood and steel have scope 0.5:

Average map: result = 0.5*wood + 0.5*steel

Direct mapping: result = 0.5*(0.5*wood+0.5*default) + 0.5*steel

Without Average map, the blending of a new material to the previous situation is repeated until all materials have been evaluated and therefore the first material will have a weaker effect than the last material.

Close the property window and save the project with a suitable name. This step is important because project saving stores the changes in the images, including airbrush painting to the woodmask.bmp. Render the view window. You should see wooden text on a white torso.



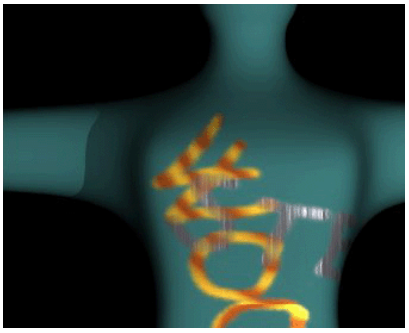
Masked wood area

Inserting more materials

Adding some steel areas happens exactly the same way. Before painting the steel areas, we have to protect the woodmask.bmp image temporarily, because 3D painting tools affect all bitmap textures hit by the brush. Therefore, select the Average map level object and open the object property window. Enable Invisible in photorealistic rendering option from the General tab. This makes the object in question inactive in all rendering actions, including 3D painting.

Next select the torso and use the UVimage tool to initialize a black bitmap called steelmask.bmp exactly the same way as woodmask.bmp above. Use white airbrush to paint the word 'Steel' to the torso. Note that the steel and wood texts may overlap without problems. When the steel mask is drawn and ready, go to property window's Col tab to remove the Texture field contents. The 'steelmask.bmp' was temporarily used as a color texture to see the painting on the surface, but from now on it will be used for masking the steel area. Use the color gadget to change the color of the torso to for example, darkish green, so that the mapped materials will be more clearly visible.

The easiest way to attach the masked steel material is to copy the already created wood items. Therefore, duplicate the material woodmask, rename the copy as steelmask and use the property window to examine its Advanced VSL hierarchy. Change the texture file name of the two texture VSL objects from woodmask.bmp to steelmask.bmp.



Blended materials

Create a new level material called 'steel+mask' and add the Brushsteel and steelmask materials into it (the same way as wood and woodmask above). Go to the object hierarchy tab of the select window and disable Invisible in photorealistic rendering property from the Average map level, so that the mapping starts working again in rendering. Make sure that the Average map level is selected and create a default mapping of type Null (see the beginning of this tutorial) referring to the new steel+mask material.

The tutorial project is now ready. Save it and render.

Modeling a Pot

Modeling the shape

Level: Beginner/Medium

This tutorial starts with creating a simple revolve, then texturing the result into a realistic pot. We make use of the Scope feature to show how to clip an image onto a surface that is already textured.



The Finished Pot

Start with a new project and select the front view, turn off perspective and reset the view.



Front View

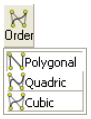


Parallel View



Reset View

In the NURBS tab, click the Curve icon. In the control bar select Polygonal curve by clicking the Order icon, and then select Polygonal from the drop down list. Leave all the options at their default settings - we want to draw an axis vertically that we can sweep our pot around. Start the curve at the lower middle of the view window; click once, then make another mouse click straight up the view and towards the top so the line is roughly vertical. (Holding the SHIFT key will keep your line vertical.)

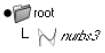


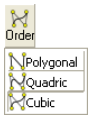
If you keep clicking at this point the curve can be made more complicated - but all we need is an axis so we will accept the curve at this point by pressing the Accept button in the toolbar (you can also right click and select accept from the pop-up menu). This finishes the creation of the curve.



Accept button

To simplify matters - it is always a good idea to name your objects. Under the select window, we can see that we have created a nurbs curve. By slowly double clicking the name next to the nurbs icon we can rename the object. Call it 'Axis'.





We now want to create the "Profile" of the pot. Think of this as a section through half of the pot that is 'swept' around our axis. Firstly though, we must select our axis - make sure it is highlighted before continuing. Under our NURBS tab again, click the Curve tool and select Cubic this time from the Order drop down list. (Polygonal curves have sharp edges - quadric and cubic curves are very smooth.)

We are going to use the axis curve we created as a point to rotate about when we make our profile. To do this, we need to select "Rotate" under our toolbar - click the Pen option to see the different methods - select Rotate.

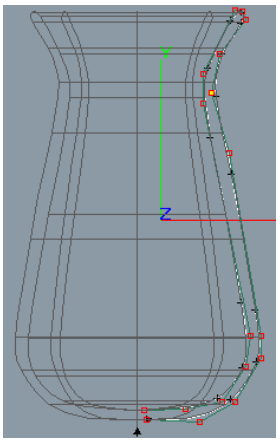


Starting at right hand side of our axis, click points from the base upwards to create a profile curve similar to the one shown. Each red point represents a mouse click. To get sharp edges in a nurbs curve, we need to put the points closer together. Hence the top has three closely grouped points. Note too that the curve draws both the outside, then goes inside the pot as well.

Do not worry too much about how accurately you place the points - you can edit them later. Notice too how you can see the creation of the revolve.

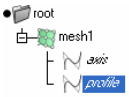
When you have finished creating all the points, click the Accept button again to finish creating the profile.

✓ *Accept*



The shape of your pot profile

Notice now that our level structure has changed and we now have a mesh in the view. We created the mesh by rotating our profile curve about the axis. The two curves are now put 'under' the mesh. Press the "+" icon next to the mesh to see the curves below. By editing these curves now will change the shape of our resultant mesh. Rename our nurbs curve to 'Profile'. Note that a mesh is a 3D surface - so when we render the image this is what we will see. Generally, curves by themselves are not rendered.



Rename the mesh to 'Pot'. Let's tweak the profile curve a bit. Select the profile curve, under the context sensitive toolbar a series of options are shown that we can do to this particular curve. Click the Edit button and we will move some points around to create the shape we want.



Edit

If you move the mouse cursor over any single point, the cursor changes to a cross. This means that you can now click on the point and move it around. Do this for each point until it looks roughly like the above image and has an even space between the inside and outside. You can now pan around the view and examine the pot you have created.



To get a better feel for the shape - turn on Open GL view to see the mesh better.





You can also raytrace the view by pressing teh render button in the view control bar.

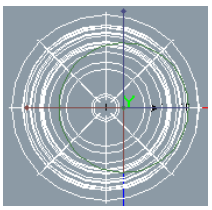
This is a good point to save your work. Under the File pull down menu, select Save As. Select the appropriate directory by clicking the '...' button next to the name field and Save the file as Pot_Part1.

Texturing our pot

Texturing a model is about applying materials to an object to achieve a given look and feel. These materials can be as simple as colors, or as complex as applying bumps, reflectivity, color and shine.

We can very simply drag and drop the standard glass material onto our pot to create a glass pot. Likewise, we can make it gold, plastic or chrome by using many of the standard materials.

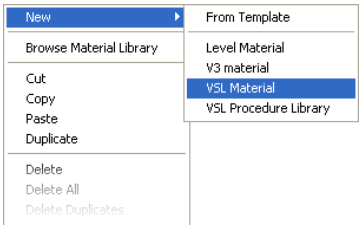
But we want the pot to use an image. This image could be a drawing, a color or a photograph. We will map this image onto our pot using the Cylindrical mapping method. We have many methods to apply the image to a model - parallel being the most obvious - it projects the image onto the model in the size and aspect you set when applying the material. Cylindrical mapping works on the same principle - but it wraps the image around the cylinder walls - much like the word COKE is written around a can. We can wrap 1 image right around the object, or we can make it Tile around. If I wanted COKE to appear twice, I would make it tile 2 times around.



In this example we are going to map a simple strip of color around our pot to make it look real. To do this we need to make this Strip tile about the pot 10 times. Here's how we do it:

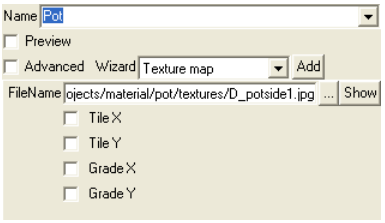
Load Pot_Part1 tutorial if its not already active (it is located at: tutorprojects/material/pot/Pot_Part1.r3d). Select the TOP view. Drag the Pot object mesh from the Select window and into the view window. This maximizes the object in the view window. (you can do this for any object including a camera.)

We should now have a wireframe view of the object like in the example image at the right.

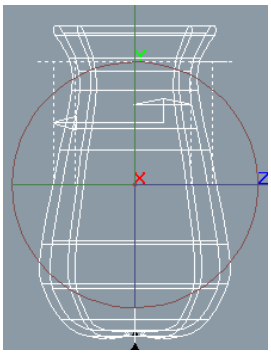


Let's go to the Materials tab: In the select window, select the Materials tab. Let's purge any unnecessary materials from this project - under the pull down menus, select Materials/Purge material Library. This deletes any UNUSED materials. If a material is being used, it is kept. This should delete all the materials as we have not assigned any yet.

Right Clicking in the blank area inside the material window should bring up our pop up menu. Select New/VSL material. This creates a new blank material for us to edit.



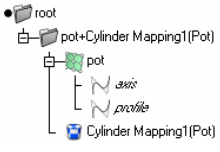
Select the new material and rename it (by slowly double clicking) to 'Pot'. Now quickly double click the material. This should bring up a new window called Property Window. This shows us all the selected objects properties. In this case, it shows us the properties for our material Pot which happens to be empty. Let's fix this by using one of the pull down wizards. Select Texture Map from the wizard pulldown, Add this by pressing the ADD button next to the wizard. This inserts a file requester. Let's grab our texture map image by clicking the '...' button. (Next to the SHOW button) Navigate to our tutorprojects/material/pot/textures directory, and select D_potside1.jpg.



Let's map this material to our pot. In our material window, right click on the Pot material and select Map/Cylinder. This activates a tool to cylindrically map our texture to our model. It automatically assigns our material to whatever object is selected in the select window. Which should be our Pot Mesh as it is the only object.

To create our cylindrical mapping, first click at the center of the mesh, then just outside of the mesh. The result cylinder should just encompass our pot at its widest point. Rotate the view and see the result. Click the Side view. It should look similar to this image.

Note that the cylinder mapping is shown dotted. It is also smaller than our pot. We need the cylinder mapping to be just a bit taller than our pot. Let's size the mapping to fit the pot:



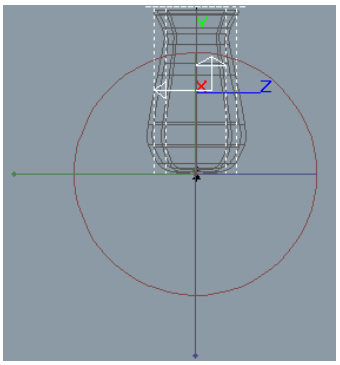
Go to the select window and examine what has happened to our structure. Our mesh has been dropped to a new level. Click the '+' icon next to the level to maximize its contents.

Note how the mapping is level vertically with the mesh. This mapping is now applied to anything inside this level - but nothing outside of it. Select the Cylinder Mapping(Pot). This selects just the mapping object. Let's use the handles of the object to position it correctly.

Put the mouse cursor over the blue line of the object handle - the cursor should change to a Move icon. Click the mouse on the blue line and move the object to just below our pot. This puts our object handle out of view. If needed, Pan the view so you can see the blue dot at the end of the object handles.



Now put the cursor on this blue dot at the bottom of our object handle - it should change to a pencil icon - this means we can scale the object about this axis. Click and drag down to scale our mapping object larger vertically. Keep modifying the object until the mapping is just larger than the pot. Note how the scaling or rotating of an object occurs about the handle, hence it scaled upward. If it was positioned in the center of the object, it would be scaled in both up and down directions. Our mapping should look like in the adjacent image.



Render the view to see the result. As you can see - we have applied our strip of color around the pot but it looks awfully dark! Let's add some lights next.



Go to Light Sources under the main toolbar to add some Point lightsources: Select Point - and click on the view for the light's center point, then another click to define how far the light should go.



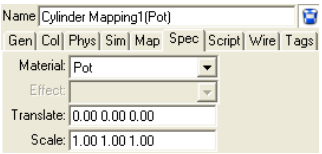
Point light tool

Add 2 or 3 lights around the pot. Render the view to see your progress and move the lights if needed. (Check your top and side views for light positions - move the lights by moving the mouse cursor over them and dragging).



Above is the image that we have stretched around the pot. (We have applied it vertically) It is very narrow and this results in a very blocky looking render of the pot. What we can do is tile this image around the pot a number of times. We change this by editing the Scale parameter of our mapping object;

Select the Cylinder mapping object, and click the Spec tab in our Property Window. (Remember we brought this window up by quickly double clicking any object). There are three numbers in the Scale field: '1 1 1'. This means that the mapping is mapped once in each three directions (X Y Z). If we want our strip to be repeated 10 times around, we actually need to make it 1/10th of the times in the X axis. Which means we need to punch in '0.1 1 1' There needs to be a space between each value as each value represents the X Y Z value. Try rendering the pot now and see what happens.



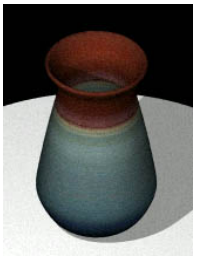


Our pot has only a strip of the image applied now! This is not what we want - it should be tiling around the object. We set this parameter in the material itself. Go back to our material tab and select out Pot material. In our Property window, there is a checkbox for Tile X. Turn this on and render again. Much better! We have now tiled the image around our pot 10 times.

Just for the sake of making our scene a little more interesting, Let's add a ground plane. In this case, let's put a disk underneath the pot so we can see some shadows. Under our Analytic tab, select the Disk icon. In a Top view, click the center point first (Center of the pot) then another point further out, well away from the pot. In a side view, move the disk to just below the pot.

Save your work as Pot_Part2.

Scope mapping



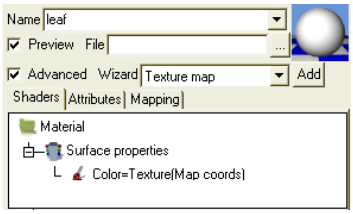
We have created a very basic pot with a simple cylindrical texture applied. We will continue fine tuning our material here and also map on a 'leaf' to the pot that shows some of the power of VSL.

Load "tutorprojects/material/pot/Pot_Part2.r3d". This scene has some lights and a ground applied to show off our pot better.

Let's continue to tweak our material and give the pot a few more realistic attributes.

Go to the Materials tab in the select window and select our Pot material. Double click the material if the Property window is not already up.

Let's use another wizard and add some 'Ping' to our pot, using the Specular Color wizard - Add it too.



Try rendering the view to see the difference. It renders as if it is made from gold. We want a more subtle appearance - so change the specular values to read:

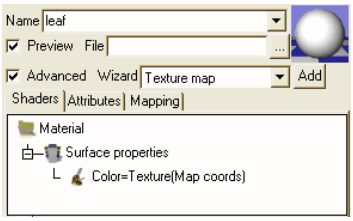
- Sharpness = 3.00
- Brightness = 0.80

We could go on to add bump and a bit of reflection if we wanted - but let's get started on a new material to map a leaf onto the pot.

Create a new material - right click in select material window and select New/VSL material.

Rename the material 'Leaf'. (We can do this by changing the name in the Property window, too.)

We have two modes when creating materials - Basic and Advanced. We are currently looking at the Basic mode in our property window. If we tick the Advanced button at the top left we can see exactly whats going on in this material. Let's do this.



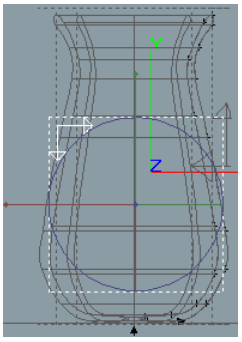
Use another Wizard and create a Texture Map. This creates a new surface property that is assigned to a texture. This texture is, by default, assigned to the color channel of an object. But we want it to be assigned to the Scope attribute in this case. The scope channel controls how much of a material is shown and how much is not. This is a powerful function that allows us to put any image onto an already mapped object (or unmapped) and clip out any undesirable background so we have only the desired portion shown on the image.

This has applications in assigning text over a material and removing the background so we only see the text. Assigning a complex window bitmap over a brick mapped wall, mapping a logo to a product, etc.

We cut our image out to do this - generally with an alpha channel (Extra information in an image that stores transparency data - typically Targa files are saved with an Alpha channel). We could also use a specific color value to do the same effect - but we have a Targa file with an alpha channel already setup.

Select our new texture map icon that has been placed in the properties window (red paintbrush icon). At the bottom of our properties window we see a file requester window similar to the basic version we used previously. Select the texture 'tutorprojects/material/pot/textures/D_potleaf+A2.tga' by pressing the '...' button.

If we were to map this texture onto our pot now, it would overwrite the material we created previously, so we need to setup our scope value so only where the leaf texture is shall be mapped onto our pot. Right click on the color=texture icon and select Copy, then Paste. This creates a duplicate of the texture we just assigned. Select the first texture map. Right click and under the Output menu select surface:scope. This changes the output from the texture map to go to the scope instead of the color.



And that is the basic structure of our scope material. Let's assign this to our pot and see what all this complicated VSL does. Go back to the select window. Click the front view and make our pot full screen (drag the mesh to the screen) Make sure that the pot level itself is selected and not the mesh as the material mapping will be put inside this level. Go to the material tab again and select our Leaf material.



Right click on the material and select Map/Parallel. Draw a rectangle roughly square - starting click at the top left and 2nd click bottom right. The map object is now inserted under our level as shown in the example image at the right. We can select the mapping object and scale/reposition it as necessary.

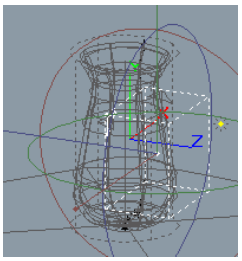
Rendering the view should yield a result similar to the adjacent image.

Make a few test renders around the object. Notice that we can move the leaf map around the object and it covers whatever is under it. But there is one small problem. The leaf appears on both sides of the pot! It is by default, mapped straight through and will continue going on forever. If we set the tile X + Y buttons on, it would repeat over the object as well. In this case, we only want 1 copy of the leaf, so we will set some attributes under the mapping object:

Select parallel mapping(leaf). Under the Spec tab in the properties window are three toggles - set them all ON - Finite X,Y,Z.



This limits the extent of our leaf material to within the boundaries of our mapping object. This is a very handy feature! Modify our parallel mapping object so it sits within one side of our pot and looks similar to the image at the right.

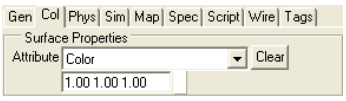


Finally, let's add a bump map to our material to give it a realistic raised brushstroke feel.

Go to the material tab in the select window. Select the leaf material and open the properties window. Using a wizard, add a bump map. In the name field, select the 'B_potleaf +A2.jpg'. Under the Bump Height field, enter '-1'.

Render your pot. Notice that all these materials have very subtle effects. To see what any of the parameters do - increase the values. (e.g. change the bump height to -5.)

To finish off our scene - let's change some of the values of the ground object without assigning a material to it. Go to the select window and select the Ground object. Under the Col tab of the properties window are various surface properties that we can change without the need to assign a material to. (A material would override these settings.)





Right click on the white square. Only one option appears: Expand. Click this. This brings up our color wheel. Select any color you like. Next to Attribute, Select the Reflection attribute. This uses a greyscale value to say how reflective an object is. Black is none, white is mirror. Select a dark grey.

Render the scene. The final result is shown in the example image.

Clip Mapping

Level: Intermediate

Clip mapping is the term used to describe using photographs or drawings as maps onto objects with an alpha channel. This creates the illusion of a detailed 3D object when in computer terms, all we have in the scene is one simple polygon.



On the left is our original image, on the right is the alpha channel that we use to "cut out" the horse.

The advantage is obvious - having a realistic looking render with a minimum of complexity in our scene. Clip mapping is particularly useful with objects such as people, trees, cars or animals. Any object that would have a high polygon count and does not need to be animated could become a good clip map instead.

There are three primary methods to assign a clip map in Realsoft 3D -

1. Use an alpha channel from a Targa file - this allows one to get various levels of subtle transparency. Targas have the advantage of being able to store the alpha channel in the same image file (32 bit = transparency channel, 24 bit = color only.)

2. Use a certain color as a clip channel. The solid BLACK background would be transparent in the tree show in the example image on the right side.



3. Use two separate images - one for color and one for the alpha channel.

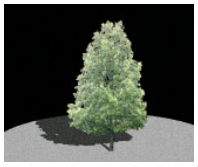
Color clipped Tree

Let's start with the most basic method first - color clip maps.

1. Create a new scene.

2. Create an analytical rectangle - any size.

3. Create a new material, use a template. Select Clipmap from the Image maps directory. Select the image "Tree1.jpg" from the tutorprojects\material\clipmap\textures directory. Map the material on the rectangle using a parallel mapping



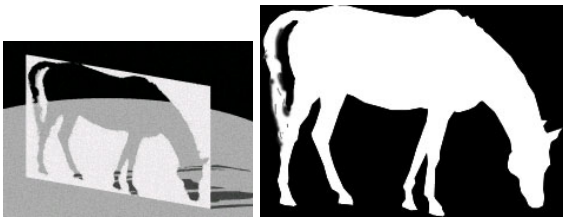
4. Render the image. At this point you can see that the default color range has used the black color to grey to define the transparency. This is probably too much, so let's select a darker color grey (just above black) for the Clip to field. Note that we have no color information on our object yet either - we have only the clip component.

5. Use the wizard to add a Texture map to our current clipmap texture. Select the "Tree1.jpg" as the color image and render. You should now see a tree that has been clipped out and will cast detailed shadows.

Fade mapping

Realsoft 3D uses the term Fade in VSL to describe the transparency component that is cut out from an object. Just as we use the color channel to put an image or color onto an object, whatever is mapped to the fade channel will be cut out from the object. It is an advanced form of trim curve if you like - allowing you to cut out detailed or arbitrary edges to an object, and even give it levels of transparency where only half a cut is achieved.

The advantages of using fade mapping is we can get a soft edge - or antialiased edges to the cut, giving it that added level of realism.

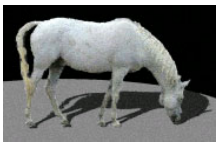


1. Create a new scene.

2. Create an analytical rectangle - any size. (Keep in mind the aspect ratio of the original image being mapped)

3. Create a new material, use a template. Select Fade map. Select the image "horse_alpha.jpg" from the tutorprojects\material\clipmap\textures directory. Drag this material onto the rectangle. Render the image.

Note how our horse is "back to front". we have mapped the image on the right to achieve the image on the left. Standard alpha uses white for solid and black for transparent. Realsoft 3D uses the opposite. Black is solid and White is transparent. Instead of going into photoshop and inverting the image, we can do this inside Realsoft 3D. In the materials Advanced tab :



4. Add a Constant above the fade texture. Change the Output of the constant (default color) to Surface:fade. Assign a value of 1 to the constant. On the fade texture map below our constant, change the operator to Subtract instead of Assign. This now takes the image alpha away from a constant of 1 or solid. This basically flips our alpha channel.

5. Render the image - our horse should be cut out correctly. But we still need to add the color information. Use the Texture map wizard and select "Horse.jpg". Render. You have just created a horse!

Targa clip maps

Targa image formats are the standard method for storing alpha information into one image. The procedure is exactly the same as the method outlined above. Only the same image is used for both color and alpha. When a Targa is selected as a Fade map, the Bindings tab must be set to use the alpha channel as the output to use for the fade channel. This is done by checking the Edit bindings button in our advanced VSL settings. Removing the "colorR -> Fade" binding, then selecting Alpha and Fade and Adding. This may not always be necessary - just check that the alpha is being used for the fade first. Often adding the texture first to a color map then changing the output to fade will set "alpha -> Fade" automatically. Inverting the alpha may depend on the source of the targa. If it is already inverted - you will not need to use the constant=1 value.

Scope mapping

Scope refers to how much of the current mapping will be mapped onto an object. We could map only text onto an object for example, by essentially clipping the background with our alpha and only the text will be mapped onto the object. But rather than actually CUTTING the object, we are cutting only the texture. This has powerful possibilities - especially for adding textures over already assigned textures! (e.g. adding windows over a brick wall.)

1. Create a new scene.

2. Create an analytical sphere - any size.

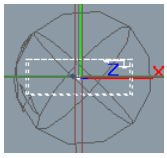
3. Create a new material, use a template. Select "color". Assign this material to the sphere by dragging it onto the object.

4. Create another new material, use a template. Select Scope map. Assign the texture "scope_alpha.jpg" to the scope field. The texture can be found in the tutorprojects\material\clipmap\textures directory. The adjacent image shows white text on a black background. This will give the white areas a scope of 1 and black areas a scope of 0.



5. Add a color gradient to our texture - use Texture map wizard and select "scope_coloring.jpg" from the tutorprojects\material\clipmap\textures directory.





6. Map this new texture onto our sphere using parallel mapping - make it roughly in the proportions to the image on the left.

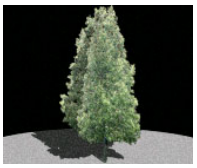
7. Render the image. We should have achieved a blue sphere (default color material) with the writing "scope mapping" written in color on the side of the sphere. Note how our color gradient was simply clipped by the scope mapping scope texture. We can also select Finite XYZ under the mappings Spec tab to restrict our texture from penetrating both sides of the sphere.

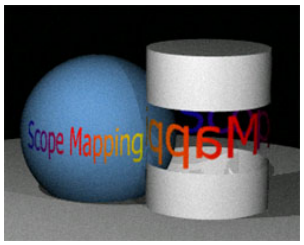
We can keep adding materials in this manner to get windows on walls, rust into crevices, writing on signs, etc.

Refer to "Adding Dirt" tutorial for further explanations in scope mapping.

Flat Image Vs 3D Object

One of the biggest challenges with any clip map is to have the image facing toward the camera at all times. We can tolerate a certain degree of movement - up to a point the illusion is ruined and the clip map is exposed for what it is - a single plane. We have certain "hacks" that we can utilize to give clip mapping an added degree of realism. One such example is to place the two planes at right angles to each other to form a "cross" in plane. Providing that the object (such as a tree) is roughly symmetrical, the illusion can be maintained from a 360 degree rotation.





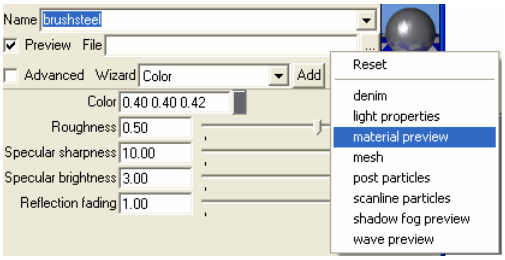
There are various mapping methods such as the Look at controller that can be used to make clip mapped objects follow the camera if needed.

But one need not be restricted to simple rectangular forms either - we can just as easily map a cylindrical map and clip out a 3D object.

Previewing Materials

When editing materials, you can preview the result by checking the Preview option of the property window. The preview window will be updated every time you change the material. The material icon is also automatically updated after each preview rendering.

The program uses a simple default scene for previewing the material. However, one single preview scene cannot show the characteristic properties of all possible materials. Therefore, the program provides an option for selecting the most suitable preview project for each material. The 'previews' folder contains a collection of predefined preview scenes. You can select the most suitable one using the Test Scene gadget. The name of the preview file is stored with the material when you save it to a disk file.

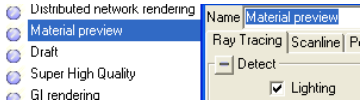


Selecting the preview scene

Note that the preview window is actually a small view window. You can zoom, rotate and pan it by dragging with the right mouse button Shift, Alt or Ctrl key held down. The window can be set back to the original scale and orientation using its Reset popup menu.

Changing rendering settings for the material preview window

The default collection of rendering settings contains an object called Material preview. You can control the rendering settings for the material preview by editing this object. Save the startup project after editing if you want to change the rendering settings permanently.



Editing rendering settings for the material preview

Post processing settings for the material preview window

Some frequently used material properties are rendered using the post processing system. Glow is a good example of this. Therefore, the material preview scene should include a valid post processing configuration.

You can define the configuration by including a post image with the suitable sub modules to the material preview scene. See the manual chapter Post Processing for further information. The name of the post image should be set to 'Material preview effects'.

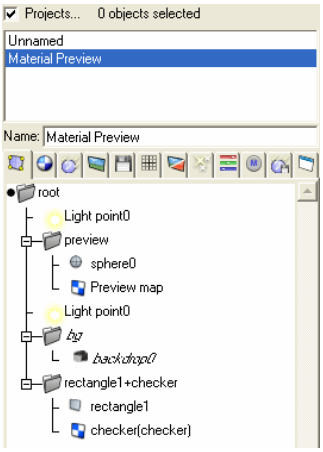
Customizing the default preview project

The file 'material preview' included in the Previews folder defines the structure of the default preview scene. The scene is always loaded at program startup. You can find the scene in the Projects list of the select window and change it as desired. Save the edited file back to disk if you want to change it permanently.

The preview scene can include any objects and materials. However, it should always include one material mapped test object. The mapping object should refer to a material called as Preview. The image below shows the hierarchy of a typical preview scene.

The material preview system loads only objects, materials, post image effects and post particle effects from a preview file. Therefore, use of the Selected Sections option (with the above mentioned sections checked) when saving a preview project consumes less disk space.

The same guidelines are valid for building additional preview scenes, which can be saved to the Previews folder.



The structure of a preview scene

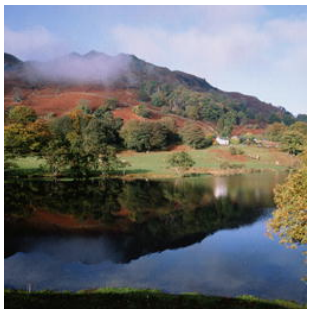
Combining 2D and 3D

This tutorial demonstrates advanced techniques for adding 3D objects into a real photographed scene. It shows various tricks for creating a seamless interaction between 2D and 3D worlds and makes frequent use of VSL. It is therefore intended for advanced users who understand the basics of programming languages.

Level: Advanced

Sample files: 'tutorprojects/material/lake/boat', 'tutorprojects/material/lake/Lake'

We will use the image lake.jpg found in the textures folder of the program installation. Go to the Images tab of the Select window, select New from the popup menu and double click the new image object to open the property window. Use the browse button of the Image file gadget ("..." button) to pick the lake.jpg file. The property window displays the image shown at the right.



A lake scene

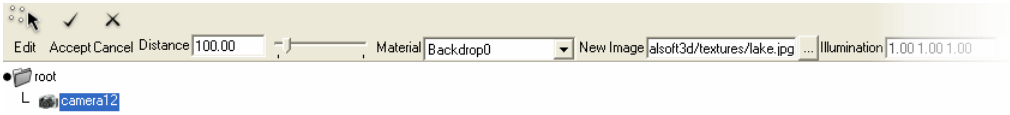
Our main objective of this exercise is to add a boat sailing across the lake. We want to see the reflection of the boat on the water (among other things) so that the animation looks convincing. We do not have exact information of the position of the camera or the dimensions of the lake. Therefore, we have to estimate their coordinates roughly to make things work well.

We start by setting up the view camera. Close the image property window. Reset the view window to the default front view (by clicking N button of view controls). Then open the view property window from "windows/view property window". Select the first tab Camera, which contains the information of view's internal camera. Select Perspective projection from it if not already selected.

We may think that the front bank of the lake is at the world origin, and the camera is approximately 10 meters behind it. The camera is held at a typical eye height - say, 1.5 meters above the ground. Therefore, enter the values '0 1,5 10' to the Position field. The camera is directed quite horizontally, therefore enter the values '0 1,5 0' to the Aimpoint field. The last important setting is Focal length, set 50 mm to that gadget. Now the camera is set and you may close the view property window.

We need to store the defined camera orientation more permanently. Therefore, go to the Creation tab of the toolbar and click Camera tool icon, and select Accept. This creates a camera object that has the same settings as the view camera.

The next step is to create a backdrop plane that shows the photograph in rendering. Make sure that the created camera object is selected in the object hierarchy. The tool control bar automatically displays the camera related tools. Look for Backdrop. Click the button, and new controls are shown. The first, Distance, defines the distance between the camera position and the backdrop plane. Set 100 meters, which is approximately the distance across the (quite small) lake. Leave the next gadget, Material, empty; we have not defined any material for the backdrop. The third gadget New image can be used for defining a texture map for the backdrop. Click the browse button beside it ("...") and select the image lake.jpg from the Textures folder. The last Color gadget won't be used, because the color comes from a texture map. Click Accept.

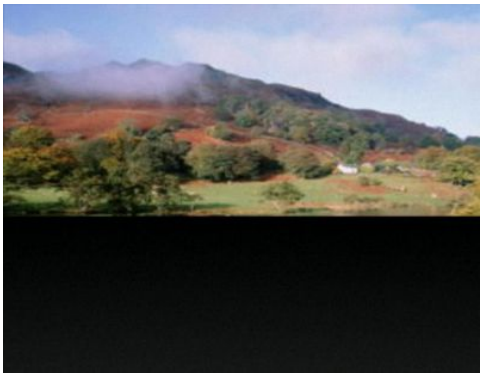


Creating a backdrop

After these steps, a backdrop plane and a texture map to a suitable material becomes inserted into the camera object's sub hierarchy. Render the view and you will see the lake image filling the view. The backdrop plane has such a special property that it automatically expands itself to fill the whole view of its parent camera object.

Next we will add a plane representing the water surface of the lake. The backdrop plane already shows the water, but it is in a wrong place (100 meters away), and is oriented vertically, so it cannot interact with objects close to the camera. Therefore, take a top view. Switch to Parallel mode by disabling View/Camera/Perspective from the view popup menu, for example. Then zoom right out until you see the backdrop plane. (Drag the root directory to the view window will set clipping planes to show all objects.) It may be wise to open the numerical window (View/NumericWindow in the main menu bar) if not already opened. Select the Analytic rectangle from the rectangle tool's shortcut menu in the Creation toolbar section, and draw a large rectangle covering the view. Note that the edge of the rectangle should go behind the backdrop plane (z coordinate over -100 meters).

When the rectangle is ready, close the numeric window. You may rename the rectangle as Lake for easier identification. Then select the camera object and pick the menu View/Camera/Object view (Right click on the view window) from the view's popup. This sets the view window to show the orientation stored in the camera object. Render the view. The created ground plane hides the backdrop's lower half. Optimally, the rectangle surface should become visible about at the waterline behind the lake. If this is not the case (for example, no water is visible at all), select the camera object, open the property window, and go to the Spec tab. This shows the camera specific object attributes. Change the Aimpoint from '0 1,5 0' to '0 1,7 0', which directs the camera slightly upwards, therefore lowering the ground level visible through it.

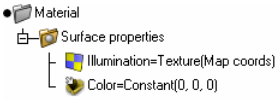


Water surface inserted

Update possible camera object changes to the view by selecting View/Camera/Object view, and render (Dragging the camera from the Select window to the view will achieve the same result). When the adjustment is suitable (in the image above, some water is slightly visible at the right edge), close the property window.

Now we have a rough model of the water. The next step is to make a material for it. Go to the Materials tab of the Select window. You may get rid of extra materials by selecting Materials/Purge Material Library from the main menu. The only material that is left is called backdrop0, which was created by the backdrop tool. Select it and select Duplicate from the popup menu. Double click the icon of the copy to open the property window. Rename the copied material as lakesurface in the Name field. We do not call it water, because the created material will not have the optical properties of water, but it will include quite a special combination of shading effects.

Enable the Advanced option and maximise the VSL Objects frame of the property window. Drop a Constant VSL object onto the Surface properties shader. The inserted Color=constant(0,0,0) removes diffuse shading from the water. This is necessary because the lake photograph already includes such shading, and we do not want to shade it again in the 3D model. The first version of lakesurface is now ready and you may close the property window. Later, we will improve the material step by step to achieve better and better integration between the 2D photo and inserted 3D objects.



The first version of the material for the water surface

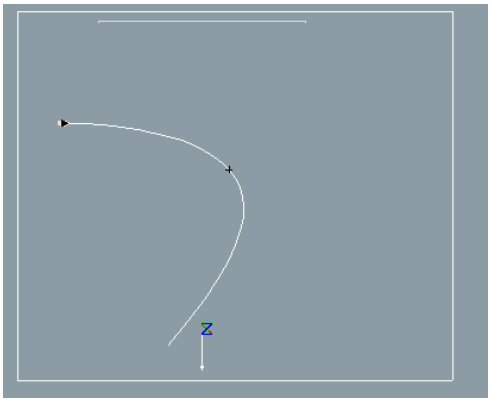
Go to the object hierarchy tab of the Select window and select the rectangle representing the lake. Select the Materials tab of the main toolbar and click the Default map tool icon. Select lakesurface into the displayed Material gadget in the tool controls, and set Mapping type to Camera. Click Accept. If you now render the view, the ground rectangle has disappeared and all you see is the lake photo. Anyway, now there is a real 3D ground plane in the scene and the color information for the water area comes from a sensible distance. Save the project as lake1 to a suitable temporary location using File/Save as menu to backup the work done so far.

Path animated boat

Next we add a small boat to the scene. Therefore, reset the view and select parallel projection. You may model the boat yourself, or load the included simple boat object, which was freeform modeled. Use Paste from file of the

Select window's popup menu for this purpose. Remember to disable the replace option if you use File/Load menu for loading the boat, otherwise you loose the already modeled scene. The boat should have sensible dimensions: the length can be about 3-4 meters. You can use the numeric window and the scale tool to adjust the boat size to the right magnitude. Using the front view, move the boat so that its bottom is slightly below the waterline (y coordinate zero; world origin is at that level as well).

Take the top view and zoom right out, to see the whole lake rectangle (alternatively, drag the lake rectangle from the Select window into the view to zoom the object). Move the boat to the top left corner of the lake rectangle. Then select the curve creation tool from the toolbar and draw a cubic curve starting from the boat, going to the right and downwards, then left and down. In other words, in the camera view, the boat will appear from the left near the back shore of the lake, make a turn at the right and then pass the camera at a close distance disappearing to the left, like in the image below. Click Accept to finish the curve tool.



The path for the boat

Select the boat and the nurbs curve, in this order. Go to the Lattice mapping section of the toolbar and select the Path tool. The default options produce a directed path, which moves and rotates the boat, which is exactly what we want in this example, so just click Accept.

Open the AnimationWindow from the main menu bar's View menu, if not already open. Play the animation to see how the boat travels along the path. If camera tracking is on, the view window changes immediately as the animation playback starts to show the camera object's view. You can control this using the view property window's Camera/Tracking section. You can also use the View window's popup menu item Camera/Object view to take the camera view.

After the playback has stopped, edit and fine tune the nurbs curve from the top view if necessary. Then return to the camera view, move the time slider to a position where the boat is relatively close to the camera, and render. The bottom of the boat is partly hidden by the lake photo. This is the first step towards the 2D-3D blending effect we want. Currently, the boat seems to be hanging in the air.


The boat does not appear to float in the water because reflections are missing

This typical problem can be solved by adding a suitable shadow, or in this case, a reflection of the boat to the water. This task is a bit tricky because only the boat should be reflected. Other reflections are already included in the scene. Therefore, we need to use VSL for customized reflections.

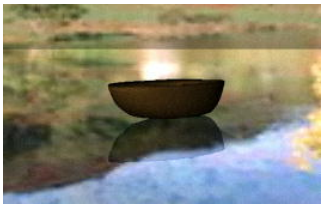
Selective reflections

Switch to the Channels section of the Select window. Select New/Float channel from the popup menu. Rename the created channel object as boatchannel. Then make the object hierarchy tab of the Select window visible and double click the boat to select it and to open the property window. Select the Col tab of the property window, and pick the item boatchannel from the Surface properties/Attribute list. Enter the value 1.0 to the field below it. In other words, we mark the boat with the boatchannel property value of 1, other objects, like the backdrop plane, have the default value 0 and will be excluded.

Go to the Materials tab of the Select window, and select the lakesurface material. Drop Constant VSL object to the surface properties shader, use the popup menu to change Output to Surface:Reflection, and change the black constant value (0,0,0) to light gray (0.7 0.7 0.7). Render the image. Note that we must be rendering with at least "Reasonable quality" rendering mode to see reflections. If this is not the case, go to the rendering tab and drag "Reasonable quality" onto the view window and render again.

There is a reflection of the boat on the water, but things are not quite right:

- The water is too light and reflects the mountains twice
- The boat reflection contains color from the scene behind the boat, just like the boat was translucent in reflections

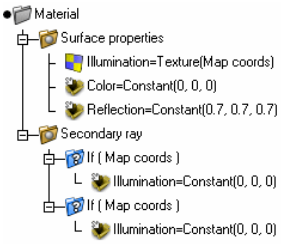


Incorrect reflections

Let's continue editing the lakesurface material on the property window. Drop a new Shader object from VSL Objects just under the Material node. Change the Type to Secondary ray. This shader controls traced rays - just what we need. Drop an If object into Secondary ray shader. Change Input 0 from the popup menu to Traced ray:boatchannel . The If object gadget at the bottom of the screen displays three conditions. Change the topmost X field to If inside and enter Lower=0.1, Upper=1.1 to the two numeric fields below it. Leave other conditions unmodified. In other words, we test here if the reflected ray contains nonzero boatchannel information.

Now drop a new Constant object into the If object. (Place it underneath and inside.) As explained above, this If block handles the boat reflection area, because it is executed only if the boatchannel value is between 0.1 and 1, and only the boat object has currently such a value (its boatchannel property was set to 1). The problem area is that the reflection also showed the background scenery. The reason is the original camera projection mapped illumination. Therefore, set output channel of the new constant shader to Surface:Illumination. The value zero is exactly what we want: we reset the illumination mapped from lake.jpg to zero before adding the reflection from traced ray to it.

Another problem was that outside the boat reflection, the lake reflected the scene behind it twice. The first correct reflection was already in the scene photograph. The second incorrect reflection comes from the added reflection property: rays reflect from the lake rectangle and hit the backdrop plane. This effect can be eliminated easily.



Select the If object and then Copy from the popup menu. Select the Secondary ray node from the VSL hierarchy and Paste from the popup menu. Then change the first condition of the new If object to If not inside. In other words, this VSL block will handle those reflections that don't have boatchannel value set.

Select the Constant object inside the If object and change its output to Traced ray:Illumination. This way we reset unnecessary reflections (not showing the boat) to zero. Test render to verify the result. This is also a suitable point to save your work again.



Correct reflections

Waves

The next effect to be added is waves around the boat. The main difficulty is that the lake photo already contains reflections of the scene in the background without any waves. Therefore, we have to somehow modify the existing reflections.

First we create a suitable procedural bumpmap for the waves. Go to the Materials tab of the Select window, and create a new VSL material. Open the property window (if not already open), and rename the material as waves. Open the advanced properties. Drop a Shader object onto the Material node, and change the type of created shader to Surface geometry. This shader is suitable for bump mapping purposes. Drop a Variable object to Surface geometry shader, change its name to tmp and type to Vector. Then drop another variable to the same shader, rename it as 'r' and set type to float. Then insert an Operation object to the shader, and select dot product from the long list in

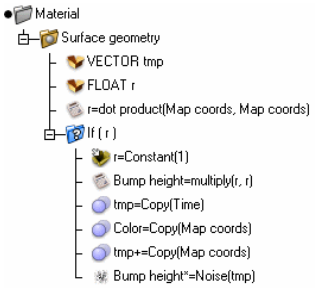
the Type gadget. Set Output to r variable using the popup. Both Map coords inputs are correct. This VSL object computes the square of the radial distance from the zero point of the mapping. We will use the radial distance to fade out the waves within a circular area around the boat.

Drop an If object to the shader, and set its Input 0 to float variable r with the popup menu. Change the first condition X from Always to If inside, and define Lower=0, Upper=1. The included code will be executed if the mapping coordinate measured distance (which can be adjusted by scaling the mapping object) will be less than one.

Drop a Constant object into the If object. Set its output to r, value to 1, and change Operator to Subtract from the popup menu. Then drop a new Operation object into the If object. Select multiply from the long Type list, Set Input 0 and Input 1 both to r, and set Output = Bump height. The purpose of these tedious and technical computations is to define a nice bell-like curve that fades out the wave smoothly. The nice and smooth curve used here is :

bump height = (r^2 -1)^2

We need a couple of VSL objects more. First add a Copy object to the end of the If sub hierarchy. Set Source to Surface:Time, and Destination to vector variable tmp. The Copy object's gadget shows three selectors which bind the sub channels. Change the first two to the None state, but set Sub channel 3 to Time. Then drop another Copy object to the If node. The default source Map coords is OK, but change Destination to tmp and Operator to Add from the popup. Now we have computed a set of modified mapping coordinates by adding the animation time to the third coordinate component. The purpose is to animate waves by sliding a noise field through the water surface. Sounds strange - but it works well! The last object to be added is Noise: Drop one to the end of the If hierarchy. Set output to Bump height and Input 0 to tmp. Set the Operator to Multiply from the popup menu. Then adjust the noise properties at the bottom of the property window: Set Octaves to 3 (it defines the level of detail in the waves), Amplitude = 0.1 (height of waves = 0.1 meters), Smooth and 1D enabled. The material is ready. Its structure is shown on the right.



The VSL structure of the wave material

Close the property window and rewind the animation to the beginning. Take a parallel top view and select the boat from the object hierarchy section of the Select window. Then select Autofocus to find the boat (Autofocus is in the view's popup menu and in the view control icons). Zoom out a bit so that you see a reasonable area around the boat. Select only the lake+lakesurface level from the Select window. Then go to the Materials tab of the main toolbar, and activate the Parallel map tool. Select the waves material from the material list in the tool controls.

Click first time just behind the boat (where the waves are supposedly highest), and move the mouse downwards and to the right (the "shift" modifier key held down) to get a uniform rectangle. Click a second time to finish the parallel map. The edge of the parallel map should be 1-2 times the length of the boat. You may resize the parallel map later to adjust the extent diameter of the waving area.

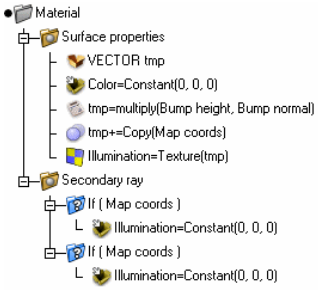
We have to animate the wave parallel map exactly the same way as we animated the boat. Make sure that the parallel mapping is selected. Move the mouse over the middle point of the object space handle on the view window, and drag the handle so that it is in the top left corner of the parallel mapping (just behind the boat). We do this to ensure that the maximally waving area follows the path, maintaining the originally defined 'behind the boat' status. The path tools fit the object space origin to the path. Press the Shift key down and multiselect the nurbs curve, which was used in animating the boat. Go to the Lattice Mapping toolbar section and use the Path tool to animate the parallel map with the default options.

Take the camera view and start animation play. You should see the parallel map following the boat across the lake. Stop the playback when the boat is quite close to the camera, and test render. You should see the boat reflection nicely twisted because of the waves. The strength of this effect can be adjusted by editing the Amplitude slider of the Noise object in the waves material hierarchy.

At this point, the original reflections of the photograph do not follow the defined local waves. Let's fix that next. Activate lakesurface from the material tab of the Select window and open the property window.

Drop a new Variable object to the top of the object list inside the Surface properties shader. Rename it as tmp and set the type to Vector. Then drop an Operation object just above Illumination=Texture object. Set Output to tmp, Input 0 to Surface:Bump height and Input 1 to Surface:Bump normal. Select Multiply from the Type gadget at the bottom of the property window.

Drop a Copy object just below the previously added multiply operation. Change Destination to tmp and Operator to Add from the popup. Then select the already existing Illumination=Texture(Map coords) object from the VSL hierarchy and change its input from Map coords to tmp. The material is now ready. The subtle effect of backdrop reflection waves is achieved by adding the bump vector to the texture coordinates directly! This is just a fake but better than static reflections.



The final version of lakesurface material

Foreground layers

The last part of this tutorial explains how to add foreground mask objects. Namely, the photograph includes a tree at the right edge of the screen, and also the front bank of the lake is visible. It may happen that the boat meets these areas while travelling on the lake and hides them, which certainly looks strange. Also, the front bank should not reflect the boat.

Close the property window and take the camera view. At this point we have to pay some attention to the image aspect. We will draw the mask objects over a backdrop image on the view window, and therefore the view must have exactly the same image width/height ratio as the camera object being used for the final rendering of the animation. The camera object was created from the view settings, so they may already match. The numerical window and other windows appearing at the bottom of the screen may have changed the view size, so you may have to close some of them to restore the original situation.

Another possibility is to use the Spec tab of the property window to edit the camera object's properties. In the Image aspect section, there are some predefined settings to choose from, or you may enter the desired custom value. The most suitable value depends on the source of the backdrop image. If the lake image was digitized from a PAL video signal, then we should use the same image aspect in the 3D model as well, otherwise the backdrop will stretch (spheres will look like eggs, etc.). Let's assume that the backdrop is a PAL image. Therefore, choose PAL, then close the object property window, open a view property window, select the leftmost Camera tab and open the Image Frame section. Then enable the Fixed size option and enter the same aspect value (0.75 for PAL) to the Image aspect field. The height of the view is adjusted accordingly. If the height grows, the bottom area of docked views becomes invisible, because the view cannot stretch. If that happens, reduce the width of the view by 50 pixels and enter the aspect again. Repeat this until the view is capable of displaying the image with correct proportions.

After these preparations, copy the camera view onto the view window by selecting the camera object and using the View/Camera/Object view popup menu function. If the preparations were successful, the wire frame of the camera is visible as a thin rectangular border matching the view edges. If the camera view rectangle is not visible, zoom out a bit. If the side edges of the camera view rectangle are visible but the top and bottom edges are not, or if the top and bottom edges are far away from view borders, the view - camera aspect matching has failed and you should recheck the aspect ratios.

After taking the camera view successfully, close the object property window and the view property window. Select the Images tab from the Select window. Drag and drop the lake image object onto the view. This activates view's continuous backdrop drawing. Select the curve tool from the toolbar. Set the Type field of the tool controls to Analytic, which option creates a filled polygon.

Start drawing a polygon from the bottom left edge of the view, so that the polygon follows the outline of the front bank. After the rightmost point of the bank, click one point at bottom right corner of the view, then at the bottom left corner, and then Accept to finish the tool (see the image). Draw another polygon to cover the tree at the right edge of the image.



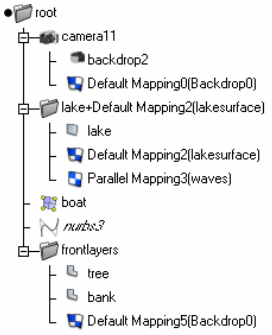
Foreground polygons

When the polygons are ready, open the view property window, go to the Draw tab and clear the Backdrop/Image field displaying the name of the image object, which was dropped onto the view. The backdrop disappears. Close the view property window.

Select the two polygons, and use the Drop to a level tool of the Select window's popup menu. Select and rename the created level as frontlayers. Open the property window and change the color of the frontlayers level to black. This eliminates diffuse shading (black diffuse color was added to the lakesurface material, but we will now use Backdrop0 material which does not eliminate diffuse color). Close the property window.

Make sure that frontlayers is still selected and activate the Default mapping tool from the Materials section of the toolbar. Select the Backdrop0 material, set Mapping type to Camera and click Accept. A default mapping object texturing the polygons becomes inserted to the hierarchy.

The long tutorial example is now ready. You can save and test render it, for example, to an .avi file, to see the created effects animated.





The final hierarchy and a sample image

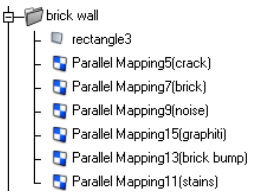
Adding Dirt

Adding Dirt

This tutorial outlines various methods of mapping your models to look realistic - in essence, dirtying up the otherwise shiny "super real" look of computer generated imagery.

Level: Medium

Perhaps the most useful method of creating additional detail to your scenes - scope mapping allows the user to add additional materials to a single object/level. Each successive material takes precedence over the previous material. e.g. the material at the bottom of the select window is placed on TOP of the material underneath in the model. (i.e. each material is layered successively, the lower in the select window, the closer to the top or front of the material.)



This is useful for adding graffiti to walls - rust stains, drains in the ground - or overlay any object or material. To effectively use scope mapped materials - one generally makes use of an alpha channel or seperate image that is used to 'cut' the material (unless a simple square parallel map suffices). This alpha channel can be applied to:

- The **Scope** channel - which will simply cut out the **extent** of the mapping.
- Or the **Fade** channel - which actually cuts the **object** that is mapped.

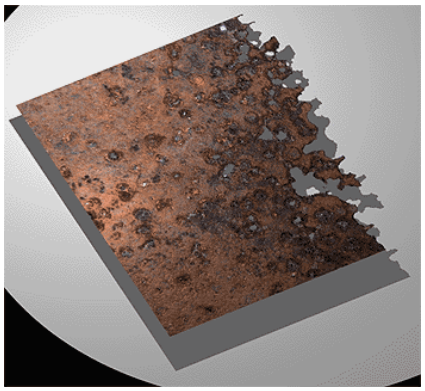
Obviously we can map any image to any channel in Realsoft 3D - but for the purposes of this exercise we shall deal with scope and fade mapping. To achieve the "cutting" action can also be brought about by a variety of means:

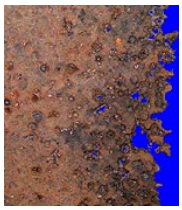
- Using the **clip** material in the standard /image templates
- Using an Image Alpha channel like in a 32 bit Targa file
- Or having a seperate Greyscale image to the color image.

Fade mapping with the CLIP function

The clip material uses a color range to determine what extent should be cut. This material is best used for areas of solid color. If the mapped material has a blue background, we can set the clipping range to encompass all the blues to cut out. Refer to:

example project file: tutorprojects/material/dirt/rusted01.r3d



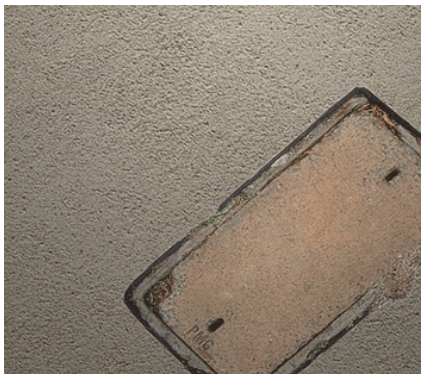


In this scene, the blue in the image is used as a mask. We can set any color as a range - ideally it would be a color that is not used in the image. This method generally produces a hard edge and is probably the least desirable.

Scope mapping with two images

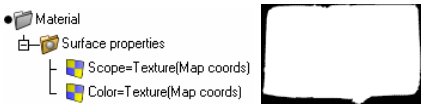
The next scene example project file is: tutorprojects/material/dirt/scope01.r3d.

This example uses two separate images. The diffuse color image D_manhole.jpg and an Alpha image (Cutout) called A_manhole.jpg. Its often a good idea to prefix all your textures with B_ for Bump, A_ for Alpha, etc.



This simple scene has mapped two materials onto one object. The foundation is the concrete material that surrounds the whole floor. On TOP of that (i.e. below it in the select window) we have put our manhole texture. Note the structure of the manhole texture.

It is very simple : The Scope channel is set to our alpha channel image (shown above). Note that black designates complete transparency - and white designates solid areas of color. Any grey between these values would give a soft transition between the two.

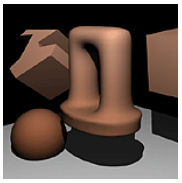
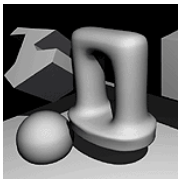
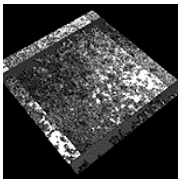
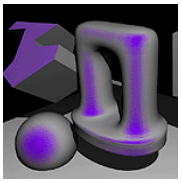


The color channel is set to the color image of the manhole. Note that the color image does not need to be cut out. The alpha image or channel does this for us.

The end result is a manhole sitting in concrete on one object. The scope channel is very useful for adding all manner of materials on top - Windows, graffiti, leaves, grime, etc.

Dripping rust often congregates under beams and appendages - adding a simple scope rust material over and over again can add whole dimensions to your scene - all from either an image map or simply a dark color mapped on top of another material with 0.25 % scope.

Materials Useful for Dirty Mapping

	**Adding Noise over materials** Using a low scope - one can "dirty" otherwise bland areas or disguise repetitive materials by simply adding a noise texture.
	**Faking shadows** Using a simple greyscale gradient image, we can add shadows under eaves, or add subtle radiosity like effects to objects. Adjust the scope level to increase or decrease the effect of the material. project: dirt/gradient.r3d
	**Angle based scope** This material takes into account an objects ANGLE to the materal mapping - hence we can add "dust" to the top of objects simply by applying this material downward. Often having a white down material and a dark up material can effectively simulate both sun burn on top and grime underneath. project: dirt/dust.r3d
	**Darker bumps** This material uses the height of the bump to add grime into the gaps. It essentially only burns the darker regions of the bump material - but can be good for tiles and very rough materials. project: dirt/dirtybump.r3d
	**Perpendicular scope** This can be used to fine tune materials that should only fall on a certain area of an object - used in conjuction with a finite mapping method - this is particularly useful for additional dust and grime. project: dirt/smscope.r3d

Truly Effective materials make use of the whole gamut of available channels: Specularity, Color, Fade, Bump, Transparency, etc. Through clever use of materials we can keep geometry to a minimum and maximize the realism of a scene. Let's have a look at a scene created earlier and examine how it works.

Dirty Wall

This example scene is a simple exercise in creating realism through materials. Note that there are only 3 rectangles and 1 cube in this scene! We could have cleaned up the junction between grass and edge with a NURB surface if desired.

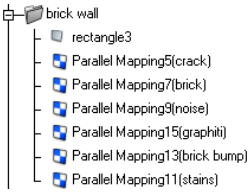
example scene: tutorprojects/dirt/dirtywall.r3d

There are several elements in this scene: The wall, grass, downpipe and grass edge. Let's examine each in turn and how we achived each material.



The Wall

This object has the most materials applied. At the "back" is our first crack - this is simply a bump image (black and white) of a crack. Note if we map consecutive bumps to each other, the background must be black as white is high, black is low. We wanted the material to crack in, so we have used a negative value to "invert" the bump.



Next is the brick photo by itself. This is only a small section of a brick wall made tilable. This is best achieved in Photoshop by using the filter/offset with 'wrap around' set, to see how a material tiles. Using the stamp tool, you can make a material tile properly.

Over the brick we have put a subtle noise. This adds a basic black and white noise on top of the brick to give it a slightly dirty appearance. Note we use the Scope parameter of the parallel mapping object itself (under the Properties/spec tab) to give the noise more or less effect. Currently set quite low (0.6).

On top of the brick we have added a stain; vertical water stains that have dripped down from the top of the brickwork. This material is simply a scope material. The scope channel uses a B+W image again, with white designating the area of effect. Note we can put any material or color in this material and the area of scope limits it to this. In this material however, we have simply used a color- hence the scope limits the color to the area of the scope image, thus creating our vertical stains.

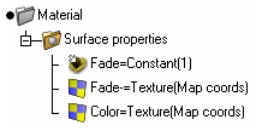
Then we have a simple brick bump. Note we could just as easily have added the bump with the brick material. We have used the original bump material for the bump channel, but ideally we would create a B+W bump material to emphasise the areas of interest (the mortar joints).

Last is the graffiti. Like our stain, the graffiti is simply a scope material. The white defines the area of color, and creates our graffiti. Note that any of these scope materials can be moved freely around the wall or duplicated to create more. Also, we have set the Finite options under the parallel mapping object. This limits the area of effect for this material within the boundaries set by the mapping object. Unless we wanted to tile this graffiti, generally we should use Finite mappings for scope based objects.

Grass foreground

This object was used to show the use of the fade mapping channel. We could have just as easily mapped this material onto the brick wall using the scope channel instead of the fade channel.

The grass foreground material uses the fade and color channels. Note that we have also used a fade constant. We set the fade as a constant of 1 (white) then with the image mapped to the fade channel, we use a subtract operator to essentially "invert" the alpha. We do this because Realsoft 3D uses white for transparent and black for solid in fade channels only. Scope uses the opposite. Hence, we invert our alpha channel by doing this.

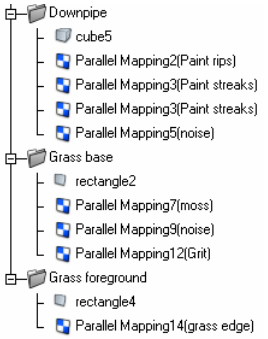


The texture is a Targa, so the alpha is embedded into the same texture. Hence, both the color and fade textures have the same image applied. Under the Bindings tab of the fade texture, you can check that it is bound to alpha->fade. An alpha JPG would have something like ColorR->fade.

This material is then parallel mapped to the rectangle and repeated in one axis. The alpha acts as a cookie cutter and actually cuts the object itself (vs using a scope which merely limits the material).

Grass

This baseplane is simple: it has a grass texture parallel mapping tiled, then another 'Grit' material applied over that - essentially simulating some sticks over the grass - and then a duplicate copy of our Noise material from the brick to simulate a bit of un-evenness.



Downpipe

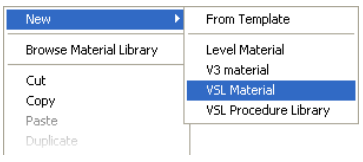
This object is probably more complicated than it needs to be - but shows a multiple layering of scoped materials. Note these materials make use of the scope - hence we have not globally set the color of the object itself - hence the color is derived from the object properties/col tab. The multiple scoped materials give a random appearance to flaking paint - all 3 of which are simply B+W scope maps and a color. We then added our noise again to give it some random discolouration.

Color Control

Creating 3D rendered images often requires the use of textures, like texture maps and bump maps. Consequently, sooner or later you may want to change the color or the brightness of the rendered texture a little. One way of doing this is adjusting your textures in your preferred 2D program and reload the modified textures. There is also another option: you can adjust colors and brightness with the use of the VSL Curve object in Realsoft 3D directly, which is much more intuitive and also a lot faster.

This tutorial chapter shows how to control the color and brightness of your texture by using the VSL Curve object.

Level: Beginner/Medium

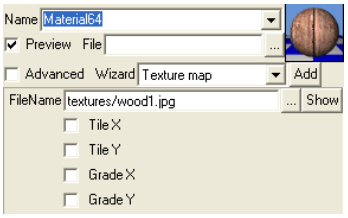


First we start by creating a new material. In the Materials tab of the select window, click the right mouse button (RMB) and select New/VSL material.

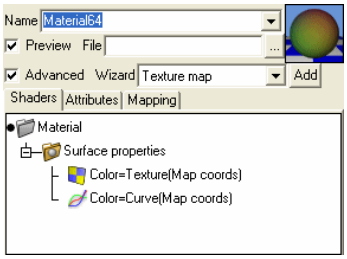
This adds a new and empty material, called Materialxx, to your library. If you have material previews activated - at the Materials tab right click and select Show Previews, a black square is added at the bottom of the materials list. Double click the new material to open its property window. You can also press the 'p' key to open the property window.

At the top of the property window, the name of your material is displayed. Rename it to "ColorWood".

In the property window, turn on the Preview checkbox, so you can immediately see the result of your actions. A white sphere appears. Now use the wizard to add a "texture map". Select wood1.iff from the textures directory and as soon as you load it, the white sphere in the small preview window changes into a wooden sphere.



The white sphere in the preview window has changed into a wooden sphere

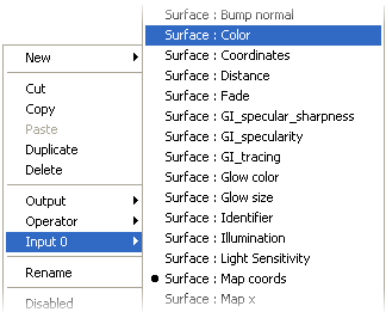


Adding the Curve object changes the sphere into a colorful object!

Next, check the Advanced checkbox to get access to the VSL components of the property window. From the VSL Objects, select the "Curve" object and drag-and-drop that in the Surface properties shader. Make sure the Curve object is the last object in the VSL hierarchy as shown below:

The effect of adding the Curve object is that the color of the sphere in the small preview window changes immediately from a wooden sphere into a colored sphere. Let's fix that. The Input0 variable of the Curve is by default set to Surface:Map coords. Select the Curve object in the VSL hierarchy and open the RMB pop-up menu. Select Input0 from the pop-up list and notice the black dot at Surface: Map coords. Now change that to Surface:Color (simply select that from the list). Immediately the appearance of the preview sphere is restored to a wooden sphere.

The material is now ready, so you can close the material property window.



By default Input0 is set to Surface:Map Coords; change that to Surface:Color

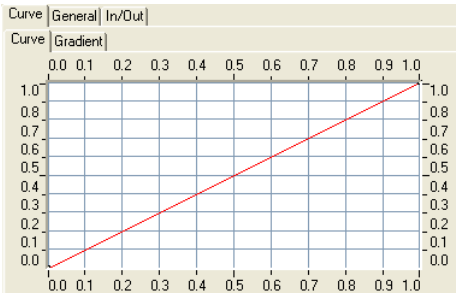
Setting up the scene

For this tutorial we need a very simple scene. In front view, create an analytic sphere and add a point light source somewhere above the sphere, so the scene is lit. This will do just fine. Make sure the sphere is selected in the select window and go to the materials tab. Select the ColorWood material, press the RMB and select Map -> Parallel from the pop-up menu. In the view window create a parallel mapping that encloses the sphere. Render the view to check your results.

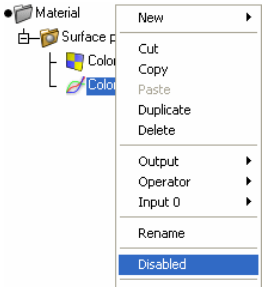
Using a curve to change color

Controlling the color and brightness of the woodtexture is now a matter of modifying a curve. So, let's have a closer look at this curve and how the curve controls color and brightness.

Make sure the property window for the ColorWood material is open and the Advanced option is set, so we have access to the VSL components of the material. Click on the Curve object and have a look at the lower part of the property window. The first thing you see is a red curve with two knotpoints; one knotpoint at (X=0, Y=0 - lower left) and one knotpoint at (X=1, Y=1 - top right).



One knotpoint has coordinates X=0, Y=0, the other has coordinates X=1, Y=1



Enable and Disable the Curve object via the RMB pop-up menu

In its current state the curve has no effect on the material. To check this just render the view. Then right click on the Curve object in the VSL hierarchy, select Disabled and render again. Nothing changes. Now in the same way, enable the Curve object again.

The Curve object makes use of 3 curves, a red, a green and a blue curve, which in this case control the RGB color values of your texture. So, the red curve contols the R(ed) value, the green curve the G(reen) value and the blue curve the B(lue) value. We now only see a red curve because all three curves are exactly on top of each other, the red curve being the one on top.

To modify a curve, one clicks in the graph on the line to add a knotpoint, then move the knotpoint to the desired position. Let's try this and see what happens:

Add a knot point in the middle of the curve (X=0,5 and Y = 0,5) and drag that knotpoint to X=0,2 Y=0.8. Note that you can enter the exact coordinates for a selected knotpoint via the numeric input fields just below the graph.



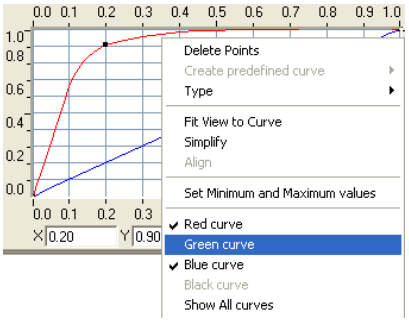
Numeric input fields for exact positioning of selected knotpoints

The first thing we notice is that only the red curve has moved. At it's previous position a green curve appears, which also has a knotpoint. If you move the knotpoint of the green curve as well, a blue curve appears, also having a knotpoint at the same position. When you create a knotpoint while all three curves are on top of each other, all three curves get one at the same position. Because the material preview in the property window is active, you can immediately see the result of your changes. Render the view and you will see that the material has now a more orange color.

The conclusion is that by modifying a curve, we can influence the color of the texture. High values in the Y-axis result in a higher RGB color value. Because we moved a knotpoint in the red curve to a higher Y-value, we increased the value of the red color component of the texture. Therefore the texture appeared more reddish. Similarly, if we move this knotpoint to a low Y-value, for example to X=0,8 Y=0,2, the amount of red is decreased and the texture will become greenish.

Of course you can also decide to move for example only the green curve, or both red and blue. Just try some of these variations and see what happens. Remember that undo (ctrl+z) restores the curve to its previous shape. If you get completely lost, drag select all knotpoints you have added and select Delete Points from the RMB pop-up menu. This will restore the curve to its original shape.

By default all three curves are visible and editable in the graph. Sometimes it is much easier if you have only one curve or two curves visible in the graph. If you want to exclude a curve from being displayed in the graph, open the RMB pop-up menu and deselect the curve. Only the visible curves in the graph can be edited.

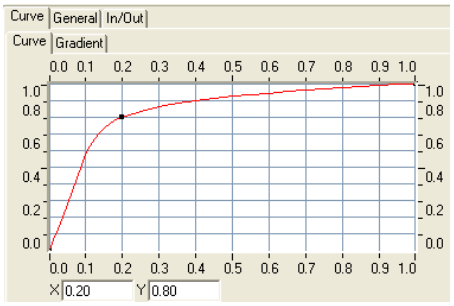


The green curve is excluded from being displayed in the graph

When you move a knotpoint of a single curve, you control the respective R,G or B value. What would happen if you move a knotpoint in all three curves at the same time? In that case all RGB values are changed in the same amount and thus will the color get lighter (if the Y-value increases) or darker (if the Y-value decreases). Let's see how this works.

Make sure all curves are enabled. Delete all knotpoints you have added, so the curves are restored to their original shape. Again add a knotpoint at X=0.5, Y=0.5. To move a knotpoint for all three curves at the same time drag-select the knotpoint (this selects the knotpoints for the red, green and blue curve simultaneously) and move it. Now move all three knotpoints to X=0,2 Y=0,8 and render the view.

As expected, the sphere now has a color which appears whiter. Now you can also predict what happens if you move the knotpoint to X=0,8 Y=0,2.



Knotpoint for all three curves moved to X=0,2 Y=0,8

Until now we have created and moved new knotpoints. You could also decide to change the position of the starting point and end point of curves. Move the starting point of the curve to X=0 Y=0,5 and move the end point to X=1 Y=0,5. What happens? The curve is now a straight horizontal line. All RGB values are made the same, which results in a shade of grey. Subsequently, if we move the horizontal curve upwards, the color will change more to white; move it down to change more to black. This shows how you can control the color of your objects and even overrule textures.



Chapter 7. Rendering

Introduction

The photorealistic rendering engine of Realsoft 3D uses scan line, ray tracing and post processing techniques to produce images with a high degree of realism.

In order to generate images, the rendering engine needs the following information:

- *A scene* containing geometric objects and materials.
- *Rendering Settings,* which control how the rendering engine performs its job. For example, rendering quality belongs to rendering settings.
- *Post Effects*, such as gamma correction, dithering and lens flare.

The default startup project defines reasonable default settings for photorealistic rendering. The only thing you have to do is to create a scene and click the Render button.

For example, to render a sphere:

1. Click the Analytic/Sphere tool and enter two points through the view window to define the position and the size for the sphere.

2. Click the Render button in the bottom of the view control bar.

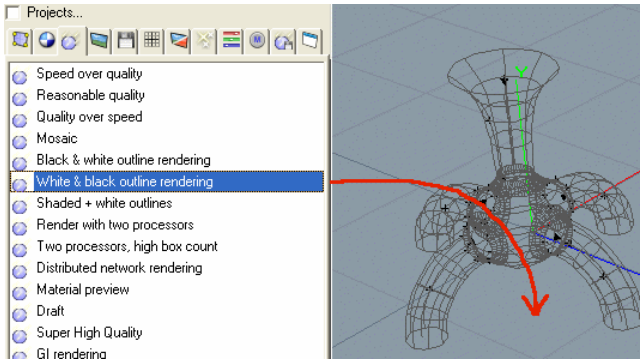
Specifying rendering settings for the view window

Rendering settings can be defined by dragging and dropping a rendering settings object into the view window. You can also select the rendering settings through the View Property window/Render tab.

For example, to activate Outline rendering:

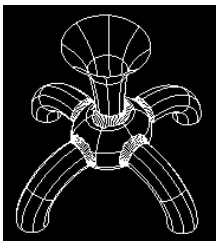
1. Go to the Select window's Rendering Settings tab. The default startup project contains several predefined rendering settings object.

2. Drag&drop one of the outline rendering settings into the view window.



Drag & drop a desired rendering settings object into the view window

3. Click the Render button and an outline image is rendered.



Outline rendering

Specifying post effects for the view window

A suitable post effect collection can be selected for the view window the same way as rendering settings i.e. using the drag&drop technique.

For example, to activate a star filter post effect:

1. Go to the Select window's Post Image Effects tab.

2. Drag & drop the 'Default effects + Star Filter' object into the view window.

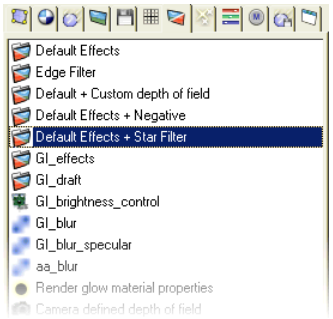


Image effects



Star filter

3. Render the view. Objects will be star filtered.

The view window of the standard configuration uses the Default Effects post effect configuration. It automatically renders glow, particles and depth of field, if the defined material properties, camera options, etc. so require. Therefore, you probably don't have to change the post effects configuration that often.



Note

The Post Image Effects tab contains two kinds of effects: Post images and actual effects applied to post images. In the example above we dragged & dropped a Post Image to the view window. Dropping individual effects to the view window adds them to the post image which is selected for the view. Effects can be rendered only by applying them to post images.

Grouping post effects to post images is explained in detail later in the post processing tutorials.

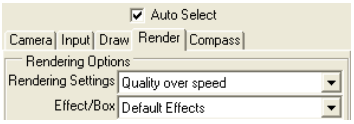
Rendering Settings

A *Rendering Settings* object is a collection of rendering options. Realsoft 3D's photorealistic rendering engine is extremely powerful and designed for maximum realism and image quality. Correspondingly, rendering settings include a large number of options for controlling the entire rendering process.

Rendering settings can be managed through the select window's Render Settings Tab. This tab shows the current library of predefined settings, and you can freely add new ones to suit your needs.

To create a new rendering settings object, select the New pop-up menu. This creates a new object into the select window. As usual, you can use the property window to modify properties of the newly created object.

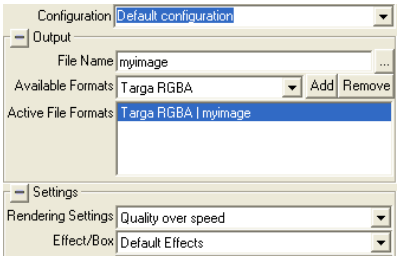
You can simply drag & drop a render settings object from the select window to a view window. Another solution is to use the view property window's Rendering tab to select the rendering settings. You can open this window from the View Properties menu, which is available in the view's popup menu (right click on view.)



Selecting rendering settings from the view property window

Rendering settings for file rendering

The File/Render window allows you to render the current project to a file. Rendering cannot be started before a rendering settings object is selected from the Rendering Settings list.



File rendering with Quality over speed

Rendering to a File

A project can be rendered to a file using various file formats, such as BMP, JPG, TGA or the native Realsoft 3D file format. The native format is the most flexible alternative, because it allows you to write any number of image channels to a file using any of the available data types: 8-bit byte, 16-bit word, 32-bit integer or 64-bit floating point.

Both still images and animations can be rendered. Animations can be rendered to a sequence of indexed images or to one (or more) AVI or FLC animation format file.

Rendering an image to a file

To render the current project to a still image file:

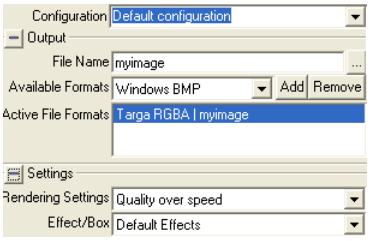
1. Make sure you have created a camera object to the current project. The file rendering tool needs one camera object to define the camera attributes such as the position and direction of the camera, focal length, etc.

Check that the camera object has a proper aspect ratio. Use the camera object property window to set the aspect to PAL or NTSC if you plan to render a video resolution image. If the purpose is to render a square-like image for a magazine advertisement, set the camera aspect to 1 (and use identical 'Width' and 'Height' in the resolution settings defined below).

2. Select the pull-down menu File/Render.This opens the file rendering window.

3. Select a suitable file rendering configuration. One of the predefined configurations, such as 'PAL', may already include suitable settings. If none of them is suitable, use 'Default configuration'.

4. Select a Rendering Settings object. Rendering settings define anti-aliasing quality and other rendering specific information. The default startup project includes a number of predefined rendering settings. You can create new rendering settings from the Select window's Rendering Settings tab.



File rendering window

5. Select a suitable post processing effect to the Effect/Box control. If you don't want to activate post processing, set this field to empty. The post processing system renders post effects such as lens flare and glow.

6. Configure Active File Formats. To do this, find the desired format from the Available Formats field and click the associated Add button. This adds the file format into the Active File Formats list. You can remove inserted formats using the RMB popup menu. Note that you can render to any number of files simultaneously. This is often a very useful feature. For example, you can render color information to one image and alpha information to another image.

7. Specify the resolution for rendering. You can also select one of the predefined resolutions from the 'Presets' control. Make sure that the resolution and the aspect ratio of the camera match. For example, if you plan to render a 'panoramic' image of width=2000, height=1000 for printing - the camera aspect should most likely be 1000/2000 = 0.5. Unfortunately dividing the height by the width does not always define a correct camera aspect ratio, because some display devices use non-square pixels. The usual video standards use non-square pixels. Fortunately you can find the appropriate resolution settings and aspect ratios from the presets.

8. Enter a File Name for the image.

9. Click the Render Image button to render the image.



Note

If the Render Image button remains disabled, make sure you have defined at least one image file format (animation formats cannot be used for still images) . Secondly, check that a proper file name is defined. Finally, check that a rendering settings object is selected.

If you frequently need to render to certain file formats with a certain resolution and other settings, it is a good idea to create a new rendering configuration using the corresponding tab on the select window, define the preferred settings and save the new configuration to the startup file for later use.

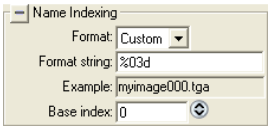
Rendering an animation to a file

Animation rendering happens the same way as rendering still images. Instead of clicking the 'Render Image' button, click the 'Render Animation' button. When rendering an animation using an 'animation' format such as AVI or FLC,

one file is generated (for example myanim.avi). Before rendering, the codec to be used is prompted - e.g. select "Cinepak" for avi - or use your discretion for format quality settings.

If an image format is used, the file render tool generates a sequence of separate images indexed with a frame number.

The Name Indexing controls define how the file name is generated from the actual file name and the current frame number. The default indexing format is flexible enough for most cases. If a more exotic indexing convention is desired, it can be achieved by selecting Custom to the Format field and entering a suitable string to the Format string field. The format string follows C language 'printf' conventions.



Indexing controls

Rendered images are named as follows: [filename][index].[extension]

For example: myimage000.bmp

The first portion is the file name defined by the user (such as d:\mypics\test). Index is the current frame plus the base index definedthrough the *'Base index'* field. The number of digits in the index portion is automatically extracted from the number of frames in the animation. For example, if the length of the animation is 120 frames, the index portion contains three digits. The extension is determined by the selected file format.

The *'Format String'* control shows the format string that is used for generating the index portion of the filename. If 'Format' is set to 'Custom', you can define an arbitrary format string to this gadget. For example, the string '%d' outputs indices without zero padding: frame0.tga, frame1.tga,..., frame99.tga... '%05d' would output: frame00001.tga. '%03d' outputs: frame001.tga, etc.

The '*Example'* field shows a complete sample filename generated by the current settings.



Note

Animation frame indices start from zero. If you want to start indexing from one (frame001.tga, frame002.tga...), set 'Base index' to one.

Box rendering

The 'Render Box' controls limit the rendering to a sub area of the total image. For example, it may happen that after spending one whole day for high-resolution rendering of a complex scene, the user notices that the color of a small object at the top left corner was wrong. Instead of re-rendering the whole scene, the user can render one quarter of the total image width and one quarter of the total height around the object to a new file. The position of the new sub image is known accurately and therefore it can be pasted over the original image using a paint or image processing program.

The controls are:

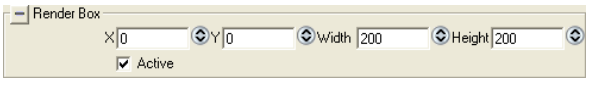
X - The horizontal position of the top left corner of the render box (measured in pixels)

Y - The vertical position of the top left corner of the render box

Width - The width of the box

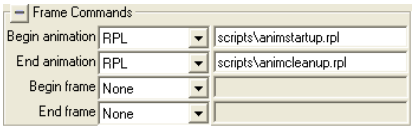
Height - The height of the box

Active - Activates box rendering



Frame commands

The Frame Commands area contains four command gadgets. The commands can be defined using any of the installed scripting languages. In addition to this, DOS (or shell) scripts can also be used.



The following commands can be defined:

Begin animation

This command is executed once in the beginning of animation rendering. For example, a special hardware for animation recording can be turned on.

End animation

Executed once when the animation is ready. This can be used for clean up actions.

Begin Frame

Executed before rendering of each frame.

End Frame.

Executed after each rendered frame. For example, the rendered image can be recorded to a single frame recorder.

The frame command can access the file name via the following predefined macros:

> *[image]* - Full name of the rendered image, for example 'd:\images\test000.bmp'.*[path]* - Path portion of the file name without the last path separator, such as 'd:\images'.*[base]* - Base name (i.e. path and extension removed). For example 'test'.*[ext]* - Extension without the dot. For example 'bmp'.

For example, if the file name is d:\realsoft3d\images\test.bmp:

> [path] - results in 'd:\realsoft3d\images'
>
> [path]\[base].[ext] is the same as [image] - 'd:\realsoft3d\images\test.bmp'.
>
> [path]\[base].tga results 'd:\realsoft3d\images\test.tga'

Distributed Rendering

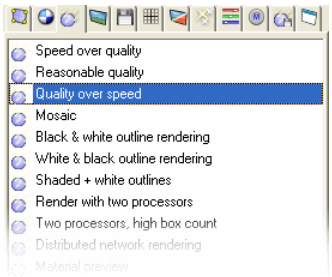
The distributed rendering system of Realsoft 3D subdivides an image to a number of sub images and uses available rendering servers to render them.

In a multi processor system, the program automatically examines the number of available processors and activates the respective number of rendering threads. The full power of the computer is used by default.

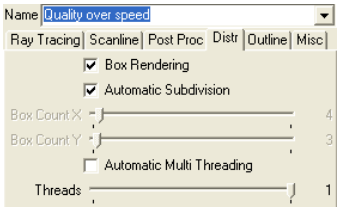
Sometimes it may be necessary to change the number of rendering threads. For example, you may want to leave one processor free for other tasks, or perhaps a complex scene requires so much memory that only one processor can be used.

To change the number of rendering threads:

1.Go to the Rendering Settings tab in the Select window and double click the desired rendering settings object to open the property window.



Double click the desired rendering settings object



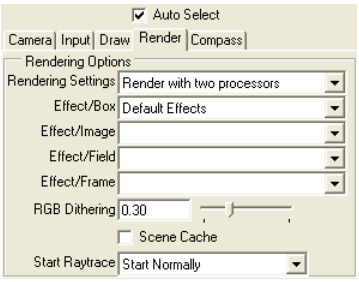
Set Threads to 1 to use a single processor

2. In the Distr tab, clear the Automatic Multi Threading option. Then set the Threads slider to a suitable value. For example, set Threads to 1 in order to use only one processors.

3. If you use multiple processors, make sure that Box Rendering is checked and that the number of boxes exceeds the number of processors. (Box Count X times Box Count Y >= Number of processors)

4. Drag & drop the rendering settings object from the select window into the view window. Click Render. If you use several threads, you can see how several render boxes become rendered simultaneously. If only one thread is used, boxes become rendered one after another. You can also select a suitable rendering settings object for the view using the view property window.

The file render window contains a similar rendering settings selector allowing you to use multiple processors to render images and animations to a file.



A view window uses 'Render with two processors' settings object for rendering

Configuring distributed network rendering

Distributed network rendering means using several workstations connected through a network for rendering.

Before you can use other workstations for rendering, the distributed network rendering system must be configured. The following things must be prepared for every workstation in the rendering network:

1. TCP/IP protocol must be installed and running. Unfortunately, this is something Realsoft 3D can't set up automatically for you. Consult your network administrator if necessary.

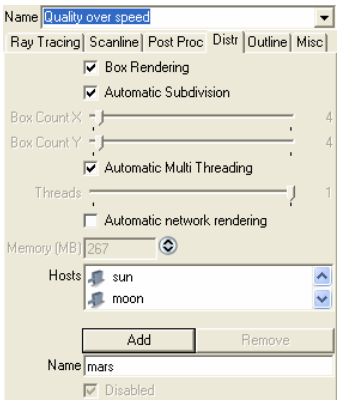
2. There must be a RendD rendering daemon (or the daemon version RendW which has a control window) running on every workstation that you want to use for rendering. You can install Realsoft 3D using the Network option to obtain a minimal render server configuration. The full program setup naturally contains all required components as well. RendD is a program, which listens to incoming rendering requests and spawns rendering server threads. It may be a good idea to insert RendD into the startup folders of render servers. Then the rendering daemon is automatically activated when a server machine is turned on.

Using network rendering

Once you have prepared the workstations for network rendering, you can ask Realsoft 3D to use one or more of them for rendering.

1. Define the host names you want to use for rendering in the Rendering Settings/Distr tab. If you have not defined the machines as "host" names then use the IP addresses instead. In a Windows XP network, you can use computer names (you can check and change the name in Control Panel/System/Computer Name tab). For example: a machine is called 'sun' and sun's IP address is 10.0.0.5. You can use either the host name 'sun' or the IP address of 10.0.0.5.

2. Check and if necessary, adjust the render box count. By default, the program estimates a suitable box count automatically. By turning off the Automatic Subdivision option, you can change the box count using the Box Count X/Y sliders. The number of boxes should be higher than the sum of threads and servers so that a single image can be divided to all threads.



Network rendering setup. In addition to one local rendering thread, three servers are used for rendering: moon, mars and sun.

Optimal box count depends on several things, like the speed of the network, available memory, resolution of the image and performance of the rendering servers. If all the systems are equally fast, the optimal speed might be achieved by setting the box count to a multiple of the sum of threads and servers. In the example image above, such box counts are 4, 8, 12, etc. A high box count adapts better to speed differences, but requires slightly more managing work from the program. Finding the optimal configuration may require some experimentation.

3. Drag & drop the desired rendering settings object into the view window and render. You should now see several boxes being rendered simultaneously in the View window.

Symmetric multiprocessing and network rendering can be used simultaneously. In other words, you can use the processors of your local workstation as well as processors in other workstations of the TCP/IP network.

Rendering over internet

When using rendering servers over an internet connection, data transfering often becomes the bottleneck. The performance can be optimized by configuring a hierarchical rendering distribution tree. For example, consider a situation where you want to use your local rendering network and another remote network. The actual computers (their symbolic host names) may be as follows:

- *local_dp_1:* local dual processor system, the workstation you are working on
- *local_sp_2:* local single processor system, a render server
- *local_sp_3:* local single processor system, a render server
- *remote_sp_1:* remote single processor system, a render server
- *remote_dp_2:* remote dual processor system, a render server
- *remote_dp_3:* remote dual processor system, a render server

1. The remote network should be configured as follows:

- remote_sp_1: start RendD just with the default parameters
- remote_dp_2: start RendD with default parameters
- remote_dp_3: start 'RendD remote_sp_1 remote_dp_2'

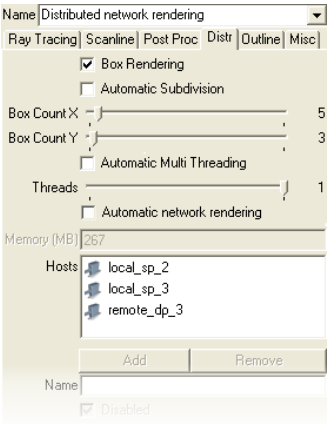
In above, the dual server (preferably the fastest of the remote servers) is started so that uses its both processors and also connects to the two other computers on the remote LAN.

2. The local render servers are started as follows:

- local_sp_2: start RendD with the default parameters
- local_sp_3: start RendD with the default parameters

3. The Distr tab render settings of the actual rendering client 'local_dp_1' are the following:

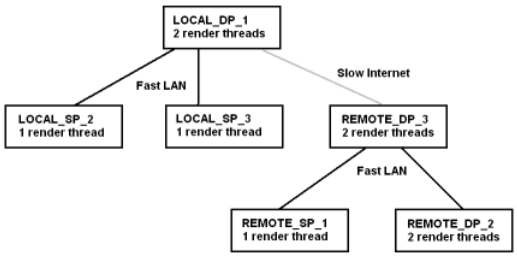
- Threads=2 (or Automatic Multithreading enabled), because there are two processors in the client itself to utilize
- The Hosts list is remote_dp_3, local_sp_2, local_sp_3



Now the rendering client uses 5 render engines (2 threads + 3 network servers), so the render box count should be at least 5. However, since the remote network with several processors is probably much faster than the local single processor render engines, it is best to use high box count that adapts to the speed differences. Multiplying the thread count by 4 might give a good result:

- Box Count X = 5
- Box Count Y = 4

Now, if the internet connection works fast enough, the remote network may render 6-8 boxes and the local network the remaining half of the boxes. The following diagram illustrates the structure of the rendering distribution:



For rendering farms, which have many workstations or whose configuration changes frequently, we recommend use of the Public Rendering Service. See the Reference Manual for details.

Lighting

A light source is a geometric object which radiates light of its own color. If a light object is black, it doesn't radiate light at all. A white object radiates all the main components (Red, Green, Blue) of light in the same amounts.

In real life, it is usually rather difficult to observe objects in lighting which have only one wavelength. Also, there isn't a material which would reflect only one wavelength. In Realsoft 3D, all this is possible, so you should be careful when selecting light source colors. For example, a totally red object isn't visible at all under blue lighting, because a red object doesn't reflect any blue light. A violet object looks red under yellow lighting.

Usually, it is advisable to create white light sources, so all objects will be rendered in their 'true' colors. Although the number of light sources is unlimited, they shouldn't be used indiscriminately. The time taken by rendering is greatly dependent on the number of light sources.

Light sources don't show directly in shaded pictures. So, if you want to see lamps or reflections of the light source on reflecting surfaces, you must put a cover made of, say, matte glass around the light source.

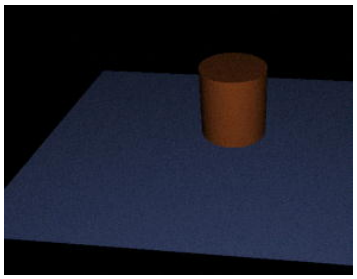
Unlike material mapping objects, light sources affect the whole scene regardless of their position in the object hierarchy.

Shadows on a floor

In the next example, we will test different light sources. The example includes a cylinder floating above a rectangle (floor), and some light sources casting shadows onto the floor (see the figure below).

Tutorial level: Beginner

Example project: 'tutorprojects/rendering/lightsources'



The first rendering of the test scene

1. Take a top view. Create an analytic rectangle (the floor), almost filling the whole view.

2. Create an analytic cylinder in the middle of the rectangle.

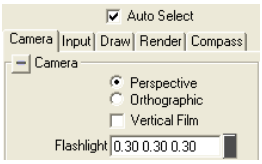
3. Go to the front view. The objects lie at the same level. Select the rectangle and move it to the bottom, then select the cylinder and move it above the rectangle. You may also change the colors of the two objects if you like.

4. Rotate the view upwards. Hit the '0' (zero) hot key to render the view.

The camera flashlight

The rendered image above shows that objects float in empty black space. However, we can see them - where does the required light come from? The answer is from the camera that took the picture. All camera objects, including view window's internal camera, include a flashlight.

The flashlight is quite a handy feature: it makes objects visible without actual light sources and also adds a suitable fill light to your images. Flashlight comes from behind the camera and therefore illuminates every visible place of the scene. To adjust the flashlight:



Changing the flashlight color

5. Select View Properties.. from the view window's popup menu. Go to the Camera tab of the view property window and enter a new value '0.3 0.3 0.3' to the Flashlight field. Then close the view property window.



Note

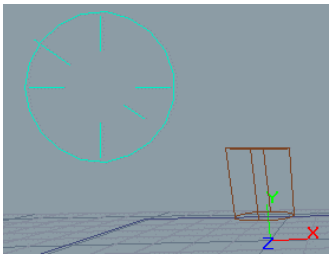
This changes the flashlight of the view window. In file rendering, the flashlight comes from the current camera object. Therefore, when preparing a real project for file rendering, remember to assign flashlight changes to the camera object after finding a suitable brightness level using view window rendering tests.

Now the flashlight is quite dim and we need some light sources to see the objects properly.

Adding a point light source

6. Take a front view. Go to the Light Sources tool tab and click the Point icon. Then click on the top and left side of the cylinder in the view window. When you now move the mouse, you can see a radiating circle. The radius of this circle will later define the intensity falloff of the light. Move the mouse until the circle includes the cylinder and most of the floor. Then click again to finish the creation.

7. Adjust the viewing angle and render the view.

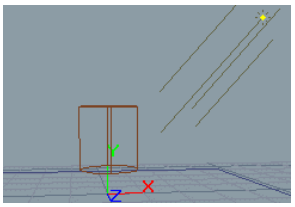


Creating a point light source

Now the point light casts a shadow of the cylinder to the floor. The shadow is 'perspective' - its width grows towards the top of the cylinder. There is another kind of light source available, which sends light from infinitely far away - namely the distant light source. All light rays from a distant light source are parallel and therefore its shadows do not have any perspective distortion. Let's experiment with the distant light source next.

The distant light

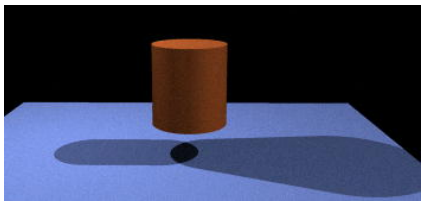
8. Take a front view. Activate the Distant light source tool from the Light Sources tool tab. Define the direction of distant light ray with two clicks on a view window. For example, draw the direction line down and left as shown in the example image.



A distant light source added

Note, that the length of the axis is not important, nor the position of it. You can even place the distant light source below the floor, and it still illuminates the upper side of the floor! The *only* relevant geometric property is the direction of the rays.

9. Render the view again (e.g. hit the zero key).



Shadows from a point light and a distant light

Soft shadows

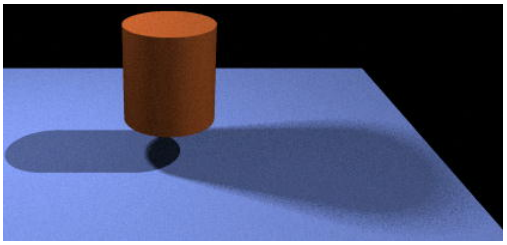
Now you should see two shadows of the cylinder on the floor. The edges of the shadows from both lights appear to be sharp. Do this to see a soft shadow:

10. Select the Light point object from the select window. Open the property window and go to the Spec tab. Here you can see the point light specific object attributes. Enter a non-zero value to the Size field, for example 0.05 meters.

Note that five centimeters is quite a realistic size for a light source. The bigger the size, the smoother the shadow boundary becomes. On the other hand, the distance from a light source to the target also affects the result - the sun, despite of its huge size, casts sharp shadows, because it is so far away from the earth.

11. Set the Quality slider to the level 6. This slider controls the rendering accuracy of the light source: the higher the value, the better the rendering quality. However, improved quality means longer rendering times. If the smooth area on the shadow edge is wide, the quality setting has to be relatively high. Only slightly smooth shadows can be computed using lower quality. Usually values from 3 to 10 are suitable.

12. Render the view.



A sharp and a smooth shadow

This time one of the shadows has a smooth edge. Rendering is slower, but probably the extra realism is worth the time penalty.

Tip: Using a high level of anti-aliasing (especially stochastic anti-aliasing) also increases shadow quality. Therefore, the point light quality can often be relatively low in final high-quality rendering.

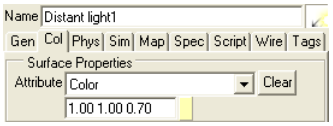
Color and intensity

So far, we have not paid any attention to the brightness of the inserted light sources. The brightness is defined by the usual object color and so called *Intensity* which is a light source specific property. The total brightness is a combination of these two:

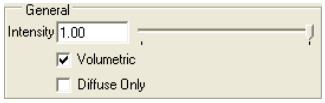
Intensity=0.5, Color=(1, 1, 1) is equal to Intensity=1, Color=(0.5, 0.5, 0.5).

We can test the effect of these attributes in our sample scene:

13. Select the Distant Light object. Open the property window and go to the Col tab. Since distant lights usually simulate sun light which is slightly yellow, change the Color of the distant light to pale yellow (1, 1, 0.7).



Changing the color of a light source



Defining light intensity

14. Go to the Spec tab. Since sunlight is quite bright, increase the Intensity from the default value 0.5 to 1.0.

15. Render the view.

Since the distant light was only slightly yellow, it can be difficult to see the effect of the color change in our colored test objects. Compare the color inside shadows - that shows the effect best. Remember that when trying to achieve a certain atmosphere in an artificial world, even subtle changes can be significant.



Note

The intensity value is not limited to 1. In the real world, the brightness of objects can vary enormously - the sun is millions of times brighter than a candle flame. Realsoft 3D is designed to deal with such differences. However, computer monitors or printed images can not duplicate this kind of phenomena. Therefore, use high intensity light sources with care: The result can be an over-exposed image showing plain white.

Controlling the falloff

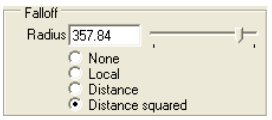
The two light sources of our test scene illuminate the whole universe. If you insert a new object 100 kilometers away from the cylinder, it is as brightly illuminated as the cylinder. This makes sense for the distant light source, which is almost infinitely far away. However, it is quite a strange property for a point light, which usually simulates light sources such as a light bulb.

Let's enhance the strange behavior of the point light:

16. Select the point light source, open the property window and go to the Spec tab. Increase the Intensity value to 1.0. Then render the view to see the result.

Let us analyze the result a bit: The floor now looks almost white. The color of the floor remains the same, but it is becoming so brightly illuminated that our 'film' (=computer monitor) gets over-exposed. The right edge of the floor is slightly darker - not because the point light is weaker there, but because light rays from the point light hit the floor at a smaller angle. We can compensate for the increased lighting intensity by selecting a suitable *light falloff* distribution for the point light.

17. The point light being selected, go to the Spec tab of the property window. The Falloff part of the window contains a falloff type selector. Select Distance squared. This means that intensity of the light decreases by the square of the distance - a point twice as far as the cylinder will receive only one fourth of the light.

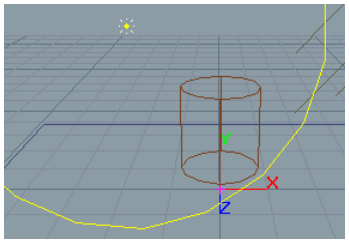


Select 'Distance Squared' falloff

This kind of falloff is physically correct for very small point-like light sources. If the light Size is greater than zero, Distance can be give a more realistic result.

Now render the view. The extreme brightness of the floor should now happen only close to the point light source. The bright area can be shrunk further by decreasing the Fallof radius of the point light. The initial falloff radius was defined by the circle shown by the point light tool (see the step 6 above). You can now change the radius using the controls in the Spec tab, or interactively as follows:

18. Make sure that the point light is selected, and hit the spacebar to enter the Edit mode. The point light source now shows a yellow circle in the view window. Move the mouse pointer over the circle so that the pointer changes. Then drag the circle towards the actual position of the light source, until the circle is about one half of the original size. Release the mouse pointer and hit the spacebar to exit the edit mode.



Editing the falloff radius of a point light

In the case of Distance and Distance squared falloff types, the falloff radius defines the distance where the light intensity is halved. The yellow circle edited above shows quite intuitively how the light decreases in your scene.

19. Render the view. The image brightness has returned to a suitable level.



Note

In Local falloff type, the circle shows the distance after which light intensity is zero.



Note

The distant light source does not (and can not) have Falloff controls. The intensity is constant everywhere.

Ambient light

The two shadows overlap slightly in the example scene. The area of overlap appears very dark, because only the weak flashlight illuminates it. If the viewing angle to the floor is small, the effect of the flashlight almost disappears.

The *Ambient* light source can be used to add a uniform base lighting everywhere in the scene. Ambient light simulates light that scatters from the walls of a room, from clouds, etc. - such light does not have any specific direction and it illuminates all parts of the scene.



Ambient light

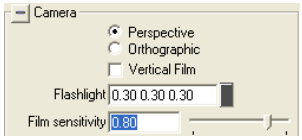
20. Click the Ambient tool of the Light Sources tool tab. Then click two points on the view window. The position and the falloff radius are not important this time (an ambient light can also have a falloff distribution, but we will not use this feature in this tutorial).

21. The default intensity of the ambient light is also 0.5, which is quite high, since we already have many light sources. Change the Intensity of the ambient light to 0.2 in the Spec tab of the property window. Render the view - the shadowed areas are now better illuminated.

Tip: one or two point light sources with 'Cast Shadow' option disabled often adds a more interesting base lighting to the scene than a single plain ambient light.

Controlling the overall brightness of the image

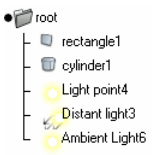
At this point, the image is again quite bright. What if we want to decrease the brightness? Do we have to laboriously edit the intensity of each light source? Fortunately, the answer is no - there is an easier way. We can change the sensitivity of the film:

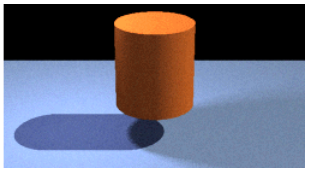


Decreasing Film sensitivity

22. Open the view property window from the view's popup menu, and go to the Camera tab. Decrease the Film sensitivity value from 1 to 0.8.

A new test render shows that the image is now 20 % darker than before.





The final result

The special light

The special light source object allows use of light sources of an arbitrary shape. This light source is a level object, which turns all sub objects into light sources. The sub objects can be any geometric objects such as NURBS curves or meshes.

Tutorial level: Medium

Example project: 'tutorprojects/rendering/speciallight'

1. To get started, model the cylinder-floor scene of the previous tutorial without the light sources, or delete the three light sources after finishing the previous example.

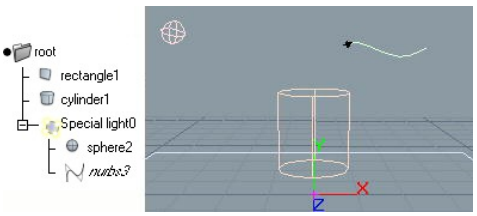
2. Click the Special light tool of the Light Sources tool tab.



3. Switch to the Analytic tab and create an analytic sphere to the left side and above the cylinder. Open the property window and change the color of the sphere to a pale red: '1.0 0.5 0.5'.

4. Create a NURBS curve to the top right side of the cylinder. Using the property window, change the curve to a pale green: '0.5 1 0.5'. Then go to the Gen tab and clear the Invisible in photorealistic rendering option. This is important - otherwise the curve will be ignored in ray tracing!

5. Select the sphere and the curve and drop them into the sub hierarchy of the Special light object.



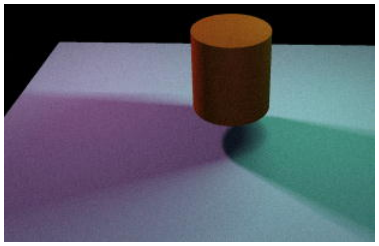
The structure of the special light example

The object hierarchy and geometry should look as in the example image.

6. Select the special light source and go to the Spec tab of the property window. Since we have only one light source included, increase the Intensity from 0.5 to 1.0.

7. Adjust the Quality value to about 10. The rule of thumb for this quality setting is the same as for the point light source: the bigger/closer the light source, the higher quality settings will be required.

8. Render the view to see the result. All shadows have smooth edges. Note also that the color of sub objects of the special light defines the color of the light.



Shadows created by a special light source



Note

Sub objects of a special light source do not define any visible surface - use object duplicates for this purpose.

Volumetric Rendering

Volumetric rendering allows you to visualize fog, smoke, fire and other atmospheric effects. For example, a light source can illuminate a fog cloud. Objects can cast shadows through a fog cloud, just like in the real world.

Defining such effects is simple and logical: first a suitable volumetric object shape is created, then a suitable material is applied to it. The program takes care of the rest. In other words, properties of foggy and gaseous objects are defined and controlled exactly the same way as properties of solid objects: using materials.

The Materials chapter explains how to build fog-like materials. This section demonstrates how you can use them in your scenes.

Defining volumetric properties of geometric objects

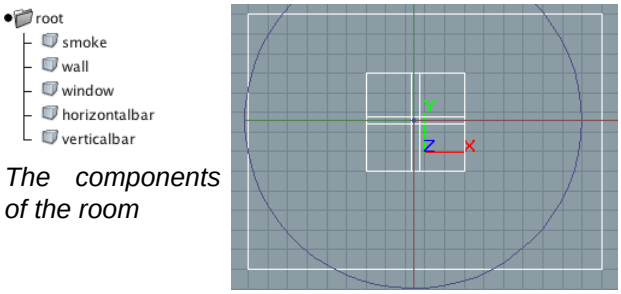
In the first example we will create a dark room filled with smoke. Light shines through a window, illuminating the smoke and casting shadows through it.

Tutorial level: Medium

Sample project: 'tutorprojects\rendering\smokyroom'

1. The first task is to model the room, or actually only one wall with a window, which is enough to show the effect. Take a front view and turn grid drawing and snap to grid on.

2. Go to the Analytic tab and select the analytic cube tool. Draw a large cube filling most of the view window. Using the select window's Rename tool (you can find it from Geometric Objects tab popup menu), rename the cube as 'smoke'.



The components of the room

3. Create 4 more analytical cubes as shown in the example image. The first new cube should have the same size as the smoke cube, because it will be the back wall behind the smoke. The second cube is the window in the middle of the wall. The third and fourth cubes are the window bars. Rename the cubes as you create them to make identification easier.

4. Take a side view. All the cubes we just created overlap each other and their thickness is not correct. Select smoke, and using either the Extend tool or the scale handles, increase the thickness of the smoke so that it fills at least half of the view window. Then move it to the right side of the other cubes either using the move handle or the Move tool.



Note

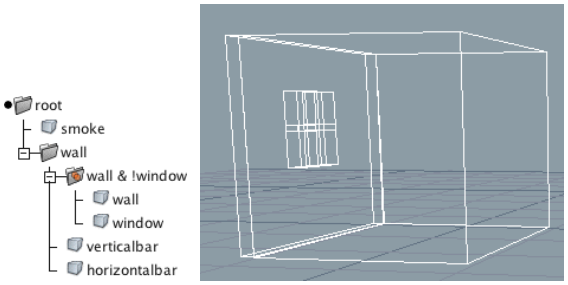
Leave a thin gap between the smoke and the other cubes. Always avoid situations where two surfaces (the smoke and the wall in this case) overlap exactly. Otherwise the rendering engine gets inconsistent results in computing where the fog ends and the wall starts. Remember that computers have finite accuracy. The gap can be very small, only a fraction of one millimeter. You may have to disable grid snapping temporarily to be able to position the smoke at a small distance from the wall.

5. Then select wall, window and the two window bars. Move them into position, if necessary. Use the extend tool (or the scale handle) to make them somewhat thinner. Remember to maintain a suitable thin gap to the smoke.

6. Select the window and increase its thickness slightly, so that it sticks out from the wall. We will use it for cutting a hole to the wall, and it should cut all the way through.

7. Select wall, then holding the Shift key down, add window to the selection. Then activate the Boolean tool from the Boolean tab of the toolbar. Click on the Remove icon, then click Accept. Now the window cuts a hole in the wall.

8. The default wireframe representation of the window is quite bad. If you like, you can improve it in the following way: Select the wall & !window level. Then open the hierarchy of the level, select the original wall and window objects and clear Invisible in real time rendering property from the Gen tab. Close the property window.



9. Let's clean up the hierarchy a bit. Multi select the wall & !window level and the two window bars. Then apply Drop to a Level from the popup menu. Rename the created level as 'wall'. The structure of the scene is now almost ready.

Room finished. There is a small gap between the smoke cube and the back wall.

10. Select the smoke object. Go to the Materials tab of the Select window and drag & drop the shadowfog material onto the view window.

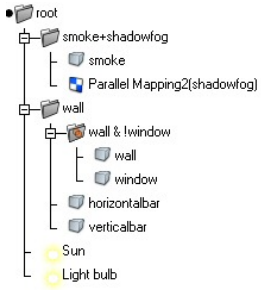
11. Take a side view and zoom out. Go to the Light Sources tab and activate, for example, the Point tool. Click behind the room so that the light will shine through the window to the room (don't put the light above the room, because there is no ceiling). Click towards the room a second time to finish the light source. Rename it to 'Sun'.

12. Make sure that the view uses reasonable quality or quality over speed rendering settings. If not, drop either of them or another suitable rendering settings object to the view. Take a perspective view. Zoom and rotate the view so that you can see the interior of the room in a suitable way. Then click the Render button.

13. It may happen that you do not see the fog, or the fog looks much too bright. The reason is that shading of most fog like materials strongly depends on the scale of the model. A big cloud in the sky can be several kilometers thick and can hide a bright object like the sun. A smoke from a candle is a couple of centimeters thick and hardly visible at all. To change the strength of the smoke shading, select the shadowfog material in the Material tab of the select window, and open the property window. Increase the Turbidity property to make the fog thicker. You may also change the Color of the fog to light blue.

14. The accuracy of the shadow appearing in the smoke can be improved by increasing the Sampling value in the property window of the material. This may be necessary if the object casting the shadow has small details (such as thin window bars). For example, set the Sampling to 20. The higher the value the better the quality (and the slower the rendering). Advanced users may add random variation to the sampling positions using VSL (see 'additive fog' VSL example for details). Together with adaptive anti-aliasing, this method produces accurate high quality shadows with a reasonably low sampling rate.

15. If the smoke still appears too weak, you can increase the brightness of the light source. Select the sun light and use the Intensity slider in the property window's Spec tab. Try values ranging from 1 to 2. You may also create a new, weaker light source on the top of the room to simulate a light bulb in the ceiling.



Smoky room finished

The example project 'smokyroom' uses the randomized sampling method mentioned above for improving shadow accuracy. You can re-use the material in other scenes. To render a high quality still image, select Stochastic anti-aliasing at level 2 or higher and set anti-aliasing threshold to 5 or lower. This ensures that fog effect computations are made very accurately on areas where fog illumination changes rapidly.

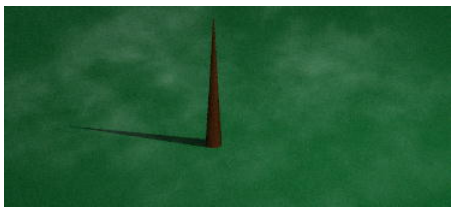


Fog clouds

Tutorial level: Medium

Sample projects: 'tutorprojects\rendering\fogcloud' and 'tutorprojects\rendering\irregularcloud'

You can simulate a fog cloud the same way as smoke in the previous example. Often a fog cloud is not homogeneous: there are dense areas and clear areas quite randomly. This effect can be achieved in two ways:

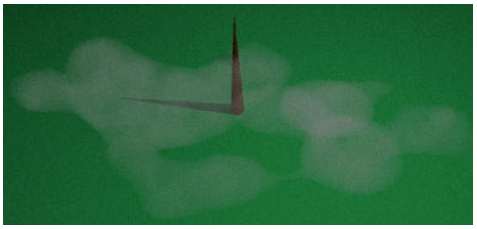


A simple elliptic fog cloud. Irregularity in fog density hides the simplicity of the shape

1. Use one big object (e.g. a flat ellipsoid) to model the fog cloud and apply a material, which defines a varying density. The density function must be carefully designed to hide the symmetric shape of the ellipsoid. The adjacent image shows an example.

2. Model an irregular shape using metaballs or subdivision surfaces. Remember to set high enough recursion depth under the renderings settings tab (10 in the example image).

The fog material can be self-illuminating ('plasma') or diffuse-illuminated ('shadowfog'), or the combination of the two.



A fog cloud consisting of metaballs

A note for users of previous Realsoft Graphics Oy products: In Real 3D, overlapping areas of transparent objects, which shared the same material, were automatically melted together. This does not happen in Realsoft 3D. For example, using a large set of analytical spheres for representing an irregular fog cloud does not produce a good result, because the thin seams between spheres usually affect the shading. Use the metaball object for this kind of applications.

Global atmospheric effects

Global atmospheric effects, such as a global fog, can be created easily with the Creation/Atmosphere tool. This tool creates an object, which defines volumetric properties for the entire (infinite) space.

Tutorial level: Beginner

Sample project: 'tutorprojects\rendering\globalfog'

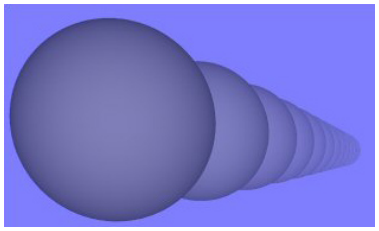
To create a global fog:

1. Activate the Atmosphere tool from the Light sources toolbar tab.

2. Select a fog material from the material list in the tool control bar.

3. Click Accept. You have now defined a global fog.

4. Create a couple of spheres at different distances and render. The higher the distance, the better the fog obscures the spheres.



The Atmosphere tool and a sample scene rendered



Note

Use only self-illuminating, homogeneous fog materials with the global fog. The shading of other kind of fogs is performed with a finite amount of samples, which are not suited to the infinite nature of the global fog object.

Visible lights

The spotlight tool allows you to select a volumetric material, which will visualize the light beams.

Tutorial level: Beginner/Medium

Sample project: 'tutorprojects\material\visiblespot'



Spot Light tool

1. Go to the Light Sources tab of the toolbar and activate the Spot Light tool.

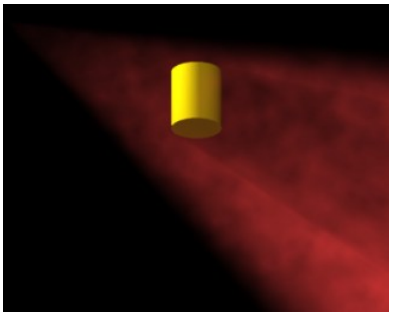
2. Select one of the materials available in the material list of the tool control bar. The list shows only those materials of the current library, which define volumetric effects. If the list is empty, all such materials have been deleted, and you should reload the standard material library. Select shadowfog.

3. Define the spot light geometry as usual.

4. Place an object inside the light cone so that it will cast a shadow.

5. Render. If the light beam is not visible enough, double click the shadowfog material in the Material tab of the select window and double the Turbidity value in the opened property window. Test render and increase Turbidity again until a suitable effect is achieved.

6. The sample image shows growing brightness as the thickness of the beam increases. This can be changed by selecting Falloff = Distance or Falloff = Distance squared from the spotlight's Spec controls on the property window.



Fog makes spotlight's beam visible

If you now look at the select window's object geometry tab, there are no material mapping objects at all. The reason is that a spotlight can have a material as a direct attribute. If you open the property window and go to the spot light's Spec properties, you can see that there is a Material gadget available. From the gadget you can change the material or disable it by selecting the empty entry of the list.

In the sample project, 'tutorprojects\material\visiblespot', an additive fog material, which is described in the 'Fog Examples' section of the VSL tutorials, is applied to a spotlight. Because of its additive nature, the illuminated fog never darkens objects behind it and is therefore especially suitable for lighting simulation.

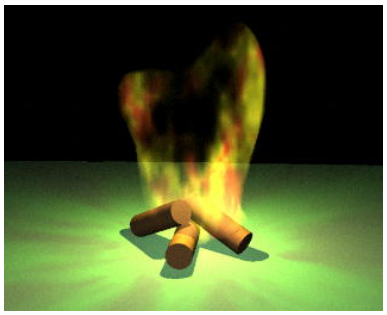
Volumetric fire

Fire can be simulated using suitable materials. Let's experiment using the Fire material in the default material library.

Level: Medium

Sample project: 'tutorprojects\rendering\simplefire'

1. Create a green analytic rectangle defining the ground level and rename it to 'Ground'. Create a pile of brown analytic cylinders representing firewood above the ground. Drop the cylinders to a level and rename the level to 'Firewood'.



Volumetric fire material

2. We could use a simple elliptic sphere to define the volume of the sphere. However, a more interesting result can be obtained using metaballs. Go to the geometric objects tab of the select window and select New/Metaball from the popup menu.

3. Double click the created metaball object in the select window to open the property window. Go to the Spec tab and set each of the Resolution X, Y and Z sliders to a value between 10 and 15. The surface of the metaball structure does not have to be smooth because only volumetric shading will be applied to it. Decreasing resolution will improve rendering speed. Set Field to Quadric (smooth blending of metaballs) and Iso Value to 0.4 (to fill most of the volume of metaballs). Activate the Simulation checkbox on the top part of the Spec tab. We will animate the metaballs using a gravity field later.

4. Create an analytic sphere above the firewood. The size of the sphere may be one quarter of the size of the total fire. Create 3-6 more spheres around it at random positions. The spheres should partially overlap. You may stretch some or all of the spheres vertically higher so that the shape looks more like a flame.

5. Select the first sphere. Open the property window, go to the Sim tab and set Gravity to Cause. The center sphere will create a gravity field in which the other spheres pulsate. Switch to The Phys tab and set mass to 100 000 000 (number 1 and eight zeros = one hundred million) kilograms. A gigantic mass like this is necessary, because Realsoft 3D uses real world units in simulation. Two apples on a table do not attract each other much, do they? Only a planet-scale mass is sufficient to create a noticeable gravity field.

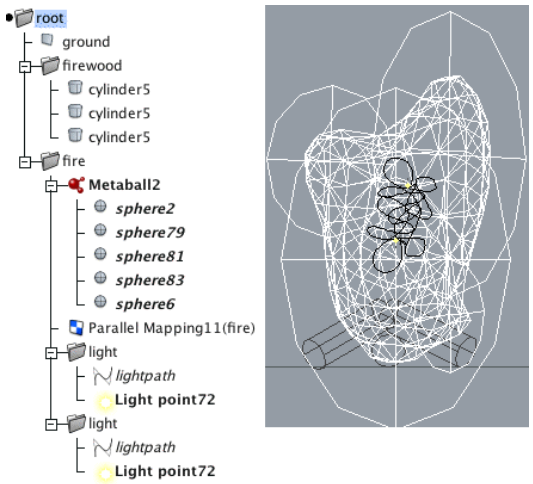
6. Select all the other spheres. Their default masses are OK, but go to the Sim tab and set Gravity to Affected. These lightweight spheres will bounce around in the gravity field caused by The heavy center sphere.

7. Select all the spheres, and drag & drop them to the metaball level on the select window. An irregular meta surface appears. Edit the spheres inside the metaball level if the shape is not good enough. After that, if you like, you may set the actual spheres Invisible in real time rendering in the property window Gen tab, to clean up the wireframes.

8. Select the metaball object. Drag the material Fire from the Select window's Materials tab to the view window. This creates a new level, rename the level to 'fire'.

9. Increase Recursion depth of the rendering settings used to at least 6 and test render to see how the fire looks. If the brightness of the fire is too low, open the property window and increase the Flame saturation value of the fire material. The Flame density attribute defines how big part of the fire volume becomes filled with flames. Too high a value fills it completely, and reveals the original shape of the fire volume clearly. This is undesirable if the volume is relatively simple. You may also adjust the size of the fire parallel map to change the size of flames appearing in the material.

10. The fire is supposed to cast light to its environment. Therefore, create two point light sources inside the fire. Change the color of light sources to a suitable light yellow. One of the light sources can be non-shadow casting (property window's Spec tab/Shadow Casting set to No shadows) to save some rendering time. Create two closed nurbs curves traveling around in the fire (see the figure below). Multi select the first light source and the first nurbs curve (in this order). Then select Animation/Path/Move from the main menu bar. Then drop the light source and the curve to a level. Do the same for the second light source and curve. The light sources follow the paths, casting changing shadows around (If set back on).



The final structure of the fire scene

The simple fire scene is now ready. Play the animation to see animated fire. If the fluctuation of the metaball shape is too slow, increase the mass of the center sphere. Decrease the mass if things happen too rapidly. A ray traced preview rendering to an AVI file shows another animation effect: the material defined flames also change and travel upwards. You can control the speed of this effect using the Flaming speed attribute of the fire material.

The workflow above shows only the basic ideas for a simple fire. The realism can be improved in many ways: adding new material details, more metaballs, etc. In the 'simplefire' sample project the Volume properties shader was duplicated, changed to a Surface filtering shader and edited slightly. The new shader filters the color of the light sources inside the fire with the color of the flames. This makes illumination of the environment much more interesting and realistic.

Rendering settings and volumetric effects

Most of the default rendering settings are suitable for rendering volumetric effects. Only the draft quality settings (speed over quality) are too simple for this purpose. The general guidelines are:

- In order to render volumetric effects, the recursion depth must be two or higher. Recursion level three is usually the minimal sensible value. Complicated fog clouds (defined by metaballs, etc.) may require a recursion depth from 5 to 10.
- You can keep all the check boxes in the Detect frame activated. Because of internal optimizations, they don't slow down rendering when you haven't defined volumetric effects. You need to modify these options only if you have included volumetric effects in the scene and, for whatever reason, want to disable them.
- If the scene includes scanline particles, check the Visible in Reflections/Refractions/Fog option of the scanline rendering settings.

Motion Blur

The exposure of a film camera takes a certain time period. Non-static objects move and change their appearance during the exposure time. Because the film records everything that happens during the exposure time, such objects do not appear sharp but blurred in the picture. This phenomenon is called motion blur and can be simulated in Realsoft 3D.

Motion blur can be used as an effect to increase the realism of images. It also improves the quality of animations. Common video and film standards show 20-30 frames per second. At such frame rates and without motion blur, the animation of a rapidly moving object may not appear continuous. Various strobing artifacts may also become visible.

The effect of motion blur is visible (and necessary) only when the animation contains objects moving a certain amount per frame. As a general rule, an object should move at least 10 pixels between two frames before the blur produces noticeable results. The use of motion blur makes rendering slower. For example, 9 motion samples usually at least triple the rendering time.

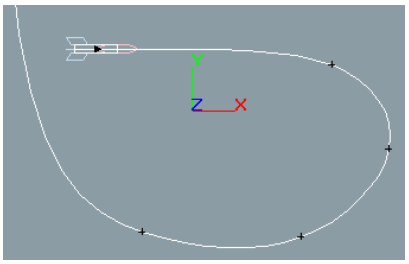
The motion blur feature of Realsoft 3D works by simulating physical reality accurately. Moving objects, shadows of moving objects, animated textures and materials become blurred if they change during the exposure of a single frame. If the camera moves itself, the whole world becomes blurred.

Rendering a motion blurred animation

When building a scene, you do not have to worry about motion blur. If you decide to use motion blur after the animated scene is finished, all you have to do is to set some options in the animation and rendering settings. The following example demonstrates this:

Tutorial level: Medium

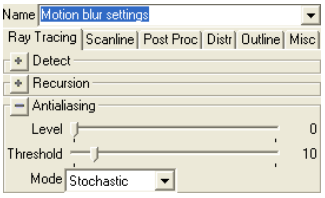
Example project: 'tutorprojects\rendering\missile'



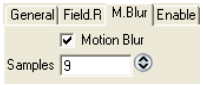
A missile follows a path

1. Create the animation in the usual way. For example, model a missile and draw a path (a NURBS curve) for it. Multi select the missile and the path and apply Animation/Path/Directed from the main menu.

2. Go to the Rendering settings section of the select window. Select the settings you plan to use for rendering. Open the property window from the select window's popup menu and go to the Ray Tracing tab. The rendering settings can be as usual, but set the anti-aliasing Mode to Stochastic and Level to zero. The motion blur system will take several samples per pixel over time, and therefore the usual anti-aliasing is not necessary. Only when using a minimal amount of motion samples, e.g. 4 samples between each frames, you should define some anti-aliasing, for example level 2. This step is important because it will save a lot of time in rendering the animation. Close the property window.



Defining anti-aliasing settings for motion blur



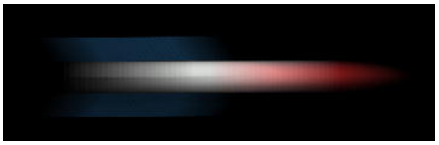
Activate motion blur and select the amount of motion samples

3. Open the Animation Settings window from the Windows menu of the main menu bar. Check the Motion Blur option and set Samples to 9. Nine motion samples are enough for most purposes, and does not slow the rendering too much. Close the window.

5. Let's preview the motion blur on the view window first. Drag and drop the rendering settings object, which was prepared in step 2, to the view window (this is not necessary if you know that the view already uses the correct settings). Open the view property window, go to the Render tab and activate the Continuous rendering option. Close the view property window.

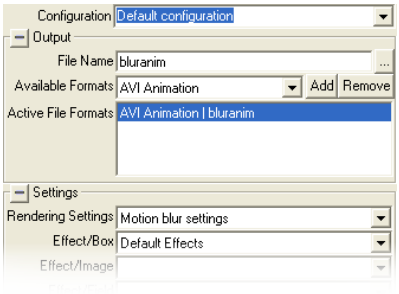
6. Open the animation window from the View main menu (if it is not already open). Hit the Play forward button. The program starts playing the animation, and renders the view after each time change because of the enabled continuous rendering option. The first rendered images look quite bad, because anti-aliasing is disabled.

Wait until nine images have been rendered. At that point, the program computes and displays a motion-blurred image in the view. You may stop the animation playback now. Open the view property window and turn off Continuous rendering option. Then rewind the animation. If the blur quality was not good enough (the missile moves too much between the motion blur samples) adjust the amount of samples to 16 using the animation settings window.



Moving missile appears blurred

7. The final step is rendering to a file. Select File/Render from the main menu. Prepare file rendering the usual way, like instructed in the manual section 'Rendering Still Images and Animations to a File', but remember to select the correct Rendering settings. Then start file rendering by hitting the Play button.



Rendering settings for motion blurred file rendering include stochastic 0-level anti-aliasing

Field Rendering

When very smooth animation is required, 25 frames per second (or even 30 fps) may not be enough. Extra smoothness of motions can be achieved by showing a new image in each video field. The PAL system, for example, shows 50 individual fields per second. Nevertheless, because of the interlaced nature of the PAL system, each field includes only one half of the vertical picture lines.

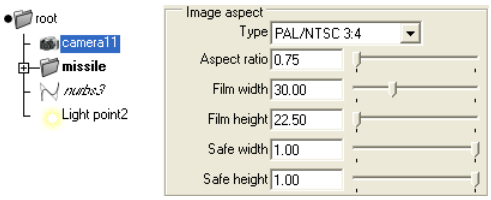
Therefore, rendering full video resolution images for field-rate recording means wasting one half of the rendering time and picture information.

Rendering fields to disk files

Using field rendering in Realsoft 3D is easy: build and animate the scene in the normal way. Then activate Field rendering from the animation options before starting the rendering. The example below shows how to prepare field rendering of an animation for PAL video recording.

1. Create the animation as usual. Use the normal amount of frames (25 frames for one second in PAL).

2. Insert a camera object to the scene using the camera tool, if not already inserted. Open the property window and go to the Spec tab. Set the Image aspect to PAL from the Presets list. Close the property window.



Selecting PAL camera aspect ratio



3. Open the Animation Settings window from the Windows menu of the main menu bar. Check the Field Rendering option.

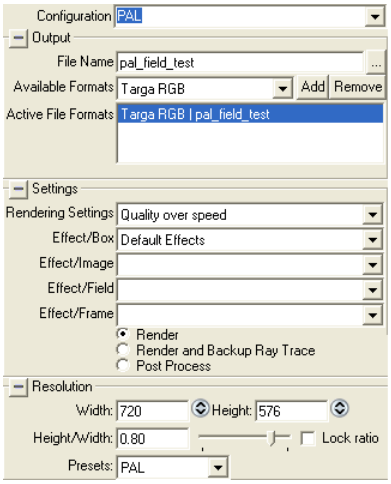
4. The animation settings window has also two other options for controlling field rendering. The options Shift latter field up and Interlace latter field above should be cleared if your video hardware defaults to the EVEN field first. If the hardware uses the ODD field first, set both check boxes.

5. The final step is rendering to a file. Select File/Render from the main menu. Select PAL from the configuration list. It already includes the correct image resolution.



Note

The resolution is the frame resolution, not the field resolution. Prepare file rendering the usual way, as instructed in the manual section 'Rendering Still Images and Animations to a File'. Then start file rendering.



File rendering options for PAL fields. The selected render settings object has appropriate field options.

Scanline Hairs

Scanline particles are ideally suited to small and thin objects such as hairs. The following tutorial demonstrates a typical workflow for creating hair:

Tutorial level: Advanced

Example project: 'tutorprojects/rendering/hairs'

1. Load in the object 'models/nurbs/simple head' (or model the head yourself).



Options for Uvimage tool



A haircut modeled using 2D particles

2. We start by texturing the hairy area of the head with a material that looks like hairs. This helps to keep the amount of real hairs reasonably low. Select the loaded head object. Go to the Materials tab of the toolbar and activate the UVimage tool. Define a suitable file name for storing the image, set color to black and press Accept.

3. Turn shaded OpenGL drawing on from the compass menu. Because of the attached black uvimage, the head appears black. Paint the area, which should be covered with hair, using the 3D painting tools (included in the Materials tool tab). Use the tools as follows:

- The paint color should be white. The image will be used later as a scope map, and white means fully hairy material.
- Paint the border of the hairy area with the airbrush tool to achieve a smooth gradient. You can reduce the hairy area by carefully brushing some black color.
- You can fill the hairy area using the Curve painting tool with the filled option set.
- It may happen that some painting appears on the neck when you paint the top of the head. Don't worry, this happens because the default UV space is not properly initialized. You can afterwards assign pointwise UV coordinates to the Nurbs model using the Define UV tool, which appears to the tool control bar when the head is selected. Then open the UV Editor window, set the head to edit mode, select the lowest point row from the view window and move it slightly downwards in the UV editor.



The hairy area painted white

4. After painting, select the head, open the property window and go to the Col tab. Clear the Texture field, which contains the name of the painted image, and hit Enter. Set the Surface Properties/Color gadget to a suitable skin color (e.g. light brown 1 0.8 0.7).

Materials for the hairy skin

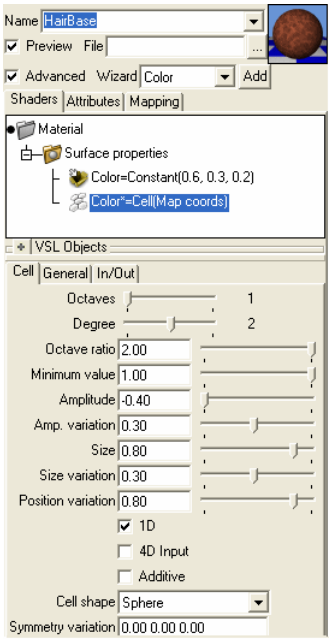
5. Next we define the materials for the hairy skin. Three materials are needed for this purpose:

- A material for hairy skin
- A scope map material which defines which part of the head is hairy
- A level material combining the two above mentioned materials

Change the select window to show the material library. Select twice New/VSL Material and once New/Level Material from the select window's popup menu.

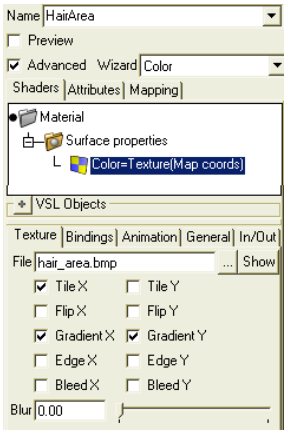
6. Select the first created VSL material. Rename it as hairbase. Enable Advanced options of the property window and add the following VSL objects to it:

- A Surface properties shader
- A Constant object that assigns a suitable hair color (e.g. dark brown 0.6 0.3 0.2)
- A Cell object, which adds irregular dark spots to the surface by scaling the color. Set Operation to Multiply from the popup menu. Minimum value should be 1 and Amplitude -0.4. This gives a range from 0.6 to 1 (in other words, darkest spots are 40 % darker than the base color). All Variation fields should be set to a non-zero value, so that the pattern becomes irregular. See the picture. Check the 1D option.

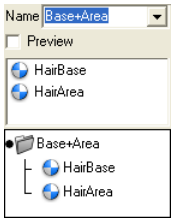


The base color of hairy skin is dark brown. Cell noise adds small darker spots to it.

7. Select the second blank VSL material. Rename it as HairArea. Add a Surface properties shader and a Texture object to it. Set the output channel of the Texture object to Surface:Scope and input to Surface:UV coords. Set the File name gadget to the name of the image, which was initialized in the step 2 above and painted later. Other options are shown in the example image.



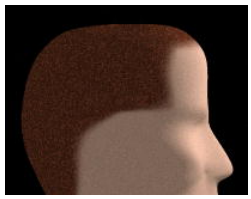
A scope map material for the hairy area



The level material combines HairBase and HairArea to one single material

8. Select the Level type material and rename it as Base+Area. Using the property window, drag & drop the materials HairBase and HairArea to the sub material list.

9. Having the level material (and the head object) still active, select Mapping/Parallel from the select window's popup menu. Hold down the shift key and draw a small parallel mapping cube over the head. The smaller the mapping, the denser the pattern. Test render the view window and adjust mapping size until the result is suitable.

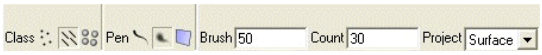


A pattern of tiny dots mapped to the head

Adding the hairs

10. Now we can start adding the actual hairs. Switch to the Geometric Objects tab of the Select window and select the original head object (not the level containing it). Go to the Wire tab of the property window and set Texture Quality to for example 128. Wait for a while until the HairBase texture pattern appears to the view window. Seeing the hairy area, which was already 3d painted to the head, will guide us where to add the hairs.

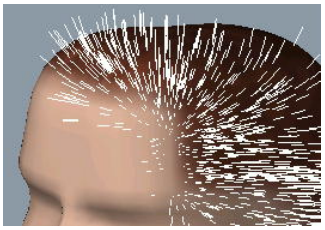
11. Zoom in so that the head fills most of the view. Activate the Particle tool from the Creation tab of the toolbar. Select airbrush painting, 2D particles, Line Width = 50, Count = 30 and Projection = Surface from the tool control bar.



Hairbrush options

12. Start painting the hair over the head by dragging with the left mouse button over the hairy areas. Rotate the view occasionally so that all areas get painted. Do not worry if hairs are added to the wrong place such as the nose - they can be removed later. On the contrary, brush a fair amount of hairs over the boundary of the hairy area, so that the boundary will have enough hairs. Click Accept when the hair is reasonably dense everywhere.

A suitable amount of hairs is 5000 -10 000. Using a very high amount can make screen refreshing slow and cause memory problems later in ray trace rendering. If this happens, it is good to know that using a high box count in box rendering decreases the memory requirements. You can check the amount of hairs from the property window's spec tab after creating the particle object.



Airbrushing hair particles onto the head. Note how hairs overlap the actual hair boundary at this stage.



Note

The tool inserts 2d particles perpendicular to the surface. It may happen that the surface normal points to the opposite direction from what was intended for hairs. If this happens, cancel the tool, take a top view and mirror the head. Mirroring inverts the normals. Then start the particle tool again.



Lasso selecting the hairs that are outside the hairline for deletion

13. Take a side view and set the created 2D particle object to its Edit state using the view's popup menu. Press the 4 key down and start dragging the lasso selector along the hair boundary. Draw a loop that includes all hairs that lie outside the hair-colored part of the head.

14. If the haircut is symmetrical, you can now click the Delete tool of the tool control bar. Otherwise take a top view and Alt-drag the hairs of the opposite side to remove them from the selection before applying Delete. Then repeat the selection and deletion from the opposite side.

15. Turn off the Edit mode. Make sure the particle object is selected. Go to the property window's Spec tab and select the Geometry sub tab. Pick Length from the long attribute list and set it to 0.01 meters (1 cm is pretty good

for this kind of simple haircut). Pick Diameter1 and set it to a small value like 0.001 (1 mm). Pick Diameter2 and set it to zero. You may try different values later but these should produce a reasonably good result.

16. Go to the Col tab and set the color of the hairs to the same color as you used for the hairbase material (0.6 0.3 0.2).

17. Test render and adjust the particle properties as required.

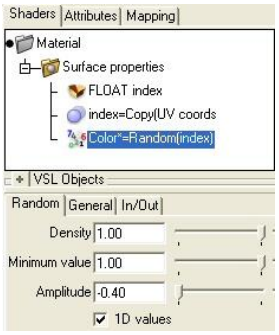
Coloring the hairs

Most hair types have a subtle color variation. When you grow older (or a software product release becomes badly delayed), gray hairs start appearing. Let us add this property to the hair next.

18. Switch to the Materials tab of the Select window. Select New/VSL Material from the popup.

19. Rename the material as HairColor and using the property window, add the following VSL objects to it:

- A Surface properties shader. Properties of scanline particles are normally defined in this shader.
- One Float type variable called index
- A Copy object that copies UV channel's last sub channel to the index variable, set Sub channel 1 = Z.
- A Random object with the following options:
  - Output = Surface:Color (because this material modifies hair color)
  - Input = index. The random variation is computed from UV channel's third component, which is the index of the hair.
  - Operation = Multiply (random factor scales the original tone)
  - Density = 1 (every hair has a slightly different random color)
  - Minimum value = 1, Amplitude = -0.4 (range 0.6 …1 makes darkest hairs 40 % darker than lightest hairs)
  - check 1D values



A material that randomizes color of hairs

20. Drag & drop the created material to the view window (hair particles should be still selected). A new test rendering finishes this tutorial.

Tips and tricks

- Curved and curly hairs can be modeled using Nurbs curves. Clear the Invisible in photorealistic rendering flag to enable scanline rendering. The Spec tab contains thickness controls for both ends of a curve. Lattice/Distribute and Lattice/Paste on Surface tools can be used for attaching curves into surfaces.
- The hair color material can also include a shader defining some specular shine. The sample material tutorprojects/rendering/shinyhair contains an example material. It uses the Surface:U direction channel, which contains the direction of hairs.



Shiny black hairs

Alpha Channel

Many video compositing related projects require use of an additional 8 bit mask channel, usually called as Alpha channel, which controls how rendered graphics is inserted over other background graphics. This tutorial introduces the basics of Alpha channel use in Realsoft 3D.

Tutorial level: Medium

Example project: tutorprojects/rendering/alpha_ready

1. Model a simple test scene of a magnifying glass floating in space or load the provided test scene 'tutorprojects/rendering/alpha_scene'.

2. We will render the alpha channel information to a disk file using a format which supports the extra channel. Therefore, go to the 5th tab of the select window; it contains the output object library. If the list contains a Targa object, LMB click it to select it. If not, apply the popup menu New/Targa. Open the property window and make sure that the Alpha option is set. You may also set the Compress option to save some disk space.

3. Open the file render window from the pull down menu File/Render. Add the Targa output to the Active File Formats list and remove other active outputs. Define an appropriate file name, for example 'my_targa_test'. Then select for example rendering settings Quality_Over_Speed. Resolution can be 600*450 pixels. Hit the render button and wait until the image is ready.

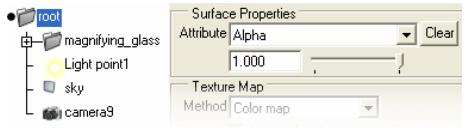
4. Start the Show Image utility program; you can find it from the Realsoft Graphics Oy program group of the Windows Start Menu. Use its File/Open pull down menu to select the rendered Targa file. If you used the example scene, the picture will look like in the adjacent example image.



The test scene rendered

Now use the Channels pull down menu of Show Image to view the Alpha channel. It is completely black! This is because we did not specify any Alpha values for objects. By default, the program processes no Alpha channel information at all, in order to speed up rendering in projects which do not use such information. Fortunately, it is very easy to assign the required Alpha values to objects, as we will see next.

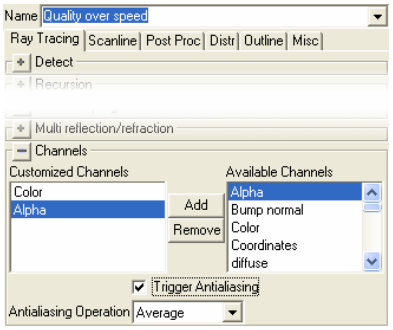
5. Switch back to Realsoft 3D and select the root level of the hierarchy of geometric objects. Open the property window and go to the second Col tab. Select Alpha from the Attribute list and move the slider below to the full value 1.0. Now all objects have this Alpha value, because the target was the root level. Only the empty background space renders with zero Alpha.



Setting Alpha value for the whole scene

The root level defined alpha can be overruled in deeper hierarchy as follows: open the hierarchy of the magnifying glass object, select the item 'lens' and change its Alpha to a semi transparent value 0.5.

6. Let's tailor the antialiasing settings for optimal alpha channel quality. Switch to the Rendering Settings tab of the select window. On the property window, select the leftmost tab and open the Channels frame on the bottom. Pick Alpha from the Available Channels list. Hit the Add button so that Alpha appears into the Customized Channels list. Activate the inserted item and turn the Trigger Antialiasing option on. Now, not only color changes but also Alpha channel variations trigger improved sampling at object boundaries. This ensures good antialiasing for the Alpha channel.



Alpha channel Antialiasing controls

7. Go again to the File Render window and restart rendering. Then view the new image version in Show Image. Choose Alpha from the Channels menu to find out that now the image contains a nicely antialiased Alpha component.

More advanced information about Alpha channel use is included in the Compositing tutorial.



The Alpha channel of the final Targa image

Advanced Illumination Examples

Illumination Maps

Big part of real-world illumination is created by surfaces that reflect illumination from actual light sources. White walls of a room spread the light from the lamp in the ceiling to shadowed places such as behind the furniture.

The UVImage tool can evaluate and store this kind of indirect lighting to so called illumination maps. The following simple example shows how to use this feature.

1. Model a simple scene shown in the image on the right: a rectangular floor surface, one wall on the left and a table standing on the floor. Alternatively, you can load the sample scene 'tutorprojects/rendering/table'. The table consists of a cube and four cylinders. Place one point light source on top and to the right of the table. Open the property window of the light source and adjust its Intensity to level 1.0. You may also open the view property window and set Camera tab's Flashlight to black color (0,0,0). Then the only light source of the scene is the point light and shadows will be quite dramatic.



A table illuminated by a single light source

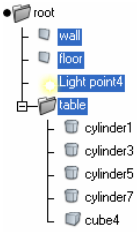
2. Next we generate the illumination maps. Go to the Materials tab of the toolbar and activate the UVImage tool. The default resolution is fine, but change the Type selector to Indirect lighting. If you have time to wait a bit more accurate computation (and better quality), increase the Samples value from 1 to 3 - 5.



UVImage options for indirect illumination

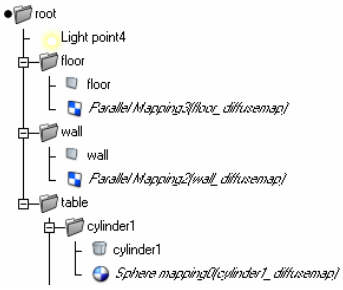
3. Select all objects for which you want to generate maps - in this case the wall, the floor and the table. Select also the point light source. This is important; surfaces have always two sides and the program must know for which side the illumination should be evaluated. The selected light source indicates it. Click Accept.

Wait until the progress indicator shows that the tool has finished. Now look at the select window: the object hierarchy has changed slightly because the tool has inserted new illumination maps to the hierarchy. The material library contains a set of new materials as well.

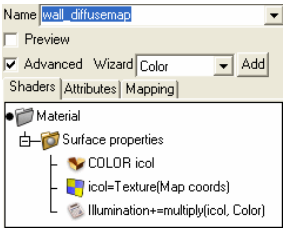


Object selection for illumination evaluation

4. Hit the render button to ray trace an image of the scene. The space under the table, which was previously totally dark, now has some subtle illumination. The indirect illumination is usually weak compared to direct lighting. If you wish to enhance it afterwards, you can simply reduce the intensity of the point light source to one half and double the film sensitivity of the view camera from the view property window.



The hierarchy with the illumination maps



The diffuse illumination material

You can naturally edit the diffuse illumination materials if you need to control illumination of a single object.

The tutorial example shows that illumination mapping is not part of the rendering process but a tool that changes the scene. This means that it is best suited for situations where it is necessary to make illumination of a static scene more realistic. The time consuming illumination maps are generated only once; after that, the camera can be animated to move in the scene. One can also add some animated objects whose contribution is not significant for indirect illumination.



Note

Before applying illumination maps to SDS objects, remember to initialize proper UV coordinates for them first.



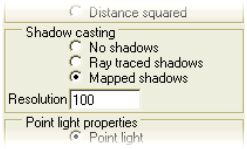
The table scene with indirect illumination maps

Mapped Shadows

Shadow mapping is another texture mapping based illumination method. Mapped shadows usually have nice smooth edges which increase realism without the high rendering time penalty of ray traced smooth shadows. On the other hand, shadow mapping often suffers from a number of accuracy problems. It may take several rendering attempts to adjust them until the desired result is achieved.

Let us continue experimenting with the table scene of the previous chapter. The indirect illumination maps may be included, but you can also reload the table scene 'tutorprojects/rendering/table'.

1. Select the point light source and open the property window. Go to the Spec tab. Then change the Shadow casting selector from Ray traced shadows to Mapped shadows. Then render the view window to see the result.

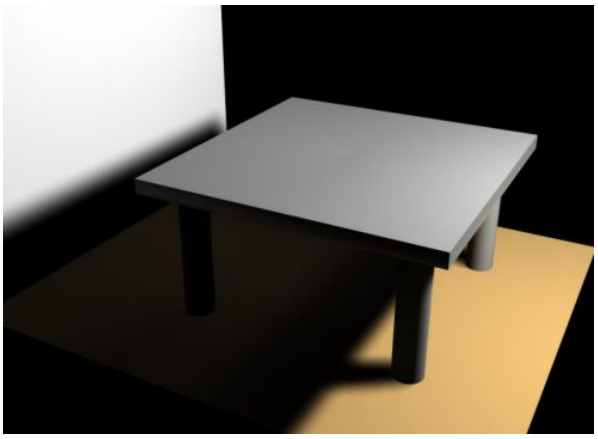


Selecting mapped shadows

Although shadows now appear smooth-edged, some quality problems can be detected easily. The shadow on the side wall has some banding, and the top part of the legs become illuminated through the table top! The default shadow map resolution is most likely not enough.

2. Make sure that the light source is select and go back to the property window. The Spec tab shows that the default Shadow map resolution is 100 pixels (actually 100*100 pixels, the resolution is always the same in both dimensions). Change the value to 400 pixels and render again. Now the quality is much better.

3. We finish this example by examining how to control smoothness of mapped shadows. Take a look at the property window of the light source again. The bottom part of the window contains a Size field, which by default has the value of 0.1 meters. Double the value to 0.2 meters and render the view to see how the shadows change. The bigger the light source, the smoother the shadows. It is usually necessary to keep the size in a sensible value which matches the scene dimensions. A distant light source hundreds of meters away from the target should have a diameter of several meters to function properly.


Mapped shadows have smooth edges

Room Illumination

The next example demonstrates the same illumination techniques as the chapter above: indirect illumination maps and mapped shadows. This time the scene is a bit more complicated, a room interior shown below. The picture shows how the example scene 'tutorprojects/rendering/room' looks like when rendered with two point light sources (lamp, sun) and some flashlight.


A room illuminated by two point light sources

First we generate the illumination maps:

1. Select the point light source inside the lamp object. Then multiselect the room level. We will generate the maps for only the largest surfaces of the scene, not for the furniture or other details.

2. Activate the UVImage tool from the Materials tooltab. Change Type to Indirect Lighting, set Samples value to 3-5 and hit OK.

3. Wait until the UVImage tool has finished the map computation. Then select the spot light source behind the room. We will change its shadow type to mapped shadows: open the property window, go to the Spec tab and activate the Mapped shadows option. Increase the resolution to 400 pixels.

4. Since the spot light is far away from the room, increase its size to 1 meter. Large sizes will also make lighting appear smooth: the spotlight will simulate indirect illumination through a window on a bright day rather than direct sunlight.

5. When using mapped shadows with a spotlight, it is best to use a smooth falloff for the spot edge (sharp edge with otherwise smooth shadows looks peculiar). Therefore, increase the Angle2 value to 5 degrees greater than angle1.

6. Now select the light source in the lamp object and activate mapped shadows for it as well. Resolution can be again 200-400 pixels. The default size of 10 centimeters is quite realistic and suitable, so you do not have to change it.

7. Now comes an important step: There are some glass objects in the scene. Mapped shadows can take only the closest (transparent, translucent or non-transparent) surface into account. Therefore, select the sphere which surrounds the lamp light source and make it Shadow Invisible using the Gen tab options of the property window. Do the same for the glass window object as well.

8. Render a new image to see the changes in the illumination. Adjust shadow map resolutions and light source sizes if necessary.



A smoothly illuminated room interior



Note

Shadow mapping can be applied to spot lights and point lights. Of these two, a spot light works much more efficiently and faster with shadow mapping, so try using that light source type whenever possible.



Note

Analytic rectangle is the most accurate and memory efficient target for indirect illumination mapping. A room wall modeled using a cube requires 4-6 times more memory than a single rectangle (for a comparable quality).

Post Processed Illumination

The table scene tutorial above explained how indirect lighting can be defined using illumination maps. There is another, quite different approach to compute indirect illumination: use of post processing methods.

Example project: tutorprojects/rendering/diffuselighting

Tutorial level: advanced

1. Load first the test scene 'tutorprojects/rendering/table'.

2. Go to the Channels tab of the Select window and create two new color channels called diffuse and unshaded. These channels will carry diffuse illumination and unshaded diffuse color information from the ray tracer to the post processing system.

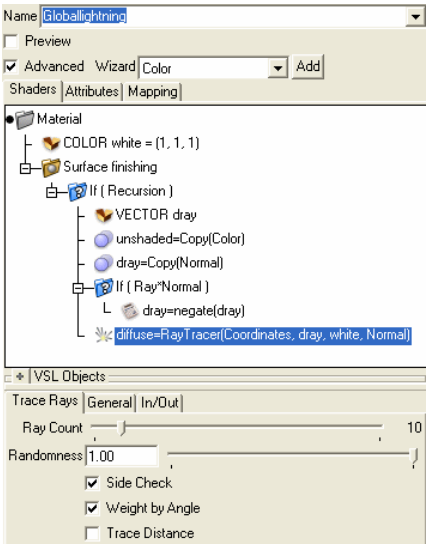


Note

These channels may already exist after loading another illumination project or the default startup file. If they exist, you do not have to create duplicate ones.

3. Switch to the material tab, create a new VSL material 'globallighting'. Check the Advanced option and define the following kind of VSL structure for it:

- The root level of the material contains first a color variable 'white' which is initialized to (1,1,1). The value controls the brightness of indirect illumination.
- The next object in the root level is a shader level of the type Surface finishing. This shader is suitable because it is evaluated only once at the end of surface shading.
- Surface finishing shader contains an If object which examines if Recursion channel is less than 1.01. In other words, diffuse illumination will be examined only at recursion depth 1 (for speed reasons).
- The If block defines one local vector variable called 'dray'. It will hold the surface normal which is turned to point towards the camera.
- Next a Copy object copies the diffuse color of the surface to the unshaded channel.
- Then a Copy object initializes the dray variable from the surface normal.

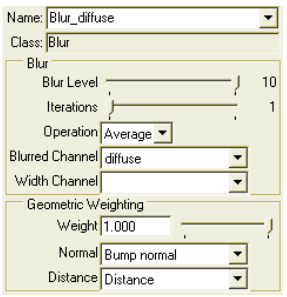


A material that collects illumination information for post processing

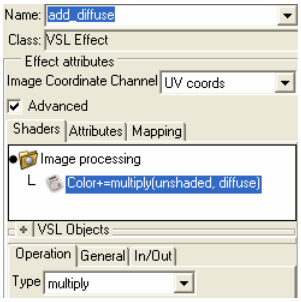
- Next an If shader checks if the normal points away from the camera by examining the dot product ray*normal.
- If the dot product is positive, dray is flipped using an Operation object with Negate. Now dray points to camera's side of the examined surface.
- The final step in the Surface finishing shader is a Raytrace object which traces 10 random rays around the surface normal at camera side of the surface; weighting signal is white because we do not want to premultiply illumination yet.

4. Go to the Geometric Objects tab, select the root level of the scene hierarchy, then switch back to the material library and drag&drop Globallighting material to the view window. The material will shade all objects of the scene.

5. Go to the Post Image Effects tab of the select window. Create a new Blur effect. On the property window, set blur level to a high value 5 - 10. The idea is to average the sampled indirect lighting over the image area. Thanks to this, random variations which would make the image dithered and grainy will disappear. Set Blurred Channel to diffuse (remember that it is the indirect lighting which was sampled by the Raytracer VSL object). Then set Geometric Weighting to level 1. This option controls the averaging so that blurring works only over continuous surface areas - illumination from the floor area that is far behind the table should not affect the table.



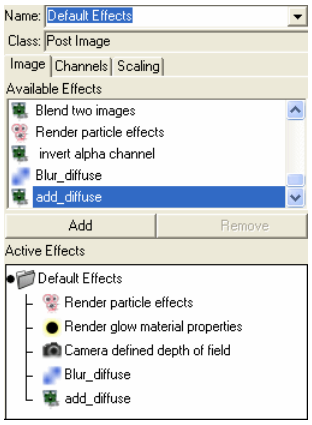
A blur effect averages the diffuse channel



A VSL effect adds indirect illumination to the image

6. Then create a new VSL Effect using the popup menu of the select window. Check the Advanced option. You see an Image processing shader already available. Add one Operation object to the VSL effect. The operation multiplies diffuse with unshaded and adds the result to the Color channel.

7. Select the Default Effects post image or other suitable post processing configuration and add the two new post effects to the end of the Active Effects list - blur first, vsl effect after it.

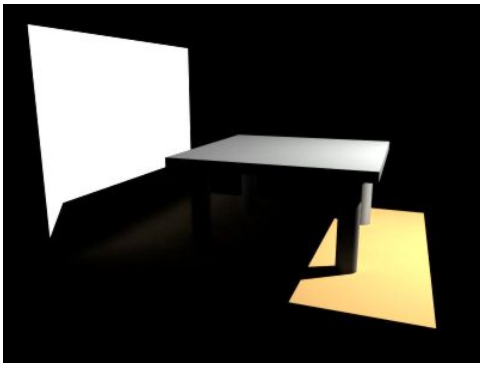


The post processing configuration finished.

8. Drag&drop the Default Effects to the view window. Select a render settings object, for example Reasonable quality, and set Undersampling values to 1. The new diffuse channel is not a trigger channel for undersampling, so it is best to sample every pixel accurately. Drag&drop the modified render settings to the view. Everything is now ready - just render the view to finish the tutorial.

Post processed illumination has some advantages compared to illumination mapping:

- Indirect illumination needs to be evaluated only for visible areas of objects
- Indirect illumination affects all objects of the scene
- There is no need to worry about well defined UV coords of SDS objects
- The technique can be applied to animated objects
- It is possible to ray trace the image once and then post process several times using the backup render features. The brightness of indirect illumination can be easily rescaled in the VSL effect.



Indirect illumination added by the post processing system

There are some disadvantages as well:

- Too low sampling can result to noisy illumination in animations
- Post illumination is computed at every render time, whereas illumination mapping can be computed only once

The globalillumination material above uses its own private recursion depth definition of 1. Indirect illumination is usually much weaker than direct illumination and therefore one recursion level is enough. Anyway, it is easy to apply a higher recursion value. The example project contains a second material version 'GI2' which does that. The second recursion step shading is otherwise similar to step one except that:

- Unshaded signal is tested against a minimal threshold. Sampling happens only if the threshold is exceeded. This both acts as an optimization and keeps indirect illumination shading separated from usual reflections etc.
- The Raytrace object uses much lower sampling value than 10 to keep the render time reasonable.
- This time we do the actual shading instead of only collecting channel information: the result of sampling is premultiplied with the diffuse color and added to illumination channel (the result of the shading will anyway end up to the diffuse channel at the recursion step 1).

Easy Global Illumination

The previous advanced tutorial demonstrated how to configure a global illumination system from scratch. In normal use, the standard components of the installation provide an easy work flow.

Tutorial level: medium

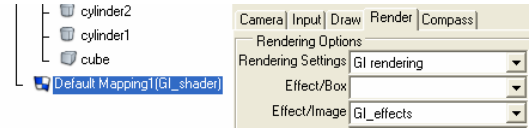
1. The first step is to model a scene as usual. No attention to the GI solution is required at this point. For example, load again the test scene 'tutorprojects/rendering/table'.

2. When the scene is ready (loaded), select the root level of the hierarchy. Then switch to the material tab and drag the material GI_shader onto the view window. This assigns the GI shader to all objects of the scene. If the GI_shader material, for some reason, is missing, you can use New/From Template popup menu to load it from the materials/illumination folder.

3. Next switch to the render settings tab of the select window. Drag and drop the object GI rendering into the view window. This render settings object includes an antialiasing channel setup which is tailored for global illumination. If GI rendering is not included in the settings library, select Paste from file from the popup menu and load it in from the rendersettings disk folder.

4. The next step is to select a suitable post processing configuration. The easiest way is to switch to the Post Image Effects tab and drag & drop the GI_effects post image into the view window. This assigns the effect as an image specific effect. As usual, if the GI_effects effect is missing from the current post effect library, just use Paste from a file popup menu and load in the posteffects/GI_effects.r3d file.

Now the GI setup is ready. Let's continue by adjusting the GI settings to an appropriate level. Since a big part of the GI is computed at post processing time, we can use some time saving tricks here:



The components for global illumination: a root level mapping to a GI shader, a render settings object and a post processing configuration

5. Make sure that Backup Ray Trace option is set in the View/Render popup menu. Then start ray tracing. The program first ray traces an image and finally post processes the global illumination into it.

6. To adjust the brightness of the global illumination into a suitable level, open the property window and select the VSL post effect GI_brightness_control from the select window. The effect has three top level controls. The gamma correction curve controls diffuse indirect illumination. Reflected GI color value scales the brightness of indirect illumination visible via reflections and transparency. GI Contrast adds some contrast to the global illumination. Change the values as you like. Then, instead of rendering everything again, just apply the Post process popup menu of the view window. Now the ray tracing part is skipped and the next image version appears much quicker.

7. The default settings include quite a low amount of indirect light sampling. Consequently, the rendered image may contain some noise which appears as an unwanted brightness variation. To reduce the noise, select the post effect called GI_blur. On the property window, increase the Iterations value by one or two units. Then apply Post process on the view window to see the improved result. If this does not remove the noise, increase the Iterations value further and post process again. A 10 pixel blur repeated 10 times will create 100 pixels wide blur.

There are other predefined GI components available as well. GI_draft post effect configuration works in a similar way, but it does not anti-aliase the result and is therefore about 4 times faster.

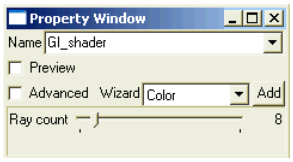
Important notes:

- A wide GI filter may be quite slow to render (several minutes). If the scene is simple and ray traces fast, a quicker way to improve quality is to increase the RayCount1 of the applied GI material. By default, the GI_shader material takes only a couple of illumination samples per surface point. Using for example Ray count value 10 improves quality significantly.
- GI blur filter does complex geometric calculations which require many channels, some of which are in the floating point form. Processing a video resolution image takes several tens of megabytes of RAM. This also means that

you should not apply a GI filter as a locally rendered image effect when using a slow internet connection to a render farm. Apply the GI effect per render box.

- If you have originally applied other custom post effects in your scene, remember to insert them also into the global illumination configuration you are using.
- GI filtering may partly remove ray trace time antialiasing. The only reliable solution is to antialiase after post processing i.e. use output down scaling. The GI_effects post effect scales the image by 50 % to ensure a proper quality. GI_draft does not scale the output and is therefore much quicker.

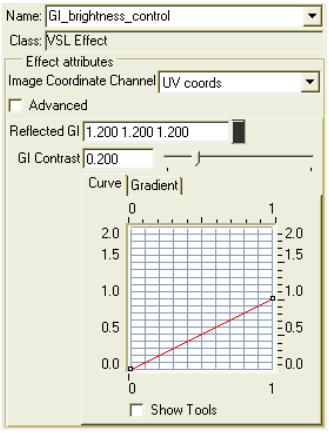
Global Illumination Components



The material which takes care of Global Illumination has only one control, Ray count. It defines how many sampling rays are triggered to examine the indirect illumination of a surface. When these GI sampling rays later hit new surfaces at deeper recursions, only a single ray is launched (for speed reasons).

Usually this is the most important quality setting for indirect illumination. The higher the sampling, the lower Iterations value can be used in the GI_blur filter. The result is finer detail and higher realism in global illumination.

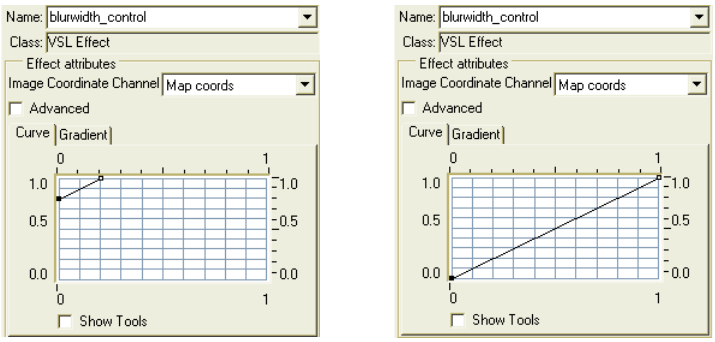
The tracing depth of indirect illumination obeys the general recursion depth of the applied render settings. If you need a dedicated limit for the GI tracing depth, drop the RayTrace VSL object of the GI_shader into an If VSL object, which tests surface:recursion channel value and activates sampling only at suitably low recursion steps. GI tracing depth can also be randomized by using a value computed by a Random VSL object as the input for the If block.



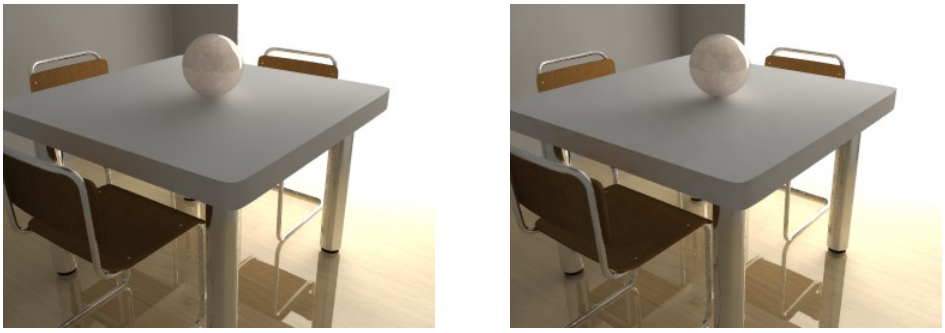
The post processing configuration for global illumination includes a component GI_brightness_control. As the name suggests, you can control the amount of indirect illumination using it.

- *Reflected GI* - Adjusts the amount of global illumination visible through reflections and transparent objects. Usually the value should be slighty larger (10% - 20 %) than the end value of the control curve for diffuse GI, because the finite recursion depth of ray tracing darkens reflected GI slightly more than diffuse GI. A higher value compensates the difference.
- *GI Contrast* - The contrast option scales GI by the viewing angle from camera. Under 'flat' lighting conditions, i.e. when a surface gets a similar amount of illumination from all directions, shadows become weak or disappear. GI Contrast adds a compensating shading factor, which helps to preserve fine surface details such as bump maps.
- *Control curve of Diffuse GI* - With this gamma correction curve, you can, for example, enhance illumination at dark corners but leave the bright areas of the scene unchanged. To brighten indirect illumination everywhere, move the middle points and the points at the right end upwards. The start point of the curve should usually stay at zero level.

The VSL effect called **blurwidth_control** provides some additional control over GI filtering. When GI ray count is low (e.g. less than 10 rays per pixel) or idirect illumination is unusally noisy, there is not much change to regulate blurring, and the control curve should be close to full level 1.0 everywhere. However, if a large number of GI rays is used (several tens of samples per pixel), the blur width control curve can be adjusted towards a diagonally ascending line, as shown in the example image below. Such a control curve reduces filtering in strongly shadowed areas. This improves the contrast of GI remarkably.



Blur width curves for low sampled GI (left) and for a high number of GI samples (right).



The effect of the blur width control curves to global illumination. The image on the right side was filtered using a diagonal blur width control line as explained above.

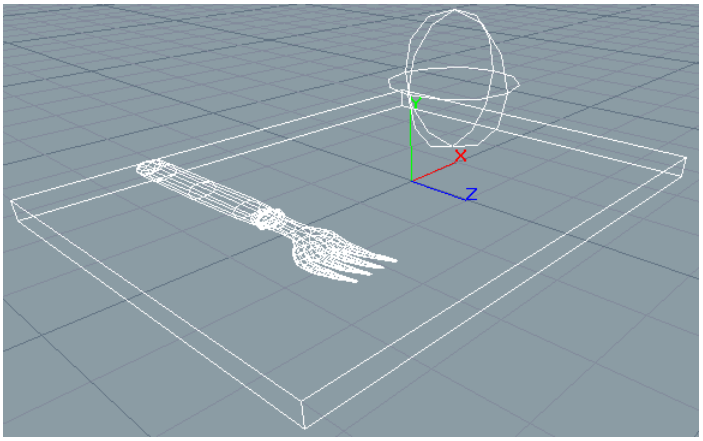
Outline Rendering

The next tutorial shows how to use outline rendering techniques of Realsoft 3D.

Tutorial level: Advanced

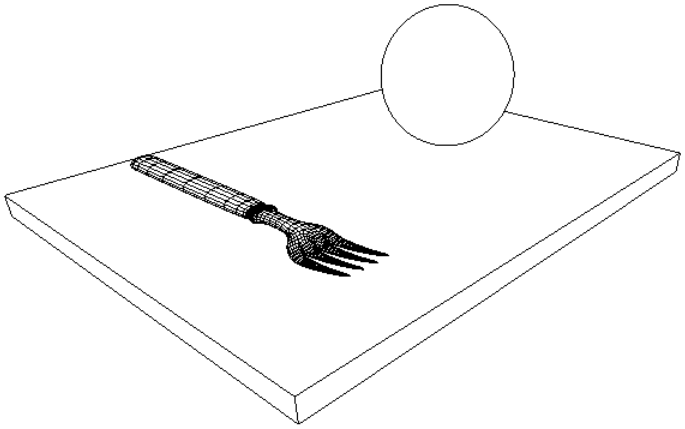
Example project:
tutorprojects/rendering/outlines

1. First we model a simple test scene: a flat analytic cube (table), a sphere and a freeform object modeled using the SDS tools (a fork). You may also load the test scene: 'tutorprojects/rendering/outlineobjects'.



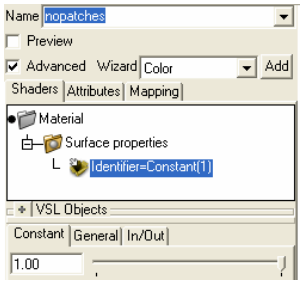
A test scene for outline rendering

2. Go to the Render settings tab of the Select window and drag&drop 'Black&white outline rendering' into the view window. Render the view - the result is a two-colored image.



Basic outline rendering

3. Let's remove the dense network of mesh boundaries from the fork object. Go to the materials tab of the select window and create a new, empty VSL material. Open the property window and rename the material as 'nopatches'. Check the Advanced option to see the VSL hierarchy. Also you may need to click on the '+' sign to access the VSL Objects. Drag & drop a Shader object into the root level of the material. Then drag & drop a Constant object into the Surface propertie shader. Set Output channel to Surface:Identifier using the popup menu of the VSL hierarchy tree. Change the assigned constant value to any nonzero value, for example 1.0. Identifier value 0 would blend the object with the background (empty space has identifier zero). If you later need to process several objects this way, just create several materials and assign value 1 in the first one, value 2 in the second one etc.

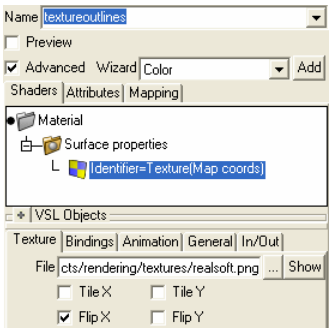


A material that hides surface patches in outline rendering

4. Make sure that the fork is selected and drag&drop the Nopatches material to the view window.

5. While we are in the material editor, let's create another material which texture maps some outlines. So, create a new VSL material called 'textureoutlines'. Check the Advanced option. Then apply the Texture map wizard. Select the Texture object from the VSL hierarchy and change the Output channel again to Surface:Identifier. Then pick a suitable texture file, such as 'textures/logo_bw'. Set the Flip X option. Close the property window. Then select the sphere from the view window and drag&drop the new material into the view.

Black texture areas will again blend with the background, and texture areas where the first subchannel (=red) value is 1.0 will blend with the fork material (its identifier was also one). It is easy to fix such conflicts without changing the actual texture map; just add a unique integer value constant to the identifier channel after reading the texture value to it:



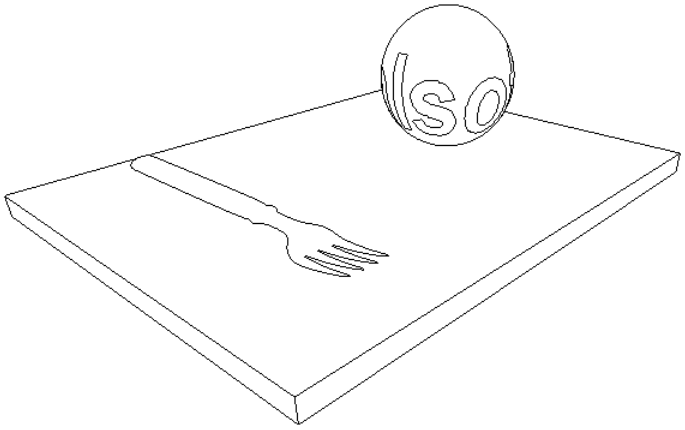
A material that texture maps outlines

```
identifier = texture(map coords)
identifier += constant(999)
```

Now render the view window:

At this point, rendered outlines appear jaggy without antialiasing. We can improve quality by using a suitable post processing configuration.

6. Switch to the Post Image Effects tab of the Select window. Select New/Line from the popup menu to create a line drawing effect.



Customized outlines

7. Select Default effects from the effects list and open the property window. Drag&drop the new Line effect from the Available Effects list to the end of Active effects hierarchy. Then go to the Scaling tab of the property window. Set Output Scale X= 0.5, Output Scale Y= 0.5. In other words, image resolution will be halved after post processing. This also has the nice side effect of making the image antialiased. Close the property window.

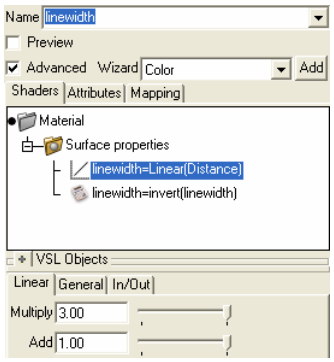
8. Drag&drop Default effects from the select window to the view, to make sure that it will be used in rendering. Switch to the Render settings tab of the select window. Open the property window of Black&white outline rendering. Go to the Post proc tab of the property window and make sure that the Safety area option is checked. Without safety area, Line effect will create gaps between render boxes. Then render the view. Rendering takes now longer but the quality is better. Outlines are now probably thicker, but you can adjust the line drawing width using the Min. Width attribute of the Line effect (Post Image Effects tab, property window of the Line effect).

9. As a final step, let's define varying line width. We need a channel and a material for that purpose. First go to the Channels tab of the Select window and add a new float channel called 'linewidth'.

10. Then go to the Materials tab and create a new VSL material. The contents of the material is shown in the example image.

The material simply defines that line width is dependent on the distance: i.e. width increases towards the camera.

11. Go to the object hierarchy tab and select the root level. Then switch back to the material library and drag&drop the linewidth material to the view. Since the material was applied to the root, all objects will obey the customized line width definition.

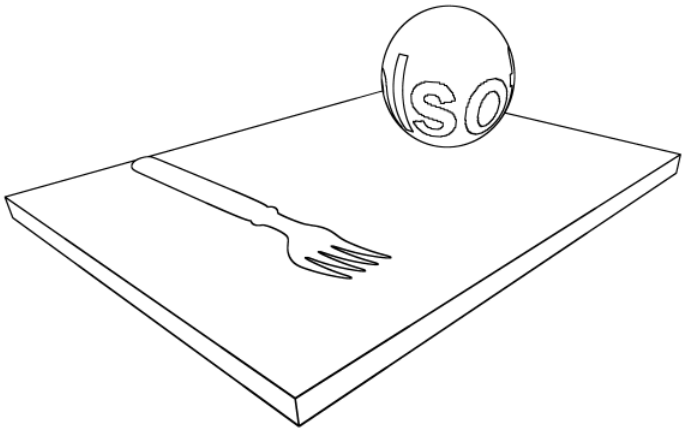


A material that defines line width

12. Switch to the Post Image Effects tab and select the line effect. On the property window, set Width Channel to the custom channel linewidth.

13. Select the Default Effects post image and decrease the output scaling factors to 0.25. This will improve antialiasing further so that the quality is good enough for final rendering. Drag&drop the Default Effects in the view window.

14. Output scaling to one quarter means that the program ray traces 16 times more pixels. To compensate the increased rendering work, select the render settings object Black&white outline rendering. On the property window, go to the Ray Tracing tab. Increase both Undersampling values to 8 pixels. Also, make sure that Interpolate is not set, it only slows the rendering without any result on the final output. Drag&Drop the render settings object Black&white outline rendering in the view window. Then render the view. If necessary, adjust Min. Width and Max. Width values of the Line effect which define the thickness range.



Antialiased outlines

Caustics

The caustics tools of Realsoft 3D compute how light reflects from shiny and transparent objects such as glass or shiny metal around the scene. Caustics can be computed into illumination maps or rendered as a part of the ray tracing process.

Adding caustics to your scene is usually easy. The first caustics tutorial about magnifying glass shows the basic steps.

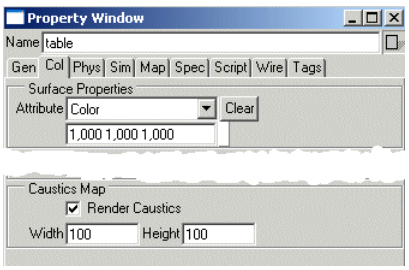
Tutorial level: Medium

Example project: tutorprojects/rendering/mag/glass

1. Model first the required test objects:

- An analytic rectangle is used as a table surface capturing the caustics
- An analytical cylinder forms the handle of the magnifying glass
- A Boolean and not operation between two analytical cylinders creates the rim around the lens
- The lens is an analytical sphere which is stretched suitably in one dimension
- A point light source is placed above the magnifying glass so that the glass will cast a shadow onto the table

2. Then map some materials to the objects: the standard glass material to the lens and a metal (steel, chrome etc.) to other parts of the magnifying glass. In the example scene, we have added a bump map to achieve a grooved handle. The bump map is basically two dense sine wave spirals, one traveling clockwise and the other counterclockwise around the handle. The bump map is not necessary for the caustics test, but it should be noted that the grooved shiny surface will create an especially interesting light scattering in the caustics lighting.



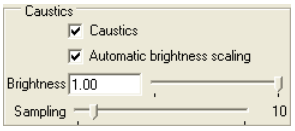
Object properties for caustics

3. Select the ground surface (table) and open the property window. Go to the Col tab. Activate the Render Caustics option and define a suitable caustics map resolution for the table surface - 100*100 pixels is quite OK for our simple scene.

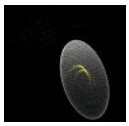
4. Before rendering, let's check the render settings. So, switch to the render settings tab of the select window and select a render settings object (for example reasonable quality). In the property window, go to the Ray Tracing tab

and make sure that recursion depth is at least 3, preferably 4 or 5. The higher the level, the more accurate and detailed caustics you will get. Recursion threshold should be also quite low, for example 5.

Then switch to the Miscs tab and make sure that the Caustics and Automatic brightness scaling options are set. Default maximal brightness level is 1.0 which corresponds the usual white color - it is quite appropriate. The default sampling level 10 is low but enough to get us started. Close the property window and drag&drop the edited render settings to the view window to make sure that the view will use it.



The caustics options of render settings


*A 100*100 pixel caustics map*

5. Start rendering. As the progress indicator messages show, the program first renders the caustics maps. Then the usual ray tracing starts. The caustics maps are placed into the 'temp' folder of Realsoft 3D home directory. If your machine is not a 3 gigahertz Pentium4 system, you may have time to switch to the image tab of the select window, create an image object and use the property window to browse and view the image file named as 'temp/causticsXXX'. The program cleans up the caustics maps automatically after rendering.

That's it! You can continue experimenting with the scene - for example, double the table's caustics resolution to 200*200 pixels and increase render settings Sampling value correspondingly. The result is that you will see crisper caustics patterns and finer details. A good rule of thumb is that usually 10-20 sampling units in the Miscs tab of the render settings is required for every 100 pixels of caustics maps: 300 pixel caustics resolution - sampling=30, 500 pixels - sampling=50 etc. If there are several caustics targets in the scene, the maximal resolution defines what is the suitable sampling rate. Too low sampling will result to noisy caustics.


Magnifying glass is not totally focused on the table



Note

Caustics illumination is attached as a texture map to target objects. A rectangle is a very efficient target because it has only one flat surface and map pixels are not wasted for the backside. Note also that a freeform object (NURBS, SDS etc) must have proper UV coordinates initialized to function properly as a caustics target.

Caustics Mapping

The previous tutorial explained how to use the automatic caustics rendering feature of Realsoft 3D. Now we examine how to add caustics manually to an object as an illumination map. The process is the same as when mapping indirect illumination: the UVImage tool computes the caustics map, creates required shaders and attaches them to target objects using material mapping.

This method has the advantage that the caustics illumination map (and other parts of the scene) can be edited after caustics rendering. This gives additional possibilities to control the rendering process.

Tutorial level: Advanced

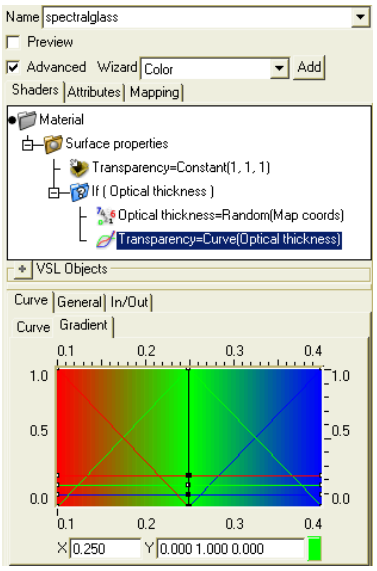
Example project: tutorprojects/rendering/spectralcaustics

1. Create a test scene: a flat analytic rectangle (floor) and a prism above the floor. Then add a point light source (a distant or spot light will also do the job) above the scene.

2. This time we will use a special kind of glass material: it processes multiple wavelengths in order to show the classical spectrum of white light which has travelled through the prism. Go to the material library, create a new VSL material called 'spectralglass' and open the property window. Define a material shown in the image below:

Let's examine the structure of the material in detail:

- The material is used only for refraction tracing. Therefore, the structure is maximally simplified to optimize rendering speed, and the root level contains only one Surface properties shader.
- In the surface properties shader, we first assign a full transparency 1,1,1.
- Next an If object examines if Exterior:Optical thickness is nonzero - in other words, if the examined ray is entering the glass material from an empty space. The wave length randomization is executed only when entering the glass, not in internal glass reflections. This both optimizes rendering speed and reduces sampling rate requirements, because each ray will use the same wave length during its traversal.
- If block starts with a random object that selects a random optical thickness value from a range that represents all sampled wavelengths. In our example, the range is quite wide, from 0.1 to 0.5.
- Each thickness value corresponds a wavelength which in turn can be seen as a certain visible color. Thickness values can be conveniently mapped to colors using a curve object.

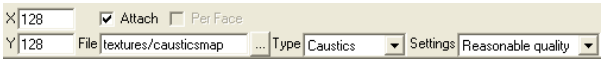


A glass material that traces multiple wavelengths

3. Map the spectralglass material to the prism using a parallel mapping (the Random object needs a proper Map coords parameter).

4. Go to the Render settings tab of the Select window and activate for example Reasonable quality. Open the property window. Go to the 'Ray Trace' tab and make sure that recursion depth is at least 3, preferably 4 or 5. Recursion Threshold should be quite low, for example 5. Then switch to the Misc tab. Activate Caustics option if not yet activated and turn on Automatic brightness scaling. Increase the default sampling level to 20 - tracing multiple wavelengths requires lots of samples even for this simple scene. You can then close the property window.

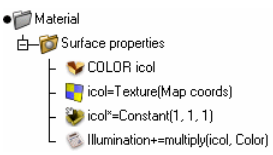
5. Select the floor rectangle from the view window. Then switch to the materials tooltab and activate the UVImage tool.



UVImage settings for caustics computation

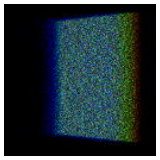
Set Type to Caustics, pick the proper render settings object from the Settings selector and enter a suitable name to the File gadget (for example 'textures/floorcaustics'). The default resolution of 128*128 pixels is quite OK. Hit Accept and wait until the tool finishes its job.

The tool adds a new material mapping object to the hierarchy and a new material to the material library. Check out the structure of the material using the property window. The structure is quite simple. The color from the illumination map is read into a variable, then multiplied with a constant value to scale the brightness and finally added to the total illumination.



A material that maps caustics to the floor

You can now select the texture object and hit the Show button to view the caustics map - it looks as in the adjacent example image. Quite promising! Before doing the first test rendering, let's modify the scene a bit.



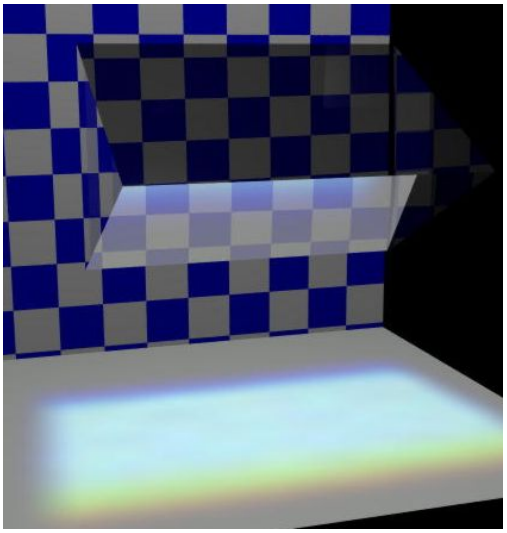
The caustics map created by a prism

6. The glass material we used was quite a special one, designed for caustics purposes. It would look quite bad in normal rendering. Therefore, go to the material tab and create a normal glass material if one is not yet included in the library. Then switch to the geometry tab of the select window, select the glass mapping from the hierarchy and change its target to the usual glass using the selector in the tool control bar.

7. You may also optimize the scene a bit now. Since the floor caustics already includes the shadow of the prism, you may apply one of the three optimization alternatives:

- Make the prism Shadow invisible on the property window
- The simple test scene has no other shadow casting objects as the prism and therefore you can also turn off Shadows completely from the render settings
- Add a Surface filtering shader to the normal glass material and assign transparency=constant(1,1,1)there

8. Render the view. If the spectrum looks noisy, go to the property window, select the Texture object of the caustics material and double its Blur value. You can also adjust the brightness at the same time if you like. The caustics map is precomputed and therefore new test renderings proceed quickly.



Light spectrum created by a prism

Animation with Global Illumination

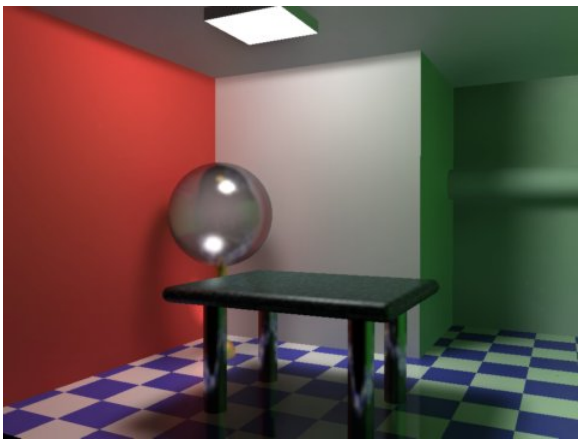
Computing the global illumination by random sampling light ray bounces is somewhat problematic for animations. Even when using a very high amount of light samples per pixel and heavy filtering, the computed illumination includes some random noise. Human eye easily notices even a low amount of noise. Just a couple of percents of noise from the full color intensity is still clearly visible. The constantly changing noise pattern usually appears quite unpleasant and distracting.

In this tutorial, we will examine how the noise problem can be solved using multi pass rendering. The big idea is to extend the illumination filtering over multiple frames. Also other random sampled shading components will be filtered in order to improve the quality.

To skip the details about how to configure multi pass rendering, go to the last section Multi Pass Rendering Summary. The summary tells you how to use the predefined components, which come with the installation.

Tutorial level: Advanced

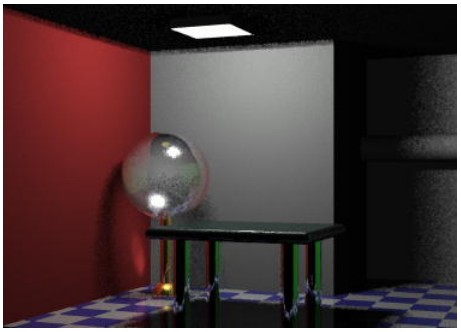
Example projects: tutorprojects/rendering/gi_room.r3d, gi_post1.r3d, gi_post2.r3d



An example shot from the final animation

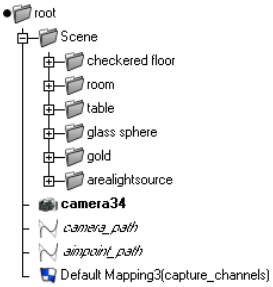
The First Pass: Ray Tracing Deep Files

1. Model a simple test scene or load the provided test scene 'tutorprojects/rendering/gi_room.r3d'. The test scene includes one area light source in the ceiling. Unless a high sampling rate and slow rendering is used, the shadows appear badly dithered. Nevertheless, you can set the area light's sampling rate to a low value, so that rendering time will be appropriate for animation use. If you model the test scene yourself, animate the scene for example by moving the camera along a path.



The test scene. The area light source is sampled with the minimal quality. Shadows are badly dithered.

2. We will apply post shading to the scene. Select the root level of the scene hierarchy. Go to the materials tab of the select window. select New/From Template from the popup menu and load in the file 'materials/illumination/capture_channels.r3d'. Then drag and drop the material to the view window to apply the material to the whole scene.

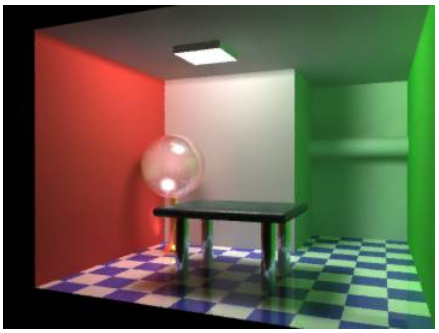


The hierarchy of the scene

3. At this point, switch to the post effects tab of the select window. First select Delete All from the popup menu to remove the effects we will not need. Then select the popup menu's Paste from a file and pick a file realsoft3d/posteffects/post_shading_effects.r3d. This loads two effect configurations, post_shading and post_shading_draft. These are the example systems for filtering shadows, reflections etc. - see the tutorial Post Processing/Post Shading for details. Drag and drop post_shading_draft from the select window to the view window.

4. Switch to the Render Settings tab of the select window. Drag and drop GI rendering to the view window. Make sure that View popup/Render/Backup Ray Trace option is enabled. Then select View/Render/Ray Trace. Adjust GI blurring and brightness parameters if necessary and apply View/Render/Post Process menu until the GI parameters seem to be suitable. You can also adjust other post shading components and shadow blurring until the shadows appear non-dithered. This is just a preliminary test stage - you can change all post processing parameters later. That is one of the main points of this tutorial.

So far, the workflow has been pretty similar to any project, which uses global illumination (or post effects in general). Next we prepare the first render pass. Instead of applying the post effects directly, we will collect all untouched raw data into very deep image files. This gives us a possibility to adjust the post effects later to produce variations from the same raw data. If the final animation shows some insufficient filtering quality, or if the customer suddenly wants a warmer lighting into the scene, this approach saves the most time consuming ray tracing phase.



Post shading applied to filter shadows and reflections

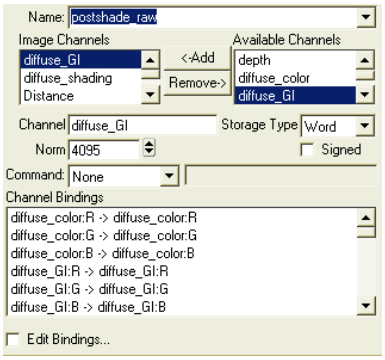
5. To define the deep image file contents, go to the output objects tab of the select window. Select New/Realsoft 3D from the popup menu. Make sure that the new format object is selected and open the property window. Use the name field to rename the format object for example as 'postshade_raw'.

We will have to add a large set of channels into this file format object. Select Distance from the Available Channels list and click Add. A new item labeled as Distance appears to the Image Channels box. Click the new image channel item and then change its Storage Type from Float to Integer. Very deep files consume huge amount of hard disk space. By optimizing the storage data type, we can reduce the file size. Word is usually enough to store distances

in a scene which has a diameter of some meters, Integer is sufficient for several kilometers and Float is enough for astronomical scales. After adding the distance, add other required channels - the full list is given below. Order of the channels does not matter.

- Distance - Integer
- distance2 - Integer. A second distance channel for reflected GI filtering.
- normal1 - Byte. The surface normal vector has unit length, therefore Byte is enough.
- normal2 - Byte. A second normal vector for reflected GI filtering.
- diffuse_color - Byte. The surface color value usually fits into a Byte.
- diffuse_shading - Word. Diffuse Illumination from light sources easily exceeds the Byte range.
- diffuse_GI - Word. Indirect diffuse illumination often has higher level than 1.0, therefore use Word.
- reflected_GI - Word. GI from reflections often needs higher bandwidth. Word is a good choice.
- postreflection - Byte. Reflected illumination. Byte is enough for this scene.
- posttransparency - Byte. Colors seen trough glass. We can use Byte here.
- self_illumination - Byte. Self illumination in this scene fits into Byte range.
- specular_shading - Byte. Specular highlights can exceed Byte range, but truncating to level 1.0 (white) does not harm in this project, because we will not utilize high intensity values for special effects.
- rn - Byte. Byte is always enough for the flashlight coefficient.

In the list above, the data types are mainly set to the minimal level in order to reduce the file size. If you have lots of free disk space available and feel uncertain about the right choice, just use one step better alternatives. Just remember that Float takes 8 times more space than Byte. The configuration below already consumes about 50 bytes per pixel. In the double PAL resolution, this means over 60 megabytes per image! A CD disk can only store about 8 of such images. For a 250 frame animation, you will need 20 GB of empty hard disk space. However, some of the channels are almost empty and contain only some local compact peaks, such as specular highlights. Therefore, the deep image files usually compress very well and a backup of the 20 BG raw data can most likely be stored to a DVD disk by using .zip compression.



Configuring the deep file format

The illumination components diffuse_GI, reflected_GI and diffuse shading are stored using Word data type. It may be possible to store them using Byte, if the lighting of the scene does not contain high intensity peaks. Insufficient data type increases sampling noise and therefore decreases rendering quality. If you use high intensity illumination effects, for example HDRI, bandwidth of the Byte data type is probably not enough.



Note

You can avoid the laborious file configuration step by saving predefined file format objects into a suitable folder. Realsoft 3D/imageformats/postshade_raw.r3d contains the required .r3i format object for this tutorial. Use Paste from a File popup menu to load it into the output objects library. You can check and optimize the data type for each channel if you like.



Note

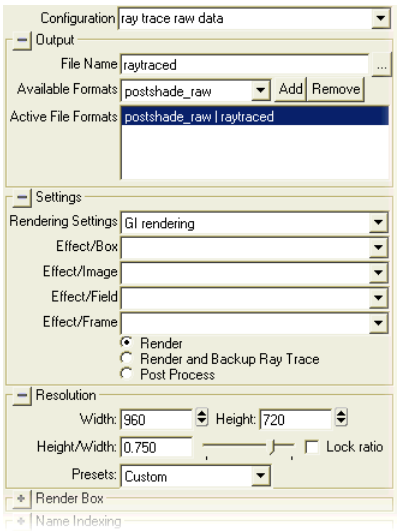
The file format defined above does not include all post shading channels. Illumination and no_fog are not stored, because the example scene does not include fog. The example version postshade_raw, which

can be found from the imageformats folder, is more extensive and includes all post shading channels. You can delete the mentioned two channels, as well as Glow size and Glow color channels to save some disk space. If you need these additional channels for other scenes, the recommended data types are:

- Illumination - Word. The total result of shading. If you need to spare disk space and don't need to trace illumination peaks accurately, use Byte.
- no_fog - Word. Illumination before a possible global fog. Use the same data type as for the Illumination channel.
- Glow color and Glow size - Byte.

When rendering your own projects, you may have to expand the channel collection even more, depending on the applied post effects.

6. When the deep file format is ready, open the file rendering window (Ctrl +r). Add the deep file format, which we configured above, to the Active File Formats list and remove other formats. Give a suitable name and path for the image files (for example 'raytraced' because these files will hold purely raytraced information without any post effects). Select GI rendering as the applied Render Settings object. Clear all post effect selections. Finally, set the resolution to the double of the intended final resolution. If you plan to create a 480*360 pixel animation, double it to 960*720 pixels. The extra resolution will be used for antialiasing purposes. Save the project as pass1.r3d and then start rendering.



Rendering setup for the first pass

The Second Pass: Post Shading and GI Blurring Single Frames

With the second rendering pass, we will filter the noisy area lighting and the global illumination. GI, which is the noisiest and most problematic shading component, will not be added to the shading yet. It will be stored for another, multi-frame filtering pass.

7. Go to the geometric objects tab of the select window. We do not need the geometry in the second rendering pass, so you can either delete all objects or make them invisible in ray trace rendering. However, it is a good idea to leave the camera object intact, because some post effects (DOF, lens flares) depend on the camera. Although we do not use such effects in this tutorial, this is a safe working habit to learn.

8. We can optimize the render settings a bit. Go to the render settings tab, duplicate the GI rendering object and rename the copy as 'QuickRT' or something similar. Using the property window's Ray Tracing tab, set Horizontal and Vertical Undersampling to 4*4 pixels and turn Interpolation off. Switch to the Distr tab and turn box rendering off. In the Post Proc tab, turn the Safety area option off (an additional continuity area around the image would not improve anything, because the rendering data comes from the deep files, which do not cover the extension region). This kind of render setting configuration initializes the empty black image in a minimal time. You can also load in the optimized settings from the 'realsoft3d/rendersettings/QuickRT' file.



Note

The render settings object above uses only one single rendering thread. Because of filtering continuity issues, use of several threads would not speed up rendering much. Later, when using frame blur effects,

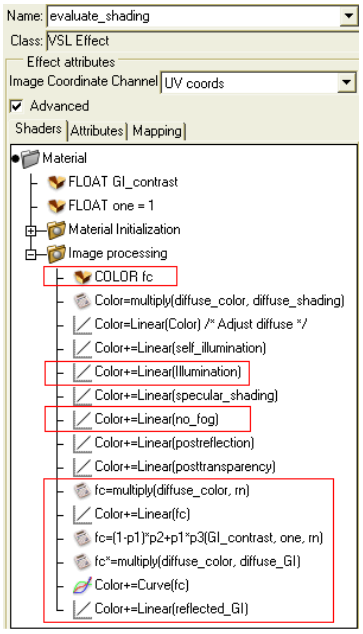
hard disk IO for several rendering threads reading simultaneously the same, very large data files might actually slow down the rendering.

9. Go to the Image Effects tab. We will now configure a post effect system, which filters the shading information from the deep ray traced files.

10. Because the filtering system will include the post shading components, one of the existing objects is a good starting point for the new system. So, select post_shading_draft and rename it as 'PostShadeFile'.

11. You can optimize the post effect system slightly to gain some rendering speed:

- Select 'remove_fog_and_custom_shading' from the Active Effects list and Apply Remove from the popup menu. There is no fog in this scene.
- Select the 'blur_fogginess' from the Active Effects list and remove it, too.
- You can remove 'Render glow material properties' and 'Camera defined depth of field' as well. There is no glow in the scene and DOF is not used (when applied, DOF should be the last effect of the final rendering pass).
- On the select window, select the VSL effect 'evaluate_shading'. On the property window, put the effect to Advanced mode and delete the following VSL objects from the shader tree:
  - Color fc - unused variable
  - Color+=Linear(Illumination) - custom shading not used
  - Color+=Linear(no_fog) - fog shading not used
  - fc=multiply(diffuse_color, rn) - no flashligt needed
  - Color+=Linear(fc) - no flashligt needed
  - fc=(1-p1)*p2+p1*p3(GI_contrast, one, rn) - GI will not be added in this pass
  - fc*=multiply(diffuse_color, diffuse_GI) - GI will not be added yet
  - Color+=Curve(fc) - GI will not be added yet
  - Color+=Linear(reflected_GI) - reflected GI will not be added yet



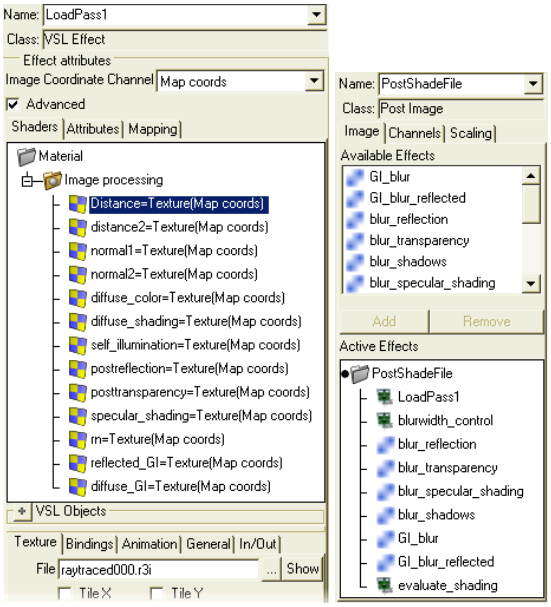
Remove inactive post shading components

12. One important effect is still missing: the one, which loads the channels to be processed from the deep files. Apply New\VSL Effect from the popup menu of the select window. Rename the new effect as 'LoadPass1'. Check the Advanced button of the property window. Activate the root node Image Processing and select New/Texture from the popup menu. Then use the browse button to select the first image of the deep image sequence. Switch to the Animation tab of the property window and change the Animation Type to Automatic. Open the popup menu of the VSL tree and change the Output parameter to first one of the required channels, Image:Distance.

13. Duplicate the first Texture object of the VSL tree using the popup menu. Change the output parameter to the next channel, Image:distance2. Repeat this for the remaining channels: normal1, normal2, diffuse_color, diffuse_shading, specular_shading, postreflection, post-transparency, self_illumination, diffuse_GI, reflected_GI and rn. Reading order does not matter.

14. Select again the effect PostShadeFile. Drag and drop the new VSL effect 'LoadPass1', which we just created, to the beginning of the Active effects list. The post effect setup for the second pass is now ready.

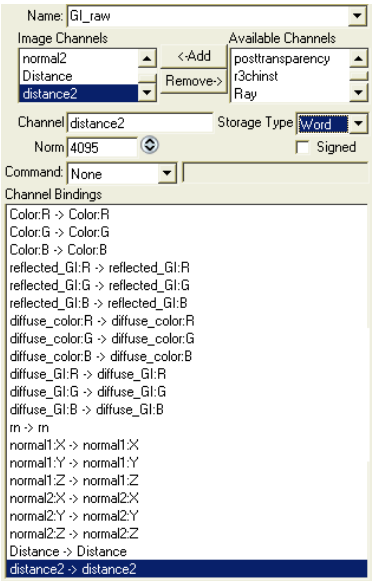
15. Go to the output objects tab of the select window. Select New/Realsoft 3D from the popup menu to add a new native format object (you can also load imageformats/GI_raw file to skip this step). Rename the new object as 'GI_Raw'. Then add the following channels to the object the same way as instructed in the step 5 above:



Define a VSL effect, which reads channels, and add it to the beginning of 'PostShadeFile' configuration

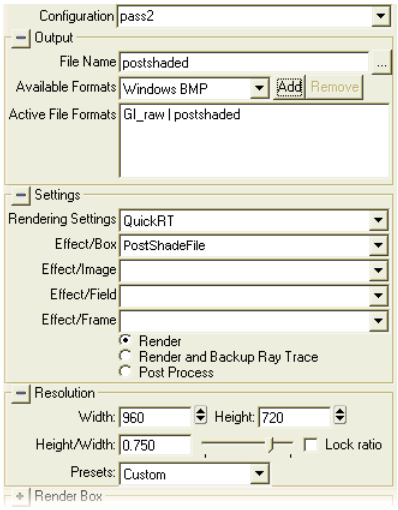
- Color - Byte
- Distance - Integer
- distance2 - Integer
- normal1 - Byte
- normal2 - Byte
- diffuse_color - Byte
- diffuse_GI - Word
- reflected_GI - Word
- rn - Byte

The new format no longer includes majority of the post shading channels, because they are already composited into the Color channel. The channels above store the GI information for the final filtering pass.



The file format which stores the channels for the final pass

16. Open the file rendering window. Change the file name and path so that images from the first pass won't get accidentally overwritten. Change the active file format to the new GI_raw object. Set the Render Settings selection to the QuickRT object, and Effect/Box to PostShadeFile. The resolution remains the same (=the final resolution doubled). Save the project as 'pass2' and start rendering the animation.



Rendering the second pass

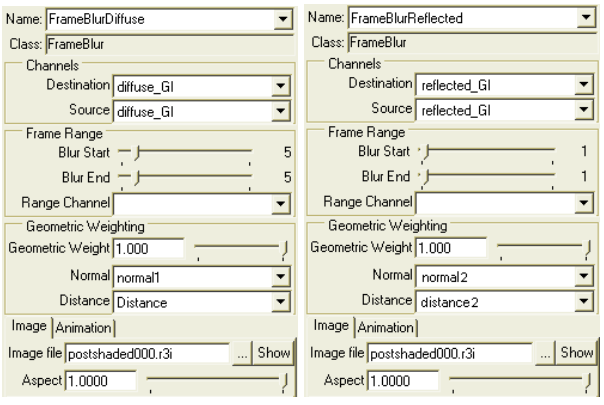
The Third Pass: Multi Frame Filtering

17. When the second pass rendering is finished, go to the post effects tab of the select window. We will now configure the last filtering system. An example system is included in the file 'realsoft3d/posteffects/MultiFrameGI.r3d' - load it in to skip the steps 18-21 of the tutorial.

18. Select New/FrameBlur from the popup menu. Rename the created effect as FrameBlurDiffuse, because we will blur diffuse global illumination with it. Use the browse gadget to select an image from the second render pass. Switch to the Animation tab of the property window and change Animation Type to Automatic. Go back to the Image tab and set Blur Start and Blur End to 5 frames. This means that the effect will combine GI information from 5 frames before the current frame to 5 frames after the current frame, 11 frames total. In each frame, we will get ten times more information about the global illumination; this will help to reduce the noise significantly.

Set Source and Destination Channels to diffuse_GI. This means that the effect will read diffuse channel from the disk files and store the blurred result to the diffuse channel of the post processed image. Then set Geometric Weighting to 1.0 - frame blur should combine GI information only from geometrically continuous areas.

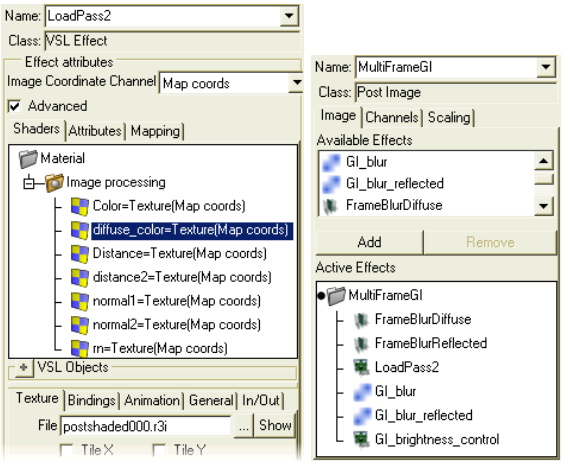
19. Duplicate the created FrameBlur effect. Rename the new effect as FrameBlurReflected. This time set the source and destination channels to reflected_GI. This part of the GI is usually much more coherent as the diffuse GI. Therefore, set Start and End to only 1 frames.



Frame blur effects for the diffuse and reflected GI

20. Create a new VSL effect and rename it as LoadPass2. Add 7 Texture objects, which read Color, diffuse_color, normal1, normal2, rn, Distance and distance2 from the pass 2 files. Remember to set the animation type of each Texture object to Automatic. Note that we do not have to read diffuse_GI or reflected_GI channels here - the two frame blur effects take care of that.

21. Duplicate the post effect PostShadeFile and rename the copy as 'MultiFrameGI'. Remove all effects above GI_blur from the MultiFrameGI configuration. Add the above created effects FrameBlurDiffuse, FrameBlurReflected and LoadPass2 to the beginning of it in this order. Leave GI_blur and GI_blur_reflected to the configuration. These effects will be used now again in order to eliminate illumination banding artifacts, which easily occur, when a moving camera reveals previously obscured areas of the scene (GI sampling within a frame is heavily randomized, whereas time stepping is not random at all but happend at regular intervals).



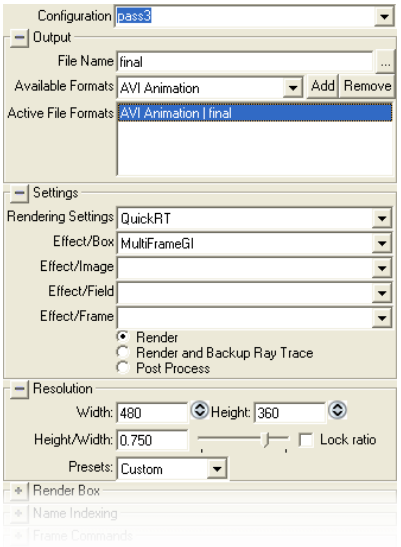
The VSL effect, which loads channels from the second pass images, and the post effect configuration for the final pass

So, we first blur GI over image area, then over time and finally again within each frame. However, you can and should reduce GI_blur values quite much for the second filtering round. For example, GI_blur Level can be 10 and Iterations 2. Similarly, GI_blur_reflected can be reduced to Level=5, Iterations=1.

Contine editing the contents of MultiFrameGI as follows: remove all effects after GI_blur_reflected. Then add GI_brightness_control to the end of the MultiFrameGI configuration. It will be the last effect. It adds the many times filtered global illumination to the images.

Select MultiFrameGI effect and go to the the scaling tab in the property window. Set Output Scale X and Y to 0.5. The last pass will shink resolution to one half, thus improving the anti-aliasing quality.

22. Open the file rendering window. Now set the Active Outputs to a suitable 'normal' image or animation format, such as .jpg or .avi. Define also a suitable file name. The render settings should be again QuickRT. Set Post effect/Box to the new MultiFrameGI. Drop the resolution to the actual final resolution (e.g. 480*360) from the doubled value. MultiFrameGI, which has Output scaling factors 0.5, halves the resolution from its source data to antialiase the images. Save the project as pass3 and start rendering.



The final rendering pass

Multi Pass Rendering Summary

A brief summary of the multi pass rendering steps using the predefined components is given below. You can naturally use your own components instead of the objects, which come with the standard installation.

1. Model the scene as usual

2. Default map the material 'realsoft3d/materials/illumination/capture_channels' to the root level.

3. Render using the deep file format 'realsoft3d/imageformats/postshade_raw' in doubled resolution and with no antialiasing.

4. Load in the post effect system 'realsoft3d/posteffects/PostShadeFile'. Select the included VSL effect 'LoadPass1' and using the property window, change the file name of every Texture object to the image sequence you rendered in the previous pass. Set the frame count in the Animation tab of each Texture VSL object to match the frame count of yor animation. Remove or disable the VSL object, which adds filtered GI to the image. Change Output Scale X and Y of the LoadPass1 configuration to 1.0.

5. Make all geometry objects except the camera RT invisible and render the second pass to 'realsoft3d/imageformats/GI_raw' format with an optimized render settings object 'realsoft3d/rendersettings/QuickRT' and 'PostShadeFile' as the post effect per render box.

6. Load in the post effect system 'realsoft3d/posteffects/MultiFrameGI'. Update the image file names of the two FrameBlur effects to match the image files from the second pass. Change the Frames value of the FrameBlur effects in the Animation tab. Then update the file names of the VSL Texture objects in the VSL effect 'LoadPass2', and set the Frames value in the Animation tab to a correct value.

7. Render the final pass to the desired format with the QuickRT settings and the MultiFrameGI post effects.

photon map Illumination

This tutorial explains how to use photon maps to add indirect illumination into an animated walk through in a static scene.

Tutorial level: Advanced



An example shot from the photon map illuminated scene

1. Load the well known Sponza Atrium reference scene from the Internet. We recommend using .3ds format; it imports all textures automatically.

2. Start Realsoft 3D and import the atrium scene. Use the triset import option; it is very suitable for the scene. Select all imported trisets and apply the Auto Smooth tool with, say, a 30 degree limit angle.

3. The imported scene should already include a point light, which could be used directly, but you can save some rendering time by creating a spot light to the same location, pointing down to the atrium and enclosing the whole building inside its cone. Then delete the original point light.



Note

If a scene includes lights in a close range (a lamp in ceiling etc), we recommend using the physically correct Distance Squared falloff type. In this scene, the light is relatively far away from the building, and the falloff type is not important.

4. We will next consider some important controls for the photon map computation. Photon mapping obeys the same rendering settings as the usual image rendering. Go to the Render Settings tab of the select window and pick a setting object, say 'Quality over Speed'. Open the property window and go to the 'Ray tracing' tab. Set recursion depth to a fairly high level, say 15 bounces or so. The value defines the maximal number of photon bounces before

the tracing is terminated. Recursion threshold should be quite high, for example 200 units (=0.20). Typically, a diffuse surface reflects less than 80 % of incoming photons. Note that surface's diffuse color defines the main reflectivity, but the recursion threshold just adds a handy maximum limit. Thanks to it, you do not have to edit all colors, materials and textures of the scene to achieve a realistic diffuse reflectivity.

5. Other ray tracing settings are not relevant for photon mapping, but the 'Misc' tab has some important options. So switch to the rightmost tab of the property window. It includes a dedicated area of photon mapping settings. Make sure that 'Auto photon map' is not set (we will use a static photon map instead of dynamic map generated at each rendering cycle). Set first a suitable value to the KPhotons field, say, 1000. This will trace hit positions of approximately one million photons. Sampling rate can be 100 and Noise Reduction can be set to 1.

6. Close the property window and go to the 'Light Sources' tool tab. Activate the Photons tool. Make sure that the edited render settings (Quality over speed in our example case) is selected in the Settings gadget. Include Direct Photons option should be left cleared; it is faster to compute direct illumination using light ray tracing. Then click Accept.

7. Wait until the tool finishes the computation. A new object appears in the hierarchy; it is a photon map, a special kind of light source. Now you can edit its brightness, sampling rate and other settings in the tool control bar, and render test images to see how it affects the scene. You will probably have to increase the Brightness value of the photon map considerably, because only small part of photons will reach the dark corners of the atrium. If the image seems to include only direct illumination, set a dramatically higher Brightness value for the map, say, 5000. After a couple of tests, you should be able to find a proper Brightness level.

8. You can see the actual stored photon-surface collision count from the toolbar when the photon map is selected. The atrium scene is not very complex, but detailed illumination cannot be achieved without several millions of photon hits. Unfortunately high density photon maps require plenty of RAM memory, so one has to be careful when setting the KPhoton value. A rough rule of thumb could be that 1 GB of RAM is enough for about 5 millions of photons (considering that there is also other kind of geometry in the scene, requiring its own part of RAM resources).

9. After the test round, we will create the actual high quality photon map. Delete the test photon map. Open the property window and select again the render settings. You may increase the KPhotons value to match your computer resources and scene requirements as instructed above (e.g. to 10000 KPhotons).

This time we will also use the Noise Reduction option. As the name indicates, it will improve the quality of the photon map, so that sharper, less noisy illumination can be rendered using the map. Unfortunately the noise reduction process is very computation intensive. To estimate roughly what kind of value you can use, multiply the computation time of the first photon map with the noise reduction value and the KPhotons increment factor. If the test map was created in 5 seconds, a new map with 10 times higher KPhotons value and Noise Reduction value 100 will take 5*10*100 seconds = 5000 s = about one and half hours. It is quite a long time, but fortunately the computation will be done only once. All animation frames will use the same map and render very quickly.

10. Create a new photon map with the new high quality settings. When the map is ready, save the scene to ensure that the valuable quality photon map will not get lost accidentally. The saved file will also act as a backup data for the step 12 of this tutorial, which will change the map permanently.

11. Test render images with the new map. The higher Noise Reduction you used, the lower Sampling (=render time photon averaging) rate is required. If the Noise Reduction value was 10, Sampling may have to be 1000 to smooth out noisy illumination, whereas Noise Reduction 100 might work well with Sampling rate of 200.

12. When a suitable Sampling rate is found, it is time to bake the filtered indirect illumination into the map itself. Select the map and click the Filter tool button in the tool bar. The tool computes the indirect illumination at photon positions and stores the illumination into the photons. For the sake of memory efficiency, photon's original illumination is not backed up in RAM, but the new filtered illumination overwrites it. When the filtering is ready, Sampling rate decreases automatically to a very low value (default = 6). Now the photon map renders very quickly. Save the scene with the finalized photon map.

13. Create a large sky dome sphere around the scene. Assign a light blue illumination and black diffuse color to it. Make sure the spotlight is inside the sky dome. Insertion of the sky dome was deliberately left this late, because we

did not want it to interfere with the photon map computation. It is not useful to store photons on the evenly lit sky surface - it would simply be waste of RAM memory.

14. Add a camera object inside the atrium and animate it to move around. Open the File rendering window. Configure suitable settings and enable the Walkthrough rendering mode. Render the animation to a suitable format.



Chapter 8. Post Effects

Post Image Effects

The Realsoft 3D rendering engine includes an open and powerful post processing system. Effects such as glow, lens flares and dust can be added to the images using the included post effect modules. The powerful features of the post processing engine include:

- **Post processing is freely configurable**. The user can define any combination of effects in any order. The same effect can be used multiple times with different parameters within the same rendering pass.
- **Expandable data channel configuration**. The user can define and use an unlimited amount of data channels to achieve desired effects. All channels used for defining material and object properties are automatically available for use in post processing. For example, the user can define a custom glow casting light source. The glow effect can add glow to those parts of objects, which are illuminated by the special glow light (although this light may be otherwise invisible). All channels can be output to any suitable configuration of file formats: Color+Alpha ->Targa RGBA, Distance+Transparency+Reflection->.bmp, etc.
- **User definable data accuracy**. The post processing system support all data bandwidths from the traditional byte/channel to the full double precision float/channel accuracy, which is certainly enough for any purpose.

Most post effects are quick and memory efficient to render. Note that they cannot fully interact with ray tracing, because they are rendered after the raytracing pass.

Post processing is an optional step in the rendering pipeline. If post processing is not activated, the rendering engine renders the images directly to the actual output device, such as a file (BMP, TGA, AVI, etc.) or to a view window. When post processing is activated, the image data is copied to a *Post Image*. There can be any number of *Post Effects* associated with the post image; each will add its contribution to the result. When all the effects have been applied to the post image, its contents are copied to the actual output device(s).

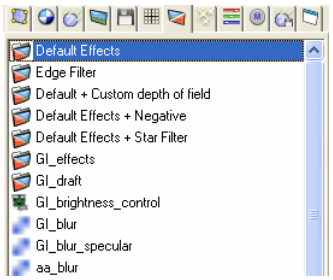
Managing post effects

Post effects can be divided into two groups: effects which are used for rendering particles ('Particle effects') and effects which are applied to the entire image ('Image effects').

Image effects can be created and managed through the select window's Post Image Effects tab.

This tab includes two kinds of objects:

- Post Images
- Effects, which can be applied to post images



Select window's Post Image Effects tab

Post images provide the necessary image buffers for post processing.



A post Image

Post effects are the actual workhorses. Any number of them can be attached to a single post image using the property window.

The most common post effects are rendered automatically. Namely, the default user interface is configured so that the file rendering window and the view windows use a post processing configuration defined by the post image called Default Effects. This configuration renders all particle effects, glow and depth of field. You can select another configuration from the file rendering window or from the view property window.

This is the basic theory you need to know in order to use post processing in Realsoft 3D. Let's go through one example demonstrating how to add post effects to your images in practice.

Rendering post effects to a view window

We will experiment using a simple post effect called Edge Filter. It examines sudden color changes in the image to detect edges of objects, and changes a shaded image to an outline drawing.

Tutorial level: Beginner

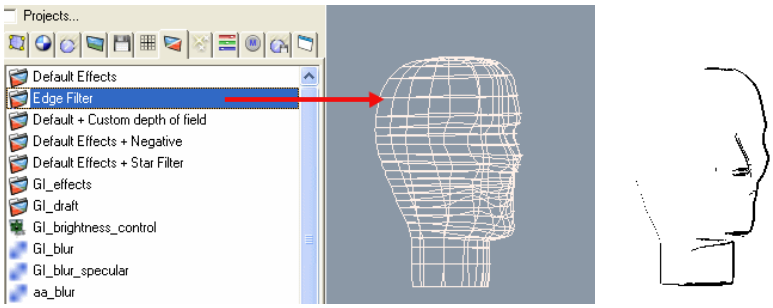
1. Model a scene, or load a provided example 'models/nurbs/simple head'. Render the view; you get the usual shaded result.

2. Select the Post Image Effects tab from the Select window. Drag and drop the Edge Filter post image to the view window.



Note

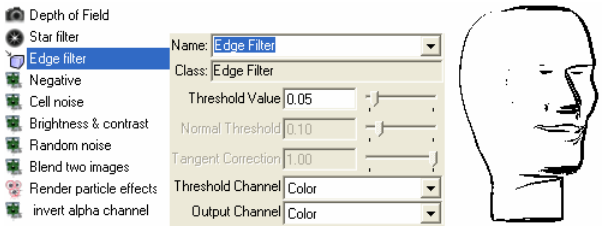
Drop the post image object. There is another 'Edge Filter' object appearing lower in the list, but it is a sub object of the post image. Remember that you always need one post image to set up post processing.



Selecting post effects for a view using drag & drop. The result rendered on the right.

3. Render the view. The edge filter changes the shaded image to an outline image.

4. The amount of detected edges depends on illumination, objects and effect settings. We can try to extract more edges. Double click the actual Edge filter effect (the lower one) on the select window to open the property window. Reduce the Threshold value to, say, 0.05, and render again.

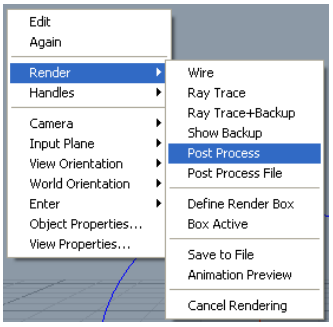


Adjusting the edge threshold value

Every time you re-render the scene, the rendering engine also ray traces the shaded image. If the scene is complex, this can make experimenting with post effects slow, because ray tracing usually takes much more time than the actual post processing. Fortunately there is a faster way:

5. We will continue experimenting with the edge filter: adjust a new Threshold value from the property window. Now, select Ray trace+Backup from view window's popup to render the image.

6. Now you have a new test image on the view and the rendering backed up. Adjust the threshold value once more. Then, instead of selecting full render, select Render/Post Process from the view popup. Only the post effects are rendered this time, and the result appears much quicker.



Rendering post effects only

Configuring post effects

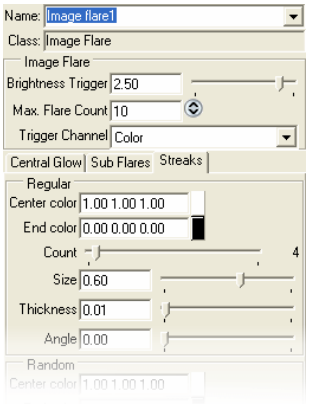
In this example, we will set up a post effect configuration, which applies a star filter effect. The brightest areas of images will get shiny star-like highlights. Such an effect is already available in the default configuration and you can activate it the same way as Edge Filter in the previous tutorial. Anyway, we will start from scratch to get familiar with the post processing tools.

Tutorial level: Beginner

1. Go to the Post Image Effects tab in the Select window and create an Image Flare object (right click and select Image Flare from the popup as shown in the image).

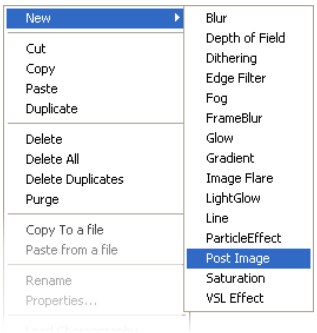


Create an Image Flare object



Specify desired image flare options

2. Double click the created Image Flare object in the select window. This opens the property window allowing you to define image flare specific options. Set the Brightness Trigger value to 2.5. The trigger value is the sum red+green+blue. For example, brightness of pure white 1+1+1 = 3 exceeds the trigger value. e.g. Any value that is within the trigger threshold receives an image flare. A value of 3 limits the image flare to only pure white pixels. 2.5 allows any reasonably bright pixel to fall within the threshold.

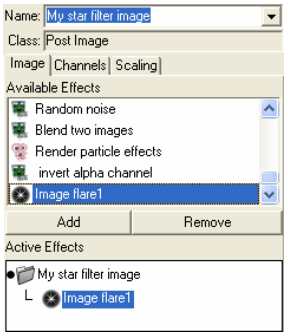


Create a new post image object

3. Leave the property window open and create a Post Image from the select window's popup menu.

4. Using the property window, rename the created post image to say 'My star filter image'.

5. The property window shows you two lists. The upper one includes you all effects available (including the Image Flare object you created above). The lower Active Effects list shows you the effects applied to the post image. Drag & drop the Image Flare object from the upper list into the lower list.



'My star filter image' contains only one effect, 'Image flare1'



Note

You can remove inserted effects by selecting Remove from the pop-up menu of the Active Effects list

6. Switch the select window to show the Rendering Settings tab. Select the settings, which the view window is using (or drag & drop 'Reasonable quality' to the view). Double click the Reasonable quality setting to bring up the properties window. Switch to the Distrib tab and disable the Box rendering option. Image flare has an exceptional property: it works better when box rendering is disabled. The reason is that Image flare analyzes continuous bright areas and adds the flare to the brightest position of each area. Render boxes may break these continuous bright areas.

7. Close the property window. Model a couple of objects and add a light source or two to the scene, so that the Brightness Trigger limit will be exceeded.

8. We have not yet assigned the new post processing configuration to any rendering engine. In the previous example, we did it by dropping the post image to a view window. This time we will use the view property window. Open it from the View Properties.. popup menu of the view window. Go to the Render tab and select 'My star image filter' from the Effect/Box gadget in the Rendering Options frame. Render to see the result.

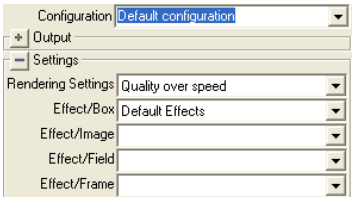


Image flares

Post processing configuration for file rendering

To activate a post processing configuration for file rendering, select a desired post image from the Effect/Box list. The post image will then be applied to each rendered image box. You can also define a post processing configuration that is applied to complete images , animation fields and animation frames.

If you don't want post effects to be rendered, clear these fields.



A post image 'Default effects' selected



Note

All selected effects are applied regardless of animation settings. For example, Effect/Field is processed even if field rendering is switched off.

Post Particle Effects

The previous post processing tutorials introduced some post image effects, which examine and modify the image rendered by the ray tracer.

The program also includes another category of post effects: so called post particle effects. They also draw their effect to a post image during post processing. The characteristic property of post particle effects is that they use geometric objects (particles) to define the position of effects in 3d space. The following comparison illuminates the difference between the two effect categories:

- 'Image flare' *post image* effect draws flares to the brightest spots of the image
- 'Lens flare' *post particle* effect draws flares to the positions defined by the particles

Mapping effects to particles

You can define the appearance of post particles using particle effects just like you define the appearance of ordinary geometric objects using materials: by mapping effects to particles. The idea is that particle objects define geometric properties such as position, size and orientation. The mapped effect uses the geometry information when rendering a shape representing the particle. The following example, a flock of lens flares, demonstrates this.

Tutorial level: Beginner

1. Create a bunch of 1D particles using the particle tool. Activate the tool from the Creation tab of the toolbar and drag the mouse around the view (left mouse button held down). Click the Accept button to finish the particle tool.



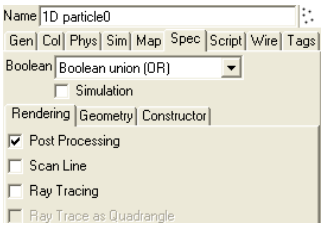
Particle tool activated

2. Render the view. You can see the particles as small dots. However, there is no post processing going on yet - what you see is *scanline* particle rendering.

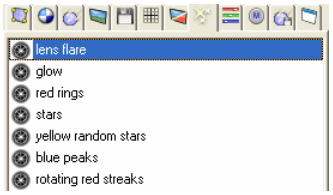


Particle object selected

3. Make sure that the particle object is selected. Open the property window from the Object Properties.. entry of the view popup menu, go to the Spec tab and clear the Scan Line option. Now the particles will be rendered using post processing system only. As a matter of fact, some post effects are sensitive to object distances, and leaving scanline drawing enabled may disturb post particle drawing later (post particles become hidden by scanline particles). Close the property window.



Scanline rendering disabled

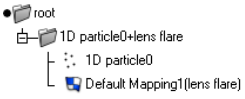


Particle Effects

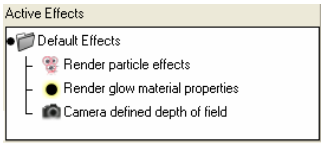
4. Go to the Post Particle Effects tab of the select window and drag&drop the desired particle effect into the view window. This maps the effect to the selected object (=particles).

5. You have now created a flock of 'lens flare' particles. Go to the object hierarchy tab in the select window, and open the sub hierarchy of the new object level '1D particle+default mapping(Lensflare)'. The hierarchy should look something like in the example image:

Note how object hierarchy is used exactly the same way as in material mapping (a wooden candlestick consists of the candlestick and wood mapping).



'Lens flare particles' object consists of two sub objects: 1D particles and a lens flare effect



The effect configuration contains the module for rendering post particles

6. Everything should now be ready for rendering the flares. However, let us check one important setting before rendering. Go to the Post Image Effects tab of the select window. To make sure that we are using the right post processing configuration, drag & drop the Default Effects item to the view window. It is used by the standard startup project, but you may have changed the configuration when experimenting with previous post image effect tutorials. Then double click Default Effects to open the property window. The window shows the contents of the selected post image.

There is one particularly interesting item included in the list of active effects: Render particle effects. It must be included or otherwise post particles will not be rendered. By changing the position of this object in the list, you can define when the post particle effect rendering takes place. The configuration is suitable, so close the property window without changes.

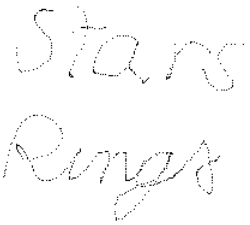
7. Click the ray trace button in the view control bar (or select the view's popup menu Render/Ray trace).



A flock of lens flares

Using multiple effects

Let's assume you want to create two particle flocks: one whose particles are rendered with a 'star' effect and another one, which is rendered with a 'red rings' effect. To do this:

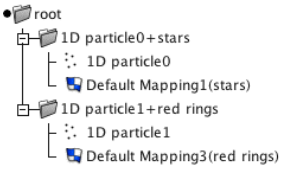


1. Activate the particle tool and create a number of 1D particles. When done, select Accept from the view's popup menu.

2. Activate the particle tool again and created the second particle object. The two particle objects might look as in the example image.

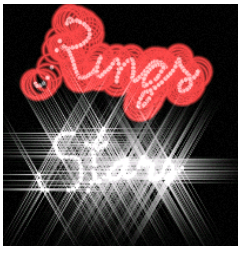
3. Select the first 'stars' particle object and drag and drop the 'star' effect from the select window's Post Particle Effects tab into the view window.

4. Select the second 'ring' particle object and drag & drop the 'red rings' effect into the view window. The current object hierarchy is shown on the right.



Two particle objects with different effects

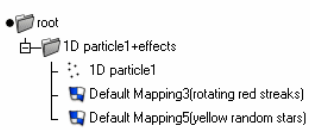
5. Start rendering.

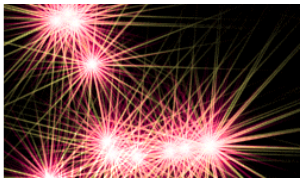


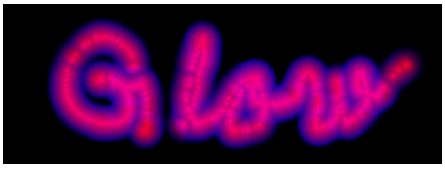


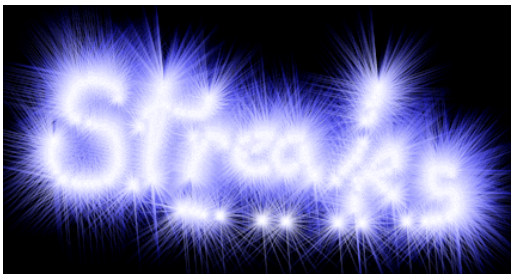
Note

You can combine effects by mapping multiple effects to a single object. The order of mapping objects in the hierarchy defines the order the effect are rendered.









Creating new particle effects

New particle effects can be created from the popup menu in the Select window's Post Particle Effects tab. The New sub-menu shows you the installed post effects. Just select an entry to create a new effect and use the property window to define its attributes as usual.



Note that one effect, such as the Lens Flare, can render a large number of different kinds of effects, from simple stars and rings to complex lens flares.

Blur effect

By default, Realsoft 3D's render engine computes razor sharp images. Sometimes it is desirable to produce 'smoother' images - to obtain a certain image style or perhaps to reduce antialiasing problems. Smoothing is a quick way to improve quality when only a low level of actual ray tracing antialiasing can be used (for speed reasons etc.).

The Blur effect is a useful tool in this situation. The fact that blur can be applied selectively greatly enhances the usability of the effect.

Tutorial level: Medium

Example project: 'tutorprojects/postprocessing/selectiveblur'

1. The first step is to model a simple test scene. For example, create a large floor plane using the rectangle tool and place some test objects above it. Texture map the floor and take a suitable view to the scene so that texture pattern gets dense in the horizon. Then ray trace the view.



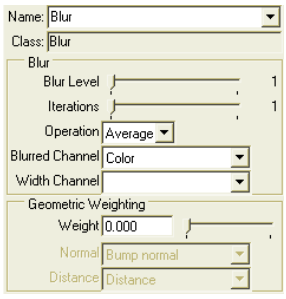
A simple test scene for the blur effect



Blur with default settings added to the image

2. Go to the Post Image Effects tab of the Select window. Select New/Blur from the popup menu. Then drag and drop the new effect that appeared to the end of the effects list to the view window. This inserts the effect to the current effect configuration. Render to see how the blur effect changes the image.

3. Select the blur effect and open the property window. By default, the blur effect used the minimal amount of blur (level 1, iterations 1). The blur was applied uniformly to all parts of the image.

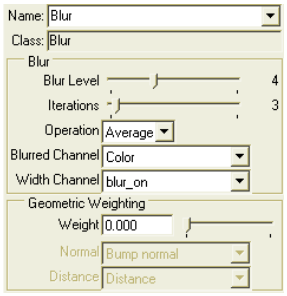


The default blur settings

Let's experiment with selective blurring. For example, we may try applying the blur only to those parts of the image, which are covered by the floor pattern. Other (foreground) objects remain sharp.

4. Switch to the User Definded Custom Channels tab of the select window. Select New/Float from the popup menu. Rename the created channel object as 'blur_on'.

5. Go to the Geometric Objects tab of the select window and select the floor object. On the property window, select the second Col tab. Then select blur_on channel from the Surface Properties/Attribute list. Define the value 1.0 to the attribute value field.



Settings for selective blur

6. Now switch back to the Post Image Effects tab of the Select window. Make sure that the blur effect is selected, and select blur_on to the Width Channel gadget of the property window. You may also increase the Blur Level and Iterations values a bit.

7. Render the view again. Now the floor is strongly blurred, whereas the foreground objects remain fully sharp.



Selective blurring

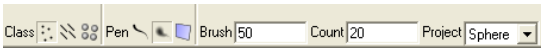
Particle Smoke

A large amount of post processed particles can be used to simulate smoke. The following tutorial shows how to do this.

Tutorial level: Medium

Example project: 'tutorprojects/postprocessing/smoke'

1. The first step is to model a cloud of particles. Activate the Particle tool from the Creation toolbar and select the airbrush options like shown below. Drag around the view window to draw an irregular cloud of particles. Select Accept to finish the tool.



Options for drawing a 1D particle cloud



The smoke scene

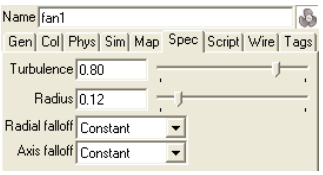
2. Go to the Simulation toolbar and activate the Fan tool. In the view window click below the particle cloud and click again slightly above to define a vertical axis for the fan (you might want to hold down the shift key to enable movement locking). Define an appropriate width for the fan.

3. To achieve a more interesting animation, place an object above the particle cloud. So go back to the Creation toolbar and activate Analytic Sphere from the Analytic tab. Create a sphere above the particles.

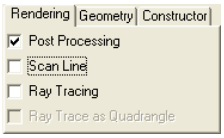
4. You may also add a light source to illuminate the scene (use Light Sources/Point from the Light Sources toolbar).

The scene should now be similar to the adjacent illustration.

5. Next you have to adjust some object properties. Go to the select window and activate the fan object. Open the property window from the popup menu. Go to the Spec tab and set Turbulence to 0.8. The other default settings are OK for our example.



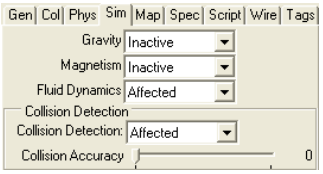
Defining fan turbulence to 0.8



Turning off Scan Line

6. Select the 1D Particle object from the select window. The property window shows three sub tabs in particle's Spec options. Clear Scan Line from Rendering options. You may also check the Geometry tab. It shows the amount of particles in the Count field. You will need several thousand particles. You can easily increase the amount by typing the desired value to the Count field.

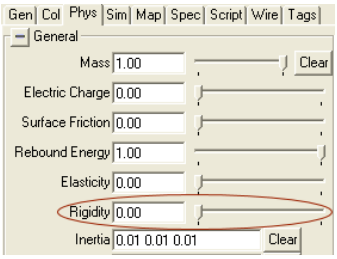
7. Switch to the Sim tab of the property window (the particles should still be selected). Set Collision Detection to Affected, because we want the particles to travel around the sphere, not through it. Set Fluid Dynamics to Affected, so that the airflow from the fan will move the smoke.



The simulation options of particles

8. Switch to the Phys tab. Change the Rigidity slider from 1 to zero. This allows individual particles to move freely instead of being a rigid object.

9. Switch to the Col tab. The Surface Properties frame includes a property list and a value gadget for the selected property. First select a suitable color for the smoke, for example a light gray. Then select Fade form the Attribute list and define a high fade value between 0.90 - 0.99. The more particles you created, the more transparent one single particle can be. Do not set fade to 1.0, because then you would not see any particles at all.



Zero rigidity allows particles to move freely in airflow



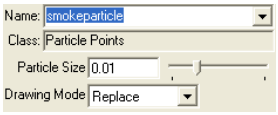
Defining the color and fade of particles

10. The particles are now ready. Next select the sphere. Go to the Phys tab of the property window and increase the Mass value to 1000 kilograms. The smoke should not move the sphere which would happen if the sphere is very lightweight. Alternatively, we could also make the smoke very light.

11. Go to the Sim tab and set Collision Detection to Cause to enable collisions with the particles.

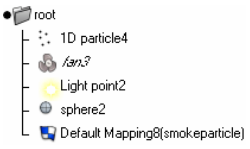
12. Now select the root level of the object hierarchy. Activate Simulation from the tool control bar icon  or from the Spec tab of the property window. This instructs the program to compute physical simulation according to the properties we defined above for the selected hierarchy level.

13. Select Post Particle Effects from the select window. Select New/Particle Discs from the popup menu. The property window now shows the default properties of the effect. Change the name of the effect to 'smokeparticle'. The other properties are quite suitable: each particle has a diameter of 2 millimeters, which should work well. The drawing mode is Replace, which is also OK, because smoke hides objects (i.e. the color of smoke replaces colors of the objects behind it). When creating fire, a more suitable drawing mode would be Add. Now close the property window.



The post effect for drawing smoke particles

14. Switch back to the Geometric Objects tab of the Select window and make sure that root is selected. Go back to Post Particle Effects tab and drag & drop the smokeparticle effect to the view window. This applies the post particle effect to the currently selected level (which is the root level in our example).



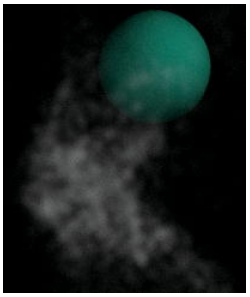
The hierarchy of the completed scene

The scene is now ready. From the Geometric Objects tab check that your object hierarchy is similar to the example image.

Having an other object than root selected while dropping the smokeparticle effect to the view window adds a new hierarchy level (which contains the selected object and the mapping). This would confuse the simulation preparations we already made.

15. You may test render the view window to see how it looks like. Adjust the Particle Size of the effect and/or the Fade value of the particle object if you need to change the thickness of the smoke.

16. Open the animation window (from the View menu) and set the frame count to a suitable value (200 - 300). Play the animation to see how the particles behave. If they are not moving fast enough, decrease the mass of particles. Another important attribute for controlling the velocity of particles is the Fluid Velocity attribute of the fan (in the Phys tab of the property window). It defines the terminal velocity of the particles.



The final step is to render the whole animation. See the rendering tutorials if you are not yet familiar with file rendering. Before starting rendering, make sure that Default Effects (or any other post processing configuration you plan to use) includes the Render Particles module.

Compositing

The post processing system has an important role when preparing rendered images for a compositing system (software/hardware that reads image sequences and blends them together using alpha channel and other information). Especially the VSL effect is useful in this context. The user can define any necessary channel modification using it.

In this section, three compositing cases are examined:

- A post processing configuration that prepares images for alpha channel keying (over a background video, etc.)
- A post processing configuration which reads an image sequence from a disk file and does the alpha preparation as a separate rendering pass
- A post processing configuration that keys an image sequence over another image sequence

Tutorial level: Medium/advanced

High quality rendering requires use of anti-aliasing. In a compositing situation, anti-aliasing has one undesirable side effect: The edge pixels of objects become blended with the background color. For example, if you render the scene over a blue background, some pixels will inevitably include color both from the rendered object and from the background.



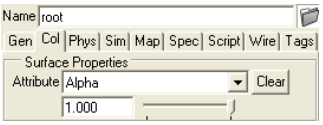
The leftmost square is a magnification of a pixel, which is split by an edge of a red object. After rendering, the color of the pixel is the average of all colors included in the pixel - dark purple shown in the second square. If the rendered scene is then 'superimposed' over a new background, the purple pixels become an undesirable glow or color error. The third square representing a 4*4 pixel array shows this situation on a white background.

A suitable VSL effect can remove this problem. The next example demonstrates this.

Example file: 'tutorprojects/postprocessing/nobackgroundantialiasing'.

1. Model a simple test scene: add a couple of objects with different colors and add a light source or two.

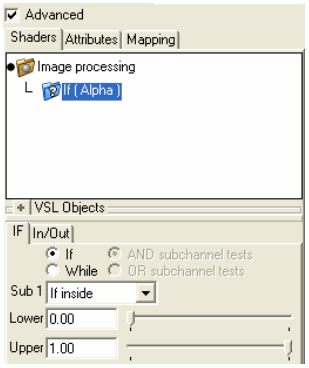
2. Compositing usually requires use of the alpha channel. Select the 'root' level of the scene, open the property window and go to the 'Col' tab. The 'Surface Properties' frame includes 'Attribute' list - select 'Alpha' from it and set the value slider below to one. In most compositing systems alpha level one means full non-transparency. If the scene contains glass (or otherwise transparent) objects, you set an alpha value of 0.5 - 0.8 for them.



Alpha=1.0 makes objects non-transparent in compositing

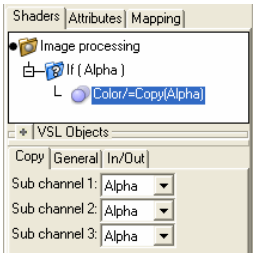
3. Go to the Post Image Effects tab of the select window. Select 'New/Post Image' from the popup menu. Then select the 'New/VSL effect from the popup. Two new objects appear to the end of effect list.

4. The property window shows the latest created effect - the VSL effect. Change its name to a more suitable one, 'removebackgroundaa'. The effect is currently empty and does nothing. Check the Advanced option to open the VSL editor, and open the VSL Objects frame.



5. Drag and drop an If object from the VSL Objects into the Image processing node. Open the popup menu of the VSL hierarchy gadget and set Input 0 to Image:Alpha. The lower part of the property window shows If specific properties. Select If Inside from the gadget labeled with Sub 1. Set Lower to 0.001 and Upper to 1.001. Now the If object tests if the Alpha channel value is non-zero - in other words, if the pixel is not a purely background colored pixel. Other sub channels labeled as Sub 2 and Sub 3 need not to be tested, because the alpha channel has no sub channels (unlike the Color channel with R, G and B sub channels).

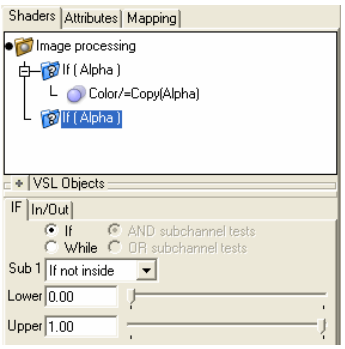
6. Drag and drop a 'Copy' VSL object into the 'If' node of the VSL tree. Using the popup menu, set 'Source' to 'Image:Alpha' and 'Operator' to 'Divide'. Set all the three sub channel selectors on the bottom of the property window to 'Alpha'.



Now the VSL code divides color of all pixels that have non-zero alpha level with the alpha level. This compensates the blending with a black background, because alpha measures the amount of non-transparency included in a pixel.

There may be some isolated pixels where the alpha level is very small ('If' operator accepted alpha level greater than 0.001). We may clean these up as follows:

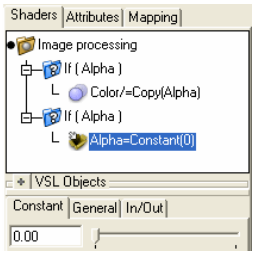
7. Drop a new 'If' object into the 'Image processing' node. Set exactly the same 'Input 0', 'Lower' and 'Upper' as in the step 5 above, but this time select 'If not inside' to the 'Sub 1' sub channel test selector.



8. Drop a 'Constant' object into the new 'If' object. Select 'Output' = 'Image:Alpha' from the popup menu. In other words, we reset very low alpha channel values to zero. This is in most cases only a 'cosmetic' operation, because alpha values below 0.001 are truncated to zero by all common file formats (such as Targa). The VSL effect is now ready.

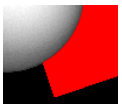
9. Select the 'Post Image' object which was created in the beginning of the tutorial, from the select window. We configure its contents next using the property window. If the scene contains particles, drag a 'ParticleEffect' object from the 'Available Effects' to the 'Active Effects' list. Then drag & drop the created VSL effect to the 'Active Effects' list. Finally rename the post image in a suitable way, for example 'No-background-aa'. Close the property window.

10. The post processing configuration is now ready for use. We can try it by rendering the view window. Drag and drop the post image (No-background-aa defined above) from the select window to the view window. Then start ray tracing.



The final structure of the VSL effect

You will see the usual shaded rendering appearing to the window, with anti-aliasing that makes object edges appear smooth and nice. After a while, the post processing replaces it with a new image where edges against the background look jagged and ugly! Do not worry, this is exactly the correct result. The alpha channel contains the information for making the edges smooth again when the image is merged over a new background.



A part of rendering magnified. Note, how interior pixels are nicely anti-aliased, but pixels against the background are not



Note

The above mentioned post processing configuration works only when rendering over a black background. Do not use a backdrop image with it.

Preparing images for compositing using a second rendering pass

When using motion blur, the background anti-aliasing elimination presented above cannot be used. The motion blur system assembles images together after post processing and the background color becomes blended with the object colors again. It is better to render the animation frames in the usual way and make another rendering pass to fix the background anti-aliasing.

Example project: 'tutorprojects/postprocessing/preparecompositing'.

1. Load in the scene modeled in the previous tutorial ('tutorprojects/postprocessing/nobackgroundantialiasing'), unless you continue directly from it.

2. We will need a short and simple test animation. Open the animation window (from 'View/AnimationWindow' menu) and set frame count to, say, 5 frames only. Select one of the scene objects, move the time slider to the last frame, activate animation recording button and move the object to a new place. Turn recording off and rewind the animation.

3. Prepare motion blur rendering (see the rendering tutorial 'Motion Blur' for details): Activate 'Motion Blur' using the 'Windows/Animation Settings' menu. Four samples are enough for the basic test. Set anti-aliasing of 'Reasonable Quality' rendering settings to 'Stochastic' and anti-aliasing level to zero. Make sure that the Output objects tab of the select window includes a 'Targa RGBA' object - if not, create one. Add a camera object to the scene.

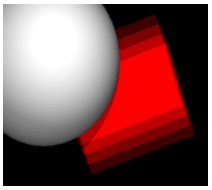
4. Open the file rendering window. Define a suitable file name (e.g. 'aa_test'). Add 'Targa RGBA' to 'Active File Formats' and remove possible other active formats. Select 'Reasonable Quality' to the 'Rendering Settings' gadget. Set 'Image Effect' to 'Default Effects' or leave it empty.



Note

Do not select the No-background-AA configuration yet. Set the resolution to a suitable test value, say 300*300 pixels. Hit the 'Render Animation' button and wait until rendering is finished.

So far we have not done anything new - the above mentioned briefly described steps only render a motion blurred test animation to a sequence of Targa files. If you take a look at one of the rendered images, it looks as in the example image on the right.



The image shows that the red moving rectangle is blurred with the black background. This will be a serious problem if the sphere and rectangle are composited over a white background. The second rendering pass fixes the Targa image sequence. This is the actual issue of this tutorial and is described next.

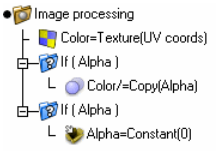
5. The scene is no longer needed, so delete all objects. Open the Animation Settings window from the 'Windows' menu and turn off 'Motion Blur'. The second rendering pass manipulates already blurred images and the motion blur should be switched off this time. Close the Animation Settings window.

6. Go to the rendering settings tab and create a new settings object. Open the property window to edit the rendering settings. Rename it as 'Compositing settings'. The purpose is to prepare a rendering settings object, which renders the empty scene as quickly as possible - the actual work is done in post processing. Set 'Undersampling/Horizontal & Vertical' both to the level 4 and clear the 'Interpolate' option. Turn off 'Box rendering' from the 'Distrib' tab. You should not use network rendering during the second rendering pass, because the simple configuration is probably faster to compute locally instead of sending image data across the network.

7. Switch to the Post Image Effects tab of the Select window, and select the VSL effect 'removebackgroundaa'. This time it should not process ray traced data (from the empty scene), but Targa disk files. Therefore, we have to modify the effect a bit. So, open the property window (if not open yet).

8. Check the 'Advanced' options of the property window if not checked. The VSL effect already contains the two 'If' blocks which were inserted in the previous tutorial. Drag a 'Texture' object from 'VSL objects', move the mouse pointer over the top edge of the upper 'If' icon before releasing the left mouse button (the cursor changes to an up arrow when insertion point is correct) and drop the texture object there. The texture should be read before 'if' tests, therefore it should be first object in the VSL tree.

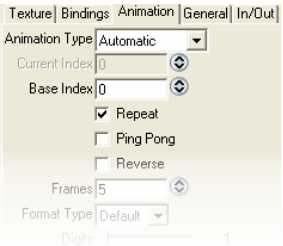
9. At the top of the property window you can see a gadget labeled as 'Image Coordinate Channel' and it is set to 'UV coords'. Therefore, using the popup menu of the VSL tree gadget, set 'Input 0' of the inserted texture object to 'UV coords'.



Color channel loader inserted

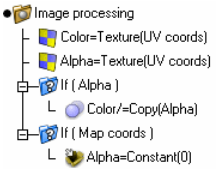
10. The inserted Texture object should load images of the motion blurred Targa sequence. Therefore, we have to define properties of an animated texture. Select the Animation tab from the bottom of the property window. Set Animation Type to Automatic for automatic frame indexing.

11. Switch to the Texture tab and enter the name of the rendered images without index but with the format postfix to the File Name gadget. For example 'AA_test.tga'.



Options of an animated texture

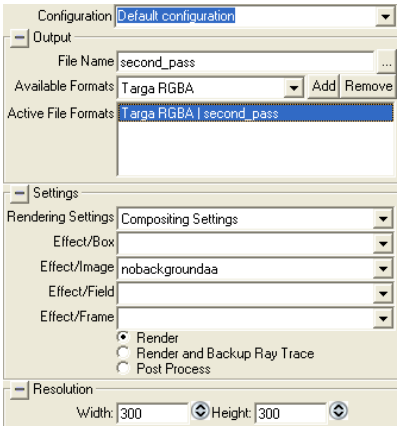
12. Make sure that the Texture object is still selected and choose Copy from the popup menu of the VSL tree gadget. Then select Paste from the same menu. A copy of the Texture object appears to the end of the VSL tree. Drag and drop it just below the first Texture object (above the If objects). Change the Output to Image:Alpha using the popup menu. Close the property window.



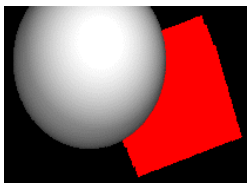
The finished VSL structure

The VSL code for the effect is now ready. Instead of processing rendered image data, the effect removes background anti-aliasing from a sequence of disk files.

13. Open the file rendering window. Change the file name to a new one, e.g. 'secondpass'. The active output list should again include only 'Targa RGBA'. Select the fast-to-render 'Compositing Settings' to the 'Rendering Settings' gadget. Set image effect to Nobackgroundaa (=the post image including the prepared VSL effect). Use the same resolution as in the first motion blur rendering. Hit Render Animation to start rendering.



File rendering settings for the second rendering pass



Background color blending eliminated

Attached is a processed sample image. There is no motion blurring left at all and the anti-aliasing is poor. Alpha channel includes the necessary information for blending the image over a new background, as the next tutorial shows.

Compositing using the alpha channel

The Targa image sequence, which was processed in the previous tutorial, includes alpha channel information and is suitable for use in various composition systems. For example, a video hardware device can insert the computer generated graphics over a live video signal.

It is also possible to use Realsoft 3D itself for doing the compositing job. Thanks to VSL, the user can freely define any kind of keying algorithm, which blends two or more image sequences. The next example shows how to achieve this.

Example file: 'tutorprojects\postprocessing\compositing'.

1. Go to the Post Image Effects tab of the Select window. Select 'New/Post Image' from the popup. Then select 'New/VSL Effect' from the popup.

2. Open the property window. Check the 'Advanced' button to edit the new empty VSL effect. Rename the effect as 'Compositing by VSL'.

3. Drag and drop three 'Variable' objects from 'VSL Objects' frame into the 'Image processing' node of the VSL tree. Set the names and types of variables using the controls at the bottom of the property window as follows:

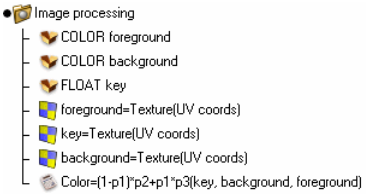
- Variable 1: name = foreground, type = Color
- Variable 2: name = background, type = Color
- Variable 3: name = key, type = Float

4. Drag and drop a 'Texture' object to the VSL tree. Set 'Input 0' = 'Image:UV Coords' and 'Output' = 'foreground'. Define the properties of the animated texture sequence exactly as in the previous tutorial, but this time use the second, processed sequence: ('Animation Type' = ''Automatic', 'Frames' = 5, 'File Name' = 'secondpass.tga').

5. Copy and paste the texture object. Change 'Output' of the pasted texture object to 'key' using the popup menu.

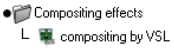
6. Drag and drop a third 'Texture' object to the end of the VSL tree. Set 'Input 0' to 'Image:UV Coords' and 'Output' to background. Select a suitable background image or image sequence. For example, use the good old 'textures/lake.jpg' image.

7. Now we have inserted three texture loaders. The remaining step is to combine their information. Drop an 'Operation' object to the end of the VSL tree. Set the 'Type' of operation (using the gadget at the bottom of the property window) to the long formula at the end of type list: '(1-p1)*p2 + p1*p3'. Mathematically oriented users may notice that this is the formula of linear interpolation. Using the popup menu, set 'Input 0' to 'key', 'Input 1' to background and 'Input 2' to 'foreground'. The default output is already 'Image:Color' which is the correct target channel.

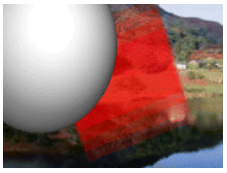


The VSL effect for image composition

8. The VSL effect is now ready. Create a new post image effect object from the select window. Rename it as Compositing effects. Insert only one effect to it, namely Compositing by VSL. Close the property window.



9. Open the file rendering window. Define a new file name, for example 'thirdpass'. The output can again be a Targa file, but you can also use other formats such as 'Windows BMP'. The VSL effect above does not maintain the alpha channel (because it is already used once for compositing). Set 'Compositing settings' to the 'Rendering Settings' gadget and 'Compositing effects' to the 'Image Effect' gadget. The resolution should be the same as in the previous rendering passes. Start file rendering by hitting the 'Render Animation' button.



The sample image above shows how the blurring appears again, but now with a new background photograph. Smooth anti-aliasing is included.

Summary

The detailed tutorial examples as outlined above, constructed all the necessary compositing tools and objects from the very beginning. In a real situation, you can reuse these example projects:

- Render the animation to targa files in a normal way.
- Load Rendering Settings and Image Effects sections from the 'preparecompositing' sample project. Edit the Texture objects of the VSL effect to include correct file names and frame counts. Render the animation second time using the same resolution, quick compositing optimized rendering settings and the loaded post processing configuration.
- Load Image Effects sections from the 'compositing' sample project. Edit the Texture objects of the VSL effect to include correct file names and frame counts. Render the animation a third time using the same resolution, the quick compositing optimized rendering settings and the loaded post processing configuration.

Of course, you can include the necessary post processing, rendering settings and file rendering configurations to your startup project. Then all the necessary tools would be available all the time.

Animated Lens Flare

Particle effect properties, such as the rotation angle of lens flare streaks, can be animated easily using key framing: simply enter a suitable attribute value on the property window, with animation recording turned on. In this example, we show how to set up a more complicated system: the rotation angle of flares based on the cameras orientation.

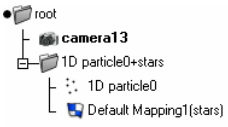
Tutorial level: Advanced

Example project: 'tutorprojects/postprocessing/rotatingflares'

1. First create the camera: activate the camera tool and hit Accept to record the current view to the camera. Turn animation recording on and go to the last frame of the animation. Open the camera's property window, go to the Gen tab and change the Rotation fields first component by say 180 degrees (for example from zero to 180 degrees). Then turn animation recording off. Camera heading will now change during the animation half a revolution.

2. Activate the particle tool and create some random 1D particles, then hit Accept button to finish the tool. Open the property window, go to Spec tab and turn off the Rendering/Scanline option. Particles will be rendered using only post effect flares.

3. Go to the Post particle effects tab of the Select window. Drag&drop the Stars effect (or another lens flare) to the view to map it to selected particles. Render the view to make sure that flares work as expected (if not, make sure that the view window's post effect configuration includes particle effect rendering).





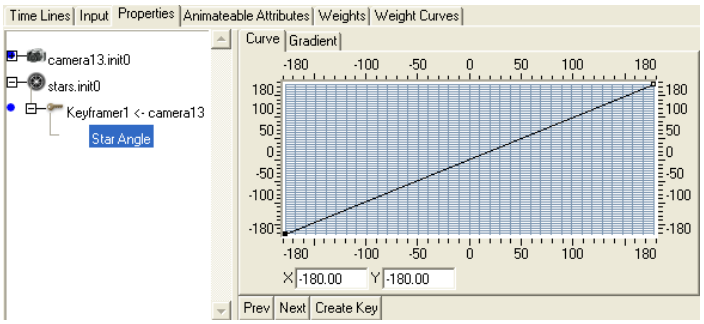
4. While the Stars effect is still selected, right click and pick the popup menu Make Choreographable on the select window. Next open the Choreographs window from the Animation pulldown menu.

5. On the Choreography window, select the stars:init0 item. Select New/Keyframer from the popup menu. Double click the stars:init0 item to open it, and select the keyframer1 object under it. Switch to the Input tab. Then go to the Select window, switch to the object geometry tab and select the camera object. On the choreography window, set Input=Geometric Object and click the Change button. The Input area is changed to show camera attributes. Select Rotate Quaternion.x from the attribute list - in other words, the cameras heading angle will act as the input for the flare animation.

6. Switch to the Time lines tab of the choreography window. Make sure that the keyframer1 item is still selected, and define a time line Start=-180, End=180 using the numeric fields on the bottom of the window. This timeline covers one cycle of the input parameter (=camera's heading angle). To ensure that all heading angle values become mapped to the range where the animation curve is defined, set the Periodic option at the bottom of the choreography window.

7. Next we define what to animate. Switch to the Animateable Attributes tab and pick Star Angle from the list. Check the Animated option above the list.

8. Finally go to the Properties tab of the choreography window. Activate the Star Angle item under the Keyframer1 choreography. Then move the mouse over the curve gadget and select Create predefined curve/45 degree. curve from its popup menu. A diagonal line appears; activate the start point and change its coordinates to X=-180, Y=-180 degrees. Activate the end point and change it to X=180, Y=180 degrees. In other words, the curve maps the cameras heading angle to the identical lens flare streak angle. Other kinds of curves could be used as well, of course.



The animation is now ready. Render it to an AVI file to see rotating streaks.

Post Shading

Tutorial level: Advanced

Example project: 'samples/rendering/post shading studio'.

In this tutorial we will examine how to collect channel information from the ray tracer in order to evaluate the basic illumination model in the post processing phase. Post processed surface shading has some advantages:

- Sampled illumination data can be filtered before the shading. This reduces noise.
- Recomputing the post shading after illumination parameter adjustments is often quicker than rendering the image from scratch. The speed advantage is remarkable when the scene geometry is complex and slow to ray trace.

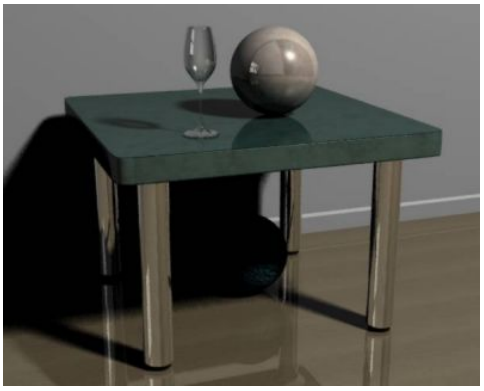
The illumination model of Realsoft 3D - and the rendering pipeline in general - is open and customizable. This means that the parameters of a post shading system are not fixed. The user can any time code a new VSL material, which changes surface illumination in a totally new way. It's impossible to reproduce the functionality of the new shader accurately in a post shading system, which assumes other kind of shading behavior.

A simple post shading system renders very quickly and allows interactive adjusting of parameters. A complex post shading solution which controls the result image in a detailed manner - providing even object or light source specific parameters - is slower to use, but naturally provides much more possibilities to adjust the image. The optimal choice between speed and flexibility depends on each scene.

The standard setup of Realsoft 3D includes a set of example components for post shading. These components allow quite a detailed control over post shading. The components are built using the standard VSL objects and post effects, and therefore they can be modified and tailored to suit all kind of projects. For example, unused elements can be deleted to speed up rendering.

Let's experiment how to use the post shading system, which is included in the program setup.

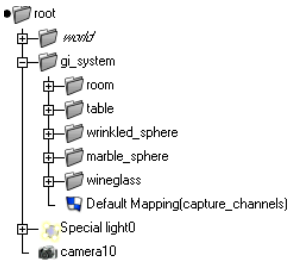
1. Model first a test scene: a floor and a wall, a table, a glass object, a reflective object, etc. Feel free to use your imagination. Add a large sphere (sky dome) around the scene. Open the property window, go to the Col tab, select Surface Properties/Attribute = Illumination and enter a value 0.5 0.5 0.5. This makes the sky dome self-illuminated. Collect all elements of the environment (the sky dome etc.) under a hierarchy level 'world' and all directly visible surfaces of the room and the furniture under another level 'gi_system'. We will later apply post shading only to the items which are directly visible in the image; this saves some rendering time.



The test scene. Shadows are noisy because of minimal area light sampling.

Add also a rectangular special light source into the room. You can leve the Quality setting of the area light source to the minimal value 0. If you test render an image, shadows will show clearly visible sampling noise. Turn also view's flashlight off, because the post shading system includes its own independent flashlight.

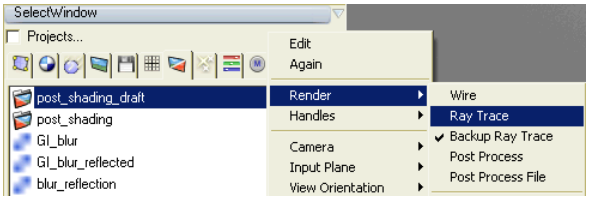
2. Go to the material tab of the select window. Choose New/From Template from the popup menu of the select window. Use the opened file browser to find and load in the material 'realsoft3d/materials/illumination/capture channels'. Select the level 'gi_system' and drag and drop the loaded material 'capture_channels' to the view window. This maps the shader to the selected level. Shading information will be collected only from the objects inluuded in the 'gi_system' level.



The channel capture shader mapped to gi_system hierarchy level

3. Switch to the render settings tab of the select window. Drag and drop the render settings object 'GI_rendering' to the view window. GI_rendering does not compute anti-aliasing, which is usually quite important. The post shading system cannot afterwards separate illumination components from antialiased pixels, and therefore ray tracer antialiasing would cause shading errors. Antialiasing must be done after the post processing by down scaling the image resolution.

4. Go to the post image effects tab of the select window. Open the popup menu and choose Paste from a File. Load in the file 'realsoft3d/posteffects/post_shading'. Drag and drop either post_shading or post_shading_draft from the select window into the view window. The draft version does not use supersampled resolution to antialiase and is therefore 4 times faster.



Set image effects to post_shading_draft using drag and drop, activate Backup Ray Trace and Render

5. Open the popup menu of the view window. Make sure that Backup Ray Trace option is enabled. Then select Ray Trace.

First the ray traced image appears, then, after a while, a post-shaded version. The post shaded version is brighter, because it adds indirect illumination to the image. The dithered shadows are now smooth.

The next step is to experiment with the post shading parameters. Switch to the post image effects tab of the select window, and open the property window.



Shadow quality improved

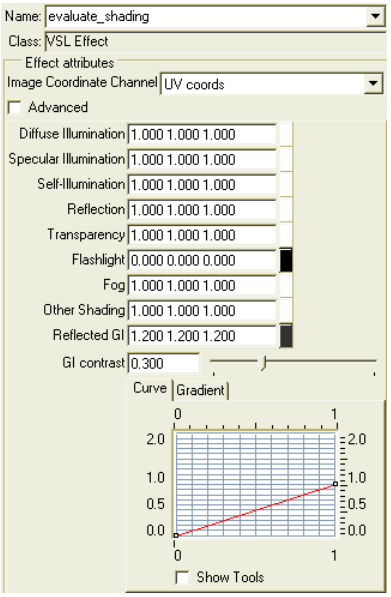
6. To make reflections softer, select the blur_reflection effect and increase the Blur Level value using the property window. Then select Render/Post Process from the view popup menu. A new image appears much quicker, because ray tracing was not computed.

7. To make shadows even smoother, select the blur_shadows effect. Increase either the Blur Level value or the Iterations value. Then apply Render/Post Process.

8. To remove all reflections from surfaces, select the evaluate_shading effect and set Reflection to black (0,0,0). Then apply Render/Post Process.

9. Change the Flashlight value of the evaluate_shading effect to pale yellow (0.4, 0.4, 0.1). Post process again. The image gets a warm fill light effect

10. Set Specularity to (1.6 1.1 1.1), then Post Process. Specular highlights become brighter and get a red tone around the white brightness maximum.



Post shading parameters



Note

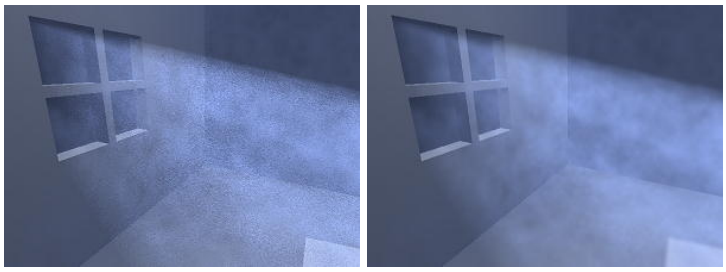
You can view the source data, from which the image is computed, using the channel menus located at the end of the view's Render popup menu. It is often useful to examine the contents of the deep backup image.

As the experiments above show, the most important component of the system is the evaluate_shading effect. It is a VSL effect, which weights the stored channels and then recombines them to a new image. The VSL structure of the effect is quite straightforward and easy to customize. By default, all weights are set to the level (1,1,1), which reproduces the original ray traced illumination. Note, that although the weights are presented as RGB colors, their range is not limited to 0..1 interval. The full list of shading parameters is explained below.

- Diffuse Illumination: rescales the usual diffuse shading. This is the most common shading component.
- Specular Illumination: the weight for all non-diffuse illumination components from light sources. Specular highlights, ambient lighting and user defined Surface Illumination shaders contribute trough this parameter.
- Self-Illumination: controls self-illumination of surfaces (usually defined as Surface:Illumination = RGB).
- Reflection: controls mirror reflections.
- Transparency: controls optical transparency i.e. refracted rays (not faded surfaces).
- Flashlight: the fill light from the camera. Replaces (not rescales) the camera flash light.
- Fog: the weight for a global fog surrounding the camera.
- Other Shading: unusual custom effects which are not included in other post shading parameters.
- Reflected GI: global illumination via reflective and transparent objects.
- GI contrast: enhances bump maps and other fine details of surfaces.
- GI control curve: the gamma correction curve for global illumination.

It should be noted that these parameters control only the first step of the recursive ray tracing. For the sake of speed and memory consumption, the system stores the channel values only from the first step of the ray trace recursion. For example, a self-illuminated surface which is seen in a mirror reflection does not become darker when you decrease the self-illumination parameter. Instead, the reflection weight controls its appearance.

The fog blurring effect, which is part of the post shading system, needs some special attention. With it, you can reduce the sampling noise of volume sampled fog shaders. Applying a pixel accurate fog shader to blurred shading components sometimes creates clearly visible errors. You can compensate this by making fog blurring as large as the maximum of the other blur components, such as the shadow blur effect.

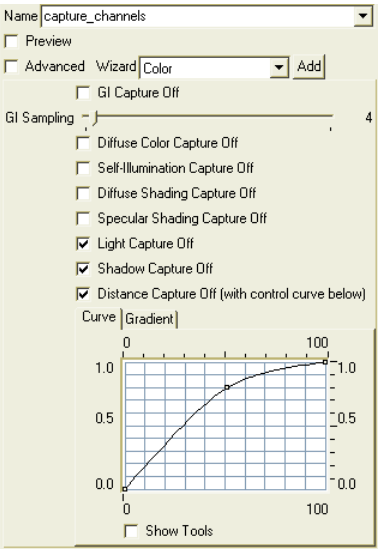


Foggy atmosphere before and after post effect blurring

Let's take a closer look at the channel capture material. Go to the material tab of the select window, make sure that the material capture_channels is selected and open the property window. The material interface displays a large number of switches 'XXX Capture Off'. With these switches, you can turn off collection of unused data and consequently speed up rendering. By default, all channels used by the post effect shading are enabled. Light, Shadow and Distance capture are disabled, because they capture additional channel data which is not needed in post shading.

This material can be used in collecting information for all kinds of post processing purposes. The switches and their relation to underlying channels are:

- GI Capture Off: do not capture global immumination.
- GI Sampling: number of GI rays (when GI capture is enabled).
- Diffuse Color Capture Off: do not capture diffuse color to diffuse_color channel.
- Self-Illumination Capture Off: do not capture self-illumination of surfaces to self_illumination channel.
- Diffuse Shading Capture Off: do not capture diffuse shading by light sources to diffuse_shading channel.
- Specular Shading Capture Off: do not capture non-diffuse shading by light sources (specular, ambient light, user defined illumination shaders) to specular_shading channel.
- Light Capture Off: do not capture the unshadowed light from light sources to light channel.
- Shadow Capture Off: do not capture the amount of shadow to shadow channel.
- Distance Capture Off: do not capture distance (z buffer) to depth channel.



The channel capture controls

Some important notes:

- Specular Shading Capture collects all non-diffuse illumination elements. For example, ambient lights are included here. The name is chosen like this because specular highlights are probably the most common illumination effect in this category.
- By enabling Shadow Capture, you can output the shadow channel into a shadow mask file. The shadow channel is not used by the post shading system.
- Distance Capture computes a non-linear curve controlled z buffer value to the depth channel. Distances from 0 to infinity are mapped to the 0..1 interval, which means that the z buffer does not overflow or truncate the values regardless of the used datatype (byte, word, float etc.). The depth channel is not used by the post shading system.

The idea behind the capture_channels material is that in includes all surface related shaders. In each shader, the material examines how the total illumination and other relevant channels change and stores the observed changes to a set of additional channels diffuse, ambient, etc. As explained earlier in this tutorial, only the first step of ray trace recursion is stored.

Technical Outline Drawing Style

A suitable post effect configuration can extract object outlines and control the line width to achieve a drawing style, which is familiar from technical illustrations.

Tutorial level: Advanced

Example project: 'samples/rendering/outline_drawing'

1. Load in 'outline_drawing.r3d'. To apply it to your existing geometric objects, enable 'Selected Sections Only' in the File Load window and uncheck 'Objects' section. Use 'Replace' loading if you want to clean up some unnecessary items. Otherwise clear the Replace option to add the items to your current libraries.

2. If the scene is not included in a closed environment, add a camera backdrop to the current camera. Otherwise empty space areas will render fully black.



Note

A backdrop plane works well in orthographic rendering. If you plan to render a perspective image, create a camera-centered analytic sphere around the scene. In orthographic rendering, backdrop plane has a constant distance from the camera. In perspective rendering, a sphere defines the constant distance background.

3. For maximal rendering speed, you can delete all materials except perhaps some outline area definitions, which you might use, from to the material library. The example configuration outputs only black and white lines so material evaluation will not be needed. Do not delete materials if you want to customize the style and include your own effects, which use the shaded color information.

4. Go to render settings tab and drag and drop 'Outline_render' to a view window. If maximal rendering speed is important, you can raise the Undersampling x and Y values from 1*1 to a higher level. Especially when rendering a high resolution image to a disk file, undersampling can be raised up to the maximal level.

The characteristic features of the Outline_render settings are:

- Distance and bump normal channels trigger more accurate sampling
- Lighting and volumetric computations are turned off and antialiasing is zero
- Antialiasing method is geometric to keep sampling intervals regular

5. Go to the Post Effects tab of the select window and drag&drop TechStyleLines to the view window. Check in the view property window that you do not have other 'per image' post effects applied.

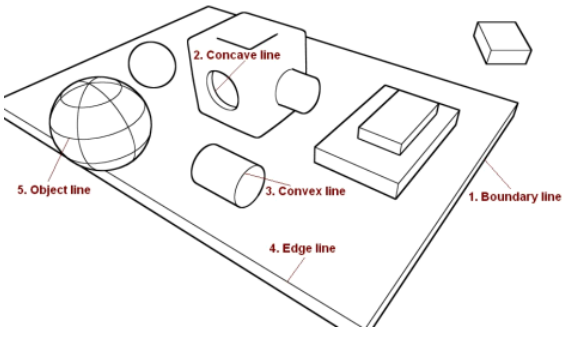
6. Render a test image. If you get too many lines, select the Edge Detection post effect and increase the Threshold and Normal Threshold values. Lowering the thresholds adds more lines to the image. If some object borders are drawn with a thin line although they should be thick, select the object_boundary_lines effect and lower its threshold value.

7. 'draw_lines' effect defines the line width range from thin to thick lines. Change it if necessary.

8. If you do not want to see all surface and object borders, select the post effect include_object_outlines and set the 'Exclude object outlines' checkbox. Note that you can customize the object outline density using materials which assign Surface:Identifier channel values. This example project includes 12 such materials. For example, the hierarchy below forces object outlines between part1 and the rest of the scene. There will be no other object boundaries.

9. 'Linewidth control' effect defines line thickness categories. Currently 5 line types are available:

- 1. Boundary line width = Width of lines between 2 disconnected surfaces
- 2. Concave line width = Width of lines between connected surfaces with a concave angle
- 3. Convex line width = Width of lines between connected surfaces with a convex angle
- 4. Edge Line Width = Base width of all lines
- 5. Object line width = Width of lines between separate geometric (connected or disconnected) objects and surfaces. This Division can be disabled or customized using materials.





Note

These lines types are not exclusive. If a line meets two or more type conditions, maximum width is used.

10. For the best possible quality, select the TechStyleLines effect, open the property window, go to Scaling tab and set Output Scale X and Y to 0.1. A lower value such as 0.25 is suitable for test renders.



Note

Edge detection depends on the image resolution and post image's Output Scale value. High resolution means that trigger channels usually change less between adjacent pixels. It may be necessary to readjust the edge detection thresholds after selecting the final output resolution.



Note

The post effect combination uses many channels, some in floating point form. The deep image requires very much ram. Use of a high number of render boxes (Render settings/Distr tab) is therefore recommended.



Chapter 9. Animations

Getting Started with Animation

Animation is the process of specifying how objects act over time. There are several ways to animate objects in Realsoft 3D. The most common form of animation is called keyframing, in which the positions, shapes, and numerous other attributes of objects are set at important (key) times. It is then left to the computer (or in the old days to junior artists) to provide the in-between frames, often called "tweens". The number of keyframes you must set usually depends on how precisely you need to control the objects in a scene. Realsoft 3D provides a number of tools to help you achieve exactly what you envision.

Before proceeding with the animation tutorials, make sure the Animation window is open. You can open this window by selecting View/Animation Window from the main menu bar. Note that the animation tutorials assume that you already know how to perform tasks like creating geometric objects such as spheres.
Tutorial project:'tutorprojects\animation\gettingstarted\movingSphere'

Tutorial level: Beginner

Simply stated, keyframing defines the starting and ending values of an object attribute, such as its position. Think about causing an object, such as a sphere, to move in a straight line. There is a starting position and an ending position. Nothing else needs to be specified. However, what if you want the sphere to move along a curved line? Clearly, you can no longer just specify a start and an end position. You need more positions to define the path.

To set keyframes in Realsoft 3D, you make modifications to an object while in animation recording mode. Let's begin by creating a simple animation of a sphere moving in a straight line.

1. Create an analytic sphere near the left top corner of the view window.

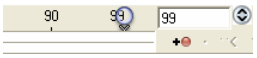
2. Click the red Record button in the animation window.



Click the red Record button to activate animation recording

3. Move the time slider to the last frame.

4. Move the sphere near the right bottom corner of the view window.



Move to the last frame using the time slider

Our moving sphere animation is now ready. Click the Record button again to exit animation recording mode and then click the Play button in the animation window to preview the animation. After the last frame, the sphere will jump back to the starting position, because the Auto Rewind option is set by default. If you like, you can clear the option from the Windows/AnimationSettings pull down menu.

The sphere moves in a straight line from the top left of the view to the bottom right. Pretty straightforward so far. So what about making the sphere move in a curve from top left to bottom right?

5. Move the time slider to frame 75. Note that as you do so, the sphere moves according to its currently defined animation.

6. Turn on animation record, and move the sphere to the bottom left of the view.

7. Exit animation recording mode.

Now click the Play button. The sphere now moves along a curved path from top left to bottom left to bottom right. As you can see, by setting more keyframes, you gain more precise control over an object's animation.

Keep the current project active. We'll use it in the next section.

Animating the color of the sphere

Tutorial project: 'tutorprojects\animation\gettingstarted\movingSphereWithAnimatedColor'

Tutorial level: Beginner

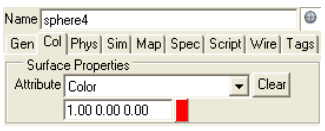
So, moving the sphere was a simple exercise in specifying where you wanted it to move from and where you wanted it to move to in the view window. But, it was mentioned earlier that all types of object attributes can be animated over time. Obviously, not all of an object's attributes can be modified through a view window. For instance, what if you want to have a chameleon sphere that changed color over time? Color is not an attribute you can control through the view window. Any object attribute can be made to change in animation recording mode by changing its value through any of the available interfaces in Realsoft 3D, such as the object property window.

Let's make our moving sphere change to red in frame 40 and blue in frame 100.

1. Activate animation recording mode and set the current time to frame 40.

2. Use the property window's Color tab to change the color of the sphere to red.

3. Now change the current time to frame 100, and set the color of the object to blue.



Modify the color of the sphere in the property window

Exit animation recording mode and view how the color of the sphere changes while you move the time slider back and forth along the timeline.

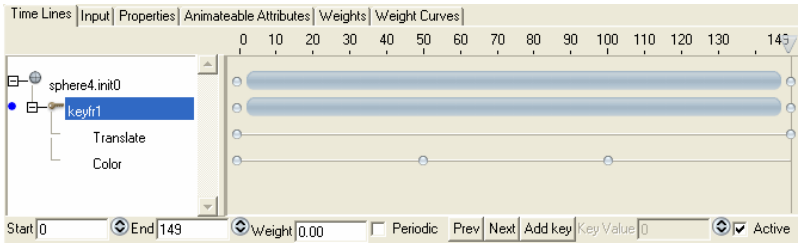
Editing key frames

So, now you have a sphere that moves and changes color. Assume that the movement and colors are okay, but the timing of the color change is not what you wanted-the color change takes too long and occurs too soon. How can you make such changes without re-recording the animation? An object's animated attributes can be edited using the Choreographer.

1. Select Windows/Choreography Window from the main menu bar (or hit the F6 hotkey).

The choreography window is opened. It shows one choreography named sphere.init0 (or something similar to this name).

2. Go to the Time Lines tab of the choreography window. Double click the 'sphere.init0' choreography object to open its sub structure. It contains one item, 'keyfr1'. Double click it, too.

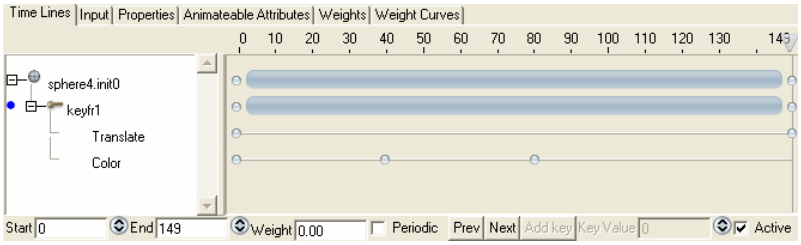


Time lines of the animated sphere

You can now see two new names below the 'keyfr1' choreography indicating that the choreography animates two attributes: Translation and Color. Each of these two attributes has its own timeline. A key indicator is shown as a small sphere on the timeline.

To move a keyframe, drag its key indicator with the left mouse button. Note how the key indicator changes its color when clicked to indicate selection. Along the timeline of the Color attribute, move the key at frame 40 to the right to frame 50, and then move the key at frame 100 down to frame 80.

To position a keyframe precisely, you can select a key indicator by clicking it with the LMB, and then enter a new frame number in the Key Value field at the bottom of the window.



Sphere turns to red in the frame 50 and to blue in the frame 80

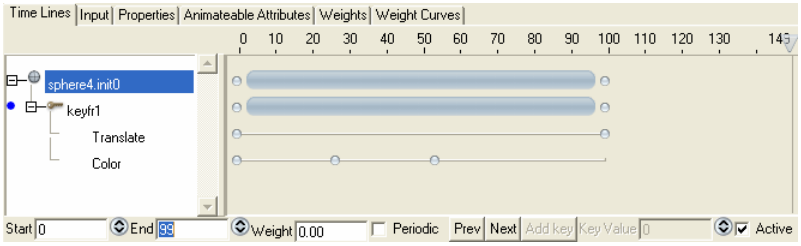
If you now move the time slider, you can verify that the color of the sphere really turns to red in the frame 50 and to blue in the frame 80.

Scaling time lines

Let's now say that you like the overall movement and color change of the sphere, but that the overall timing is too long. Rather than 150 frames, you need the animation to play in 90 frames. This is easily accomplished using the choreography window. If you adjust the end knob of the 'sphere.init0' time line, the result is that all animation components of the sphere object are modified. All the keyframes are scaled accordingly.

To change the length of the 'sphere.init0' time line, do the following:

1. Drag the ending time indicator of the 'sphere.init0' timeline from frame 149 to frame 90 (or select the choreography and use the Start and End fields to define the timeline numerically).



Sphere.chor0 choreography is changed to execute within frames 0 ... 90

If you now play the animation, the sphere moves in a curve to the bottom right corner of the view window during the frames 0 to 90 and then stops.



Note

The time line window can be zoomed and panned like many other windows in the program. To pan, press Ctrl and drag with RMB. To zoom, press Shift and drag with RMB. You can also adjust the vertical splitter control between the choreography name list and the time line canvas.

In addition to this, there are couple of other modifiers specific to the time line window:

- *Alt* + *drag* a time line modifies only the time line of the choreography without scaling the actual keys with the time line.
- *Shift* + *drag* snaps the selected knot to the nearest frame.

Animation Controls

Besides creating an animation, there are a number of *controls* for the animation system that you need to set, such as the length, and the type of time code and video system to use. This tutorial shows you the animation controls that are available in Realsoft 3D.

Animation length

You can change the total length of your animation at any time using the frame count field in the animation window.



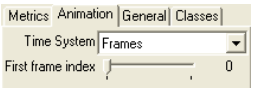
Animation consisting of 150 frames

To add new frames to your animation (increase the length), enter the desired value in the field and press the Enter key. A dialog appears asking if you want to scale the timelines for any animated objects.

To add new frames without affecting the length of any currently defined animations, select No (do not scale the time lines). If you want to scale the existing animation timelines to include the new frames being added, select Yes.

You remove frames (shorten) from an animation by the same process as adding frames: type a new length into the frame count field and press Enter. However, be aware that if you decrease the total number of frames to less than the length of any particular choreograph, and do not scale the timelines, such choreographs will be affected. Again, you always have the option of using the Time Lines tab of the choreography editor to make adjustments.

Setting the time code and video system



Selecting time units

Before you start an animation project, you should set the type of time system and video system to use. To access the Time code settings, open the Preferences window from the main menu

File → Preferences

and click the Animation tab.

For the Time System, you have three options:

- Frames (default). Time values are in units of frames. The number of frames needed to comprise one second depends on the video system setting.
- Real Time. Time values are in units of minutes, seconds, and frames. This is the standard time code system used by video equipment, and is also dependent on the video system setting.
- Seconds. Time values are shown in units of seconds. This system may be more natural if you are doing physical simulations, or are using physical formulas within an animation.

You can change the time system at any time. The user interface automatically adapts to the selected time system. For example, if you select the time code value, three fields (minutes, seconds, frames) are used to show animation timing. Also, changing the time system does not affect any existing motion definitions. The time system only affects how the user interface displays time attributes.



The time line showing real time units

The video system can be selected from the Animation Settings window. You can find this window below the select window in the default startup project, and in the Windows pull down menu.

Suspending and resuming animation rendering

Realsoft 3D allows you to start and stop the rendering of a project or scene at any time, without losing the work that has been performed. This is quite useful when rendering long animation sequences.

Assume that you have started rendering a long animation, say 1200 frames, and you want to stop the rendering for some reason, and then resume it again later from the point at which it was stopped.

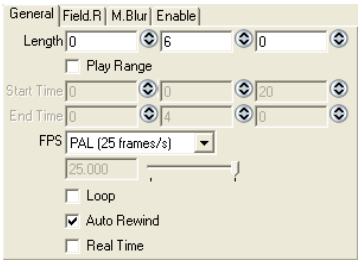
The controls for doing this can be accessed either from the file rendering window, or from the main menu.

To stop rendering an animation, press the Stop button on the file rendering window, or make the following selection from the main menu: Animation/Playback/Stop. Then use File/Save As to save the entire project. The current time is saved with the project. You can then load the project again at any time, and click the Render Animation button on the file rendering window to continue rendering the project.

Defining project-specific animation settings

Every time you create a new animation, you have to define a number of project-specific animation options. For example, you have to decide whether to use motion blur. You might also want to use field rendering. These options can be defined through the Animation Settings window. You can find this window below the select window in the default startup project and in the Windows pull down menu.

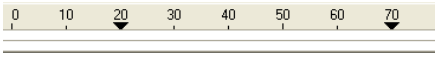
If you select the Motion Blur option, you also need to specify the desired sample count in the Samples field. The higher the value, the better the quality of the generated motion blur (Note that higher values will affect your rendering time).



Project-specific animation settings

If field rendering or motion blur is activated, the animation playback window will show gadgets that allow you to control and select time in units of fields and motion blur samples, as well as frames. This allows you, for example, to access the time line at finer resolutions, such as Frame=2, Field=odd, and Sample=3, and then, for instance, modify an object to create a key frame.

You can also define Start frame and End frame attributes which allow you to play desired sub range of the animation. Check Play range and enter desired start frame and end frame values to activate sub range feature.



Play Range option activated, start and end frame knots shown

This also adds range knots to the time slider. You can drag the knots to adjust the frame range.

Choreography Basics

In the Animation/Getting Started tutorial, you were introduced to a few of the more common features of the choreography window, such as how to move keyframes, and scale an entire timeline. You were also shown that keyframes you define, whether through the view window or by modifying an attribute on the properties window, end up as a choreograph attached to the object. So far, we have only scratched the surface of what can be done using the choreography tools.

Although it is true that you can keyframe animations to your heart's content using the animation record mode and setting keyframes interactively (through a view or property window), this approach will take you only so far. The choreography system provides a number of tools and methods for creating any type of animation, from simple to very complex (for instance, a combination of simple mouth movements to create *individualized* facial expressions).

This tutorial introduces you to more of the choreography system's features, such as editing animation curves, loading and saving choreographs, and building hierarchical animations. More choreography features are presented in the Choreography Essentials tutorial.

To begin, delete all previous test objects from your current scene, and do the following:

Set up an animation timeline of 120 frames.

Set the view window to a front view.

Create a sphere near the left edge of the view window, roughly centered vertically.

Turn on animation record mode and set the time slider to the last frame (the index of the last frame is 119 or 120 depending on the indexing convention selected from the Animation tab of the Preferences window). Then move the

sphere in a straight line across to the right edge of the view window. Use the Shift modifier or the red translation handle for accurate horizontal translation.

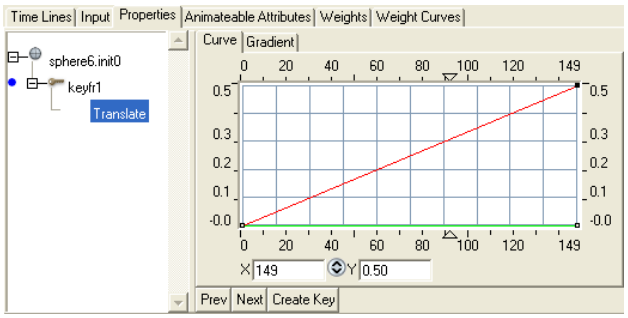
Exit the animation record mode.

Open the choreography window, and click on the Properties tab. Double click the sphere.init0 choreography. Double click the keyfr1 object under the init choreography. Click on the Translate attribute.

Whenever you modify an object in the animation recording mode, the animation system saves all your modifications to a set of animation curves, grouped under a choreograph name. These curves determine how the attribute changes during the animation. The choreography window provides a way to edit these animation curves directly.

Let's have a look at the straight line movement animation we just created for the sphere. In the Properties tab, you should see the following:

For object translation, the red line represents the X axis, green the Y axis, and blue the Z axis. Currently, you see an ascending red line because you moved the sphere in the X direction. The green and blue lines have a constant value zero (the green line hides the blue line below it). Study the graph of the animation curves for a moment. The X axis of the graph shows the number of frames of your timeline, and the Y axis shows object position in the current units. Note the positions of the "knots" of the lines (there should be one at each end on both lines).



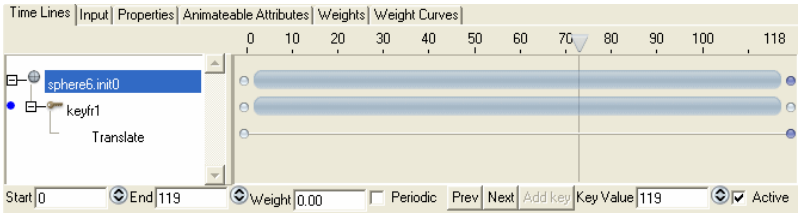
The curves defining the translation of the sphere



Note

The curve display graph can be zoomed and panned by dragging with the right mouse button while holding down the Shift or Ctrl key, respectively. From the popup menu of the curve graph, you can hide any of the red, green and blue curves in order to make editing of a single curve easier. These tools become available also when you open the curve editing toolbox by checking the Show Tools option.

Now compare what you see in the animation curves graph to the information shown on the Time Lines tab.



A time line showing keyframes for the sphere

Now return to the Properties tab. Try moving the knots on both the X and Y animation curves, and observe the results. As suggestions, do the following:

1. Move the end knot of the red curve to a value of 80 (this changes the value shown in the X field below the graph). Keep the value of the Y field the same. Move the time slider manually to view the animation. Also look at what has changed on the Time Lines tab.

2. Return the end knot of the red curve to its original location, and move the beginning knot to an X value of 40. Run the animation, and then check the Time Lines tab.

3. Move the beginning knot of the red curve to a Y value of 0.3. Note that you can enter values directly in the X and Y fields, and press the Enter key to set the new value. Run the animation, and then check the Time Lines tab.

4. Set the beginning knot of the red curve back to its original position. Now move the end knot of the green curve to a Y value of 0.4, and observe the animation and the changes in the Time Lines tab.

All the manipulations we have done so far are rather basic; however, maybe you can now envision the type of control you have available for creating animations. Moving on to more interesting things, let's assume you want another keyframe for the movement of the sphere along the X axis. You could go back to the view window and use the animation record mode, but this is not necessary. Try the following:

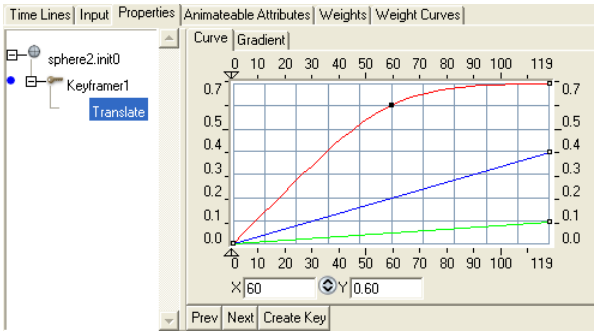
Make sure the knots of both the red and green curves are set back to their original positions. Now place the cursor over the red X curve at the point where it intersects the line representing frame 60, and left click once. A new knot is inserted. Best of all, this new knot is a keyframe. Verify this by viewing the Time Lines tab. Now you can move this new knot, changing the shape of the X curve, and thus, the movement of the sphere along the X axis. Move the new knot to a value of 0.6 for the Y field, and run the animation.

What has happened? The sphere still moves in a straight line, but you have modified how it does so. Now the sphere moves most of the distance across the screen by frame 60, and then takes another 60 frames to travel the remaining distance. The result is as if the sphere has encountered some sort of force field or other type of external resistance. (This isn't really the case, but by adding other elements to the scene you could certainly make your viewers *believe* the sphere is meeting some type of resistance, and isn't that the point?)

Next, let's modify the movement of the sphere along all three axes. Adjust the translation curves to look like the following:

These are the settings:

- X (red) curve is left as it was from the last adjustment. Knot 1: 'X=0, Y=0.0'; Knot 2: 'X=60, Y=0.6'; Knot 3: 'X=119, Y=0.7'
- Y (green) curve settings. Knot 1: 'X=0, Y=0.0'; Knot 2: 'X=119, Y=0.1'
- Z (blue) curve settings. Knot 1: 'X=0, Y=0.0'; Knot 2: 'X=119, Y=0.4'



Modifications to all three translation curves

Now play the animation. First view from a front view. Then rerun it and view it from a top view. Now the sphere moves along all three axes. As you should now start to realize, this form of editing, when applied to other parameters such as Rotation, Scale, Skew, Color, Transparency, etc., gives you very precise control over your animations.

Let's do one more adjustment before closing this discussion, which is, let's make the sphere move along a circular path in the XY plane, using the starting and ending positions that we will define along the X axis.

To do this, set the end point of the blue curve back to a Y value of 0.0, and set the X curve back to its original state. To do this, select the knot at frame 60 that you added to the red curve (left click the knot to select it). Right click and select Delete Points from the popup menu. Now we have no Z motion, and the X motion is set to what it was originally.

Now we need to do two things: make the X movement return to its starting point at frame 119, and adjust the movement of the sphere in Y so that the sphere goes to the bottom and the top of the screen when the sphere is at the middle of the screen in its movement along X.

To modify the sphere's movement in X:

1. Add a knot to the red curve at frame 60. Move this knot to a Y value of 0.7

2. Move the end knot of the red curve at frame 119 to a Y value of 0.0

3. Right click over the curve graph and select 'Types/Polyline' from the popup menu. If you now play the animation, the sphere moves back and forth across the screen.

To modify the sphere's movement in Y (the green curve):

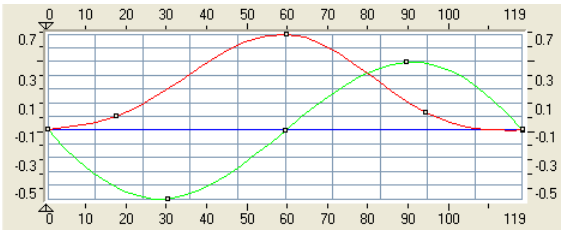
5. Move the end knot of the green curve to a value of 0.2 so you can more easily see the curve. Add new knots to the curve at frames 30, 60, and 90.

6. Set the Y value of the knot at frame 30 to -0.5.

7. Now set the knot at frame 60 to 0.0, the knot at frame 90 to 0.5, and the knot at frame 119 to 0.0

8. Play the animation.

Note that the sphere does not really move along a smooth circular path. This is because we earlier set the animation curve type to Polyline. Right click over the curve graph again and select Type/Nurbs from the popup. Add two new points to the red curve and bend it to a bell shape as shown in the figure below. Now play the animation. The sphere moves along a (roughly) circular path, returning to its starting position.



The final translation curves



Note

The animation system uses special PNurbs curves by default. These curves attempt to pass through the key points as smoothly as possible. If you need to combine smooth, round bends and sharp edges in a same curve, use the Bezier option available in the curve editor's popup menu.



Note

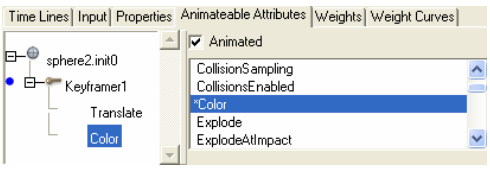
Animated transformations are defined in object's space. The red X curve does not show you movement in 'world' space but with respect to its own native orientation. The red axis of the object handles shows you this direction. Therefore, you can edit animation components of an object independently of the motions of its parent levels. For example, a butterfly moves its wings up and down. When you insert the butterfly to a moving space ship, up direction may become unclear - however, editing the wing motion remains quite as easy.

This section has only scratched the surface of what you can do by editing animation curves. As always, the best course of action is to experiment. Save the project you just completed because we'll use it in the next section.

Adding animated attributes to a choreograph

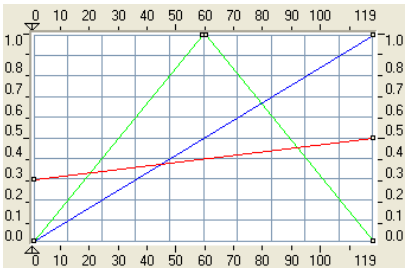
In the previous section, we used the Properties tab to modify an animation that was initially defined interactively. If we now want to add rotation or scaling effects, for instance, is it necessary to do this first by using the animation recording mode? The answer is No. In fact, you do not even have to set up an initial animation in animation recording mode at all (more about this in the next section).

Using the animation we created in the previous section, let's animate the color attribute of the sphere so it changes color as it moves. First, go to the Time Lines tab and select the choreograph 'keyfr1' - we will define the color animation using an existing keyframer object. Then switch to the Animateable Attributes tab, Scroll down the list of attributes, highlight Color, and then click the Animated checkbox. An asterisk to the left of an attribute name indicates that it is animated (even if the animation settings do nothing).



Select the Color attribute from the list and click the Animated checkbox

Go to the Properties tab and click the new Color attribute in the choreography list to select it. Right click over the graph window and select Create Predefined Curve/Constant curve from the popup. Three animation curves are created for the Color attribute: red, green, and blue lines. These lines will be set one on top another initially. At this point, the vertical scale of the curve editor may be too small (0 - 0.5). Select Set Minimum and Maximum Values from the popup menu, change Max Y to 1.0 and click OK.



Curves defining object color

Move the knots of the red, green, and blue curves so that you have something similar to that shown in the following illustration. Use the curve tools available via the the Show Tools option to make editing easier.

Note that for the green color component, two knots were added around frame 60 (the curve type is the default of PNurbs) to achieve straight line segments. However, since all the other color curves are straight lines, you could achieve the same effect by adding only one knot to the green curve at frame 60, and changing the line type to Polyline.

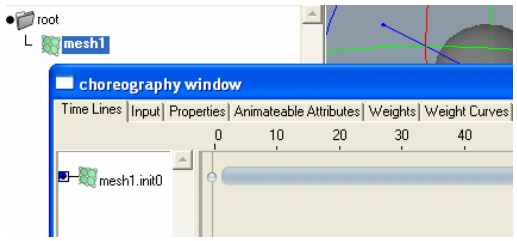
Play the animation. You now have a sphere that changes color as it moves, and you were able to add this effect solely in the choreography editor.

Creating a choreograph for an object

As mentioned in the previous section, it is not necessary to use the animation recording mode to create an initial choreograph for an object. Realsoft 3D's choreography editor can be used exclusively to construct your animations.

To demonstrate this, first delete the sphere from your current scene to get a clean start. We will use the same frame count as in the previous example, 120 frames.

Create a NURBS sphere at the left side of the view window. Select the sphere in the select window, and then right click to display the popup menu. Near the bottom of this menu is the Make Choreographable option. Select this menu, then display the choreography window, and go to the Time Lines tab. There is now an entry called 'mesh*n*.init0'. If you double-click this choreograph, no sub items are displayed because it currently does not have any animated properties.



A new choreograph

Let's make this sphere expand and contract as it moves back and forth across the view window. First we define the motion to make the sphere move across the view and back to its starting position:

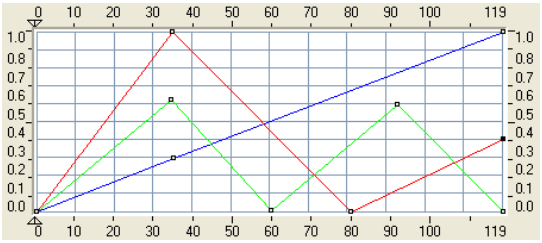
1. On the Time Lines tab, highlight the mesh*n*.init choreograph. Move the end of the time line to frame 119.
2. Select New/Keyframer from the popup menu. Double click mesh*n*.init to open its sub structure.
3. Click the keyfr1 choreograph to select it.
4. On the Animateable Attributes tab, select Translate from the list of attributes, and set the Animated checkbox.
5. Open the keyfr1 choreograph with a double click and select the new Translate attribute under it.
6. On the Properties tab, select Create Predefined Curve/Constant curve from the popup of the curve graph.
7. Select Type/Polyline from the popup.

Let's change the editing area of the curve graph. Right click over the graph and select Set Minimum and Maximum Values. Set the fields as follows: Min X = 0; Max X = 119; Min Y = -0.8; Max Y = 0.8. Click the OK button. Uncheck Green and Blue curves from the popup menu; only the red curve remains visible. Click on this line at frame 75 to

add a new knot, and move it upwards to a Y value of 0.8 (you may need to adjust the Y value to match the scale of your view window. The main point is to have the sphere move from one edge of the view to the other and then back).

Now let's animate the sphere's scale. Find the Scale attribute in the Animateable Attributes tab, select it and check Animated. Return to the Properties tab and select Create Predefined Curve/45 deg curve from the popup. Select Type/Polyline from the popup. Add points to the X and Y scale curves so that they look as follows:

Now play with the time slider to see how the scale curves affect the shape of the object. The Scale curves, as well as Translation curves, define a change from the initial state. This means that when a scale curve reaches the Y value 1, the total scale value is 1+1=2 (the initial default scale is one), doubling the size of the object in the respective dimension.



Animate the Scale property

Loading and saving choreographs

So, you have created a nice animation that moves an object and causes it to "pulse", increasing and decreasing in size. What if you wanted other objects to do the same thing, or wanted to save this motion into a library of animation effects? Well, choreographs can be saved to a file. This allows you to create choreography libraries.

To save animation data of an object to a file, such as scaling and motion of the sphere created earlier, select the top level chorograph of the sphere (mesh.init0) and choose Save from the pop-up menu. Save it as 'Pulse'.

Now create another object, for example an analytic cube. On the select window, choose the popup menu Make Choreographable. On the choreography window, select the init choreography of the cube. Then select Load from the popup menu. From the file loading dialog, select the 'Pulse' choreograph you just saved, then play the animation. The cube moves and pulses in exactly the same way as the sphere did.

Rules for Attaching Choreographs to Objects
In general, any choreograph can be attached to any type of object even if the new target object does not have a common set or subset of object attributes. When a choreograph is attached to a new target object, all features that the target object does not support are removed. For example, an analytic sphere object defines a Center attribute, but a NURBS curve does not have a Center attribute. A choreograph created for an analytic sphere that only contains animation of the Center attribute would produce no effect if assigned to a NURBS curve.

Deleting animation effects

If you are working with animating an object, and decide that a particular choreograph is not what you want, you can easily delete it from the object. If a choreograph has been saved as a file previously, deleting it from an object does not delete the choreograph file, just its attachment to a particular object.

Using our example from the previous sections, if you want to delete the 'Pulse' choreograph from the cube object, do the following:

1. Go, for example, to the Time Lines tab of the choreography window and select the keyfr1 under the mesh.init0 choreograph.

2. Select Delete from the pop-up menu.

You can also remove animation effects from a certain attribute. For example, maybe you want the Pulse choreograph just for the scaling that causes the pulsing behavior, and not the movement. To remove the defined animation effects from the Translate attribute:

1. In the choreography window, click the Translate attribute under the keyfr1 choreography to select it.

2. Select Delete from the popup menu.

You can also remove all choreographs from an object. To do this, select the topmost 'init' choreograph and apply Delete from the popup menu. Alternatively, you can activate the object from the select window and choose Delete Choreographs from the popup menu of the select window.

Exchanging choreographs

In the tutorials above, a choreography was assigned from a NURBS mesh to a cube by saving and reloading it. You can also assign a choreography to a new object using drag & drop.

For example:

1. Create a cube. Turn animation recording on, go to the last frame, then rotate and move the cube. Turn animation recording off.

2. Create a sphere. Select Make Choreographable from the popup menu of the select window.

3. Multi select the sphere and the cube or their parent level (because chorerography window may have the Only Selected Objects option enabled).

4. Open the choreography window. Open the Cube.init0 choreography by a double click. Drag the keyfr1 choreography under it over the sphere.init0 choreograph and drop it there. This attaches the dragged choreography to the sphere.

If you now play the animation, both objects will follow the same choreography.

Choreography Essentials

Each animation choreography can be controlled through a set of parameters. Therefore, in many cases it is often better to record different kinds of animation effects as separate choreographs. For example, a bouncing sphere animation can be accomplished by combining two motions: the *forward* movement and the *bounce* movement. If created in this way, such choreographs can be used to make many types of "bouncing sphere" animations by editing the parameters of the choreography rather than keyframing a new animation.

In addition to its time line, a choreography has a number of other parameters that can be used to control timing.

Parameter Name	**Description**
Weight	Controls how strongly a certain choreography affects an object. For example, you could create a mouth movement that consists of 10% of the 'smile' and 40% of the 'open mouth' choreographs.
Periodic	Repeats the choreography. The time line is used to define the start and end times for one period (cycle). For example, record one 'bounce' cycle for a sphere, and make it bounce all the time by setting this option.
Active	Turns a choreography on/off. Often, it is easier to add new motion components when other components are turned off.

The weight parameter can be animated. For example, a talking head can be created by animating the weight parameters of various basic expressions.

Animating weights

Each choreography can be assigned a *weight*, which controls how strongly it affects an object. To see how this works in practice, we'll create a sphere with motion defined by two separate choreographs.

1. Create an analytic sphere near the top left corner of the view window.

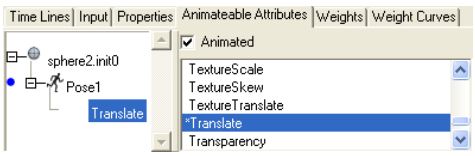
2. Select Make Choreographable from the popup menu of the select window.

3. Open the choreography window and select the 'sphere.init0' choreography. Select New/Pose from the popup menu of the choreography window.

4. Double click the init choreography to open its sub structure. Select the pose choreography under it.

5. Go to the Animateable Attributes tab, find and select Translate from the attribute list and check Animated.

6. Move the sphere horizontally near the top right corner of the view window. You have now created the first choreography for the sphere. Apply the Rename tool of the popup menu to change the name of the pose choreograph to 'right'.



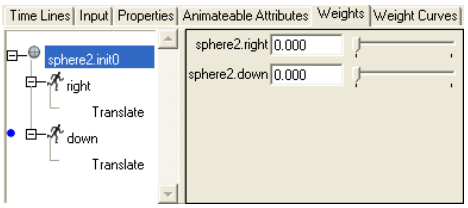
Add Translate attribute to the pose

7. Go to the Weights tab and try the sphere.pose1 weight slider. The sphere moves between its original position and the new position. Leave the weight slider to zero.

8. Select New/Pose again. By default, it inherits the attributes of previous choreographs. The new choreography is automatically the current choreography (a little yellow triangle on the choreography window marks it), and therefore new posing actions will be stored to it. So, just move the sphere vertically to the bottom left corner of the view window. Rename the second choreography as 'Down'.

We now have two choreographs. If you now play with the time slider, nothing happens, because the poses just define static attribute states, which do not depend on the time.

By combining these two choreographs with different weights, you can create 'bouncing sphere' animations easily. For example:



Two choreographs created for the sphere

9. Select New/keyframer from the popup menu. Then enable animation recording. The weight modifications presented below will be recorded to the keyframe choreography.

10. In the first frame, using the Weights tab of the choreography window, set:

- sphere.right = 0
- sphere.down= 0

Since the effect of both choreographs is now zero, the sphere returns back to its original position near the left top corner of the view window.

10. Move the time slider to half way in time and set weight sliders to:

- sphere.right = 0.5
- sphere.down= 1

This takes 50% of 'right' position and 100% of 'down' position.

11. Move the time slider to the last frame. Then define:
- sphere.right = 1
- sphere.down= 0

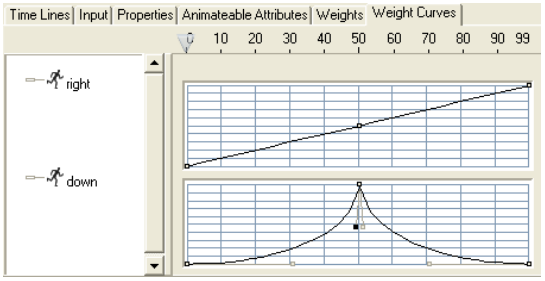
12. Turn animation recording off and play the animation. You can now see the result of the two blended choreographs: the sphere travels along a curved path across the screen.

We can improve the result by editing the weight curves as follows:

13. Go to the Weight Curves tab of the choreography window. You can now see two curves. They indicate how strongly each choreography affects the objects as the time changes.

14. You can edit these curves directly to fine tune the blending of the choreographs. The weight curve for 'right' choreography is OK, but change the type of the 'Down' curve to Bezier from the popup menu of the curve graph. The figure below shows a better weight curve for the 'down' choreography - it is easy to edit the Bezier curve to this kind of shape.

The strength of the first choreography increases with time. In other words, the 'right' choreograph affects more and more as the time proceeds, making the object to move vertically. The 'down' weight curve has a peak in the middle of the animation. This makes the sphere to fall down and bounce back. The combination is a moving and bouncing sphere.



Bouncing created by animating weights of 'right' and 'down' choreographs

Hierarchical animations

In the modeling specific tutorials we have demonstrated how geometric objects can be constructed to hierarchical structures using level objects. For example, you can create a 'leg' object, which consists of two sub objects: a thighbone and a shinbone.

Correspondingly, you can attach a choreography to the entire level object and another choreography to the its children.

To demonstrate this, let's create walking legs.

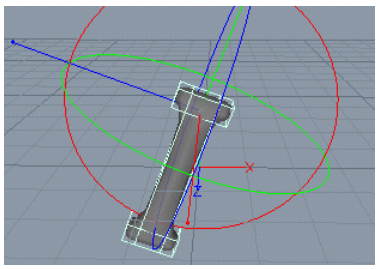
Tutorial level: Medium

Example project: 'tutorprojects\animations\gettingstarted\walking_legs'

First, we need to create a leg consisting of two sub objects: a shinbone and a thighbone.

1. Create a simple object representing shinbone (for example, edit a SDS cube). Move the pivot point to the kneecap (the leg object will be rotated about the knee).

2. Luckily the thighbone looks pretty much like the shinbone. So, duplicate the shinbone and move it upwards so that the lower endpoint of the thighbone matches the upper endpoint of the shinbone.

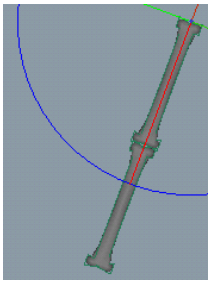


A simple shinbone in the initial 'forward' position



The structure of the leg

3. Now select both the objects and apply 'Drop to a level' from the select window's pop-up menu. This creates a new level object and drops the selected bones inside the level. You can rename the new level object as 'left leg'.



4. Select the left leg level in the select window and move its pivot point to the hip. We do this because the entire leg will be rotated about the hip.

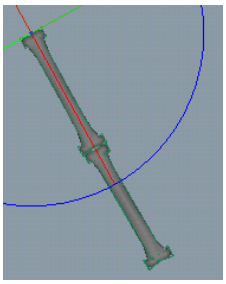
The hierarchical leg model, in its initial position, is now ready. Let's animate it.

5. Set animation recording on, jump to frame 20 and use rotation handles to rotate 'left leg' about its hip so that the leg now represents its back position.

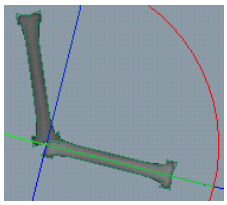
6. Jump to frame 40 and rotate 'left leg' back to its original position.

7. Now, jump to frame 30 and rotate the shinbone so that it represents the back position.

8. Go to frame 40 and rotate the shinbone back to its original position.



The whole leg rotated backwards in frame 20

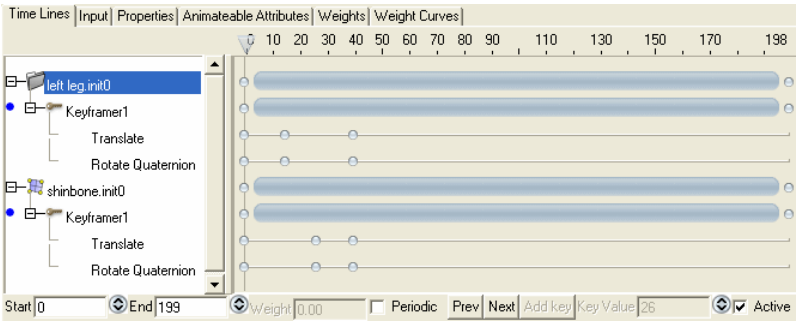


The shinbone rotated in frame 30

Reset animation recording and play the animation. The shinbone rotates about the kneecap while the entire leg rotates about the hip. Because of hierarchical construction, the parameters used for rotating the shinbone are rotated with the thighbone.

However, the leg takes only one step! As soon as the time reaches the last key frame 40, the leg stops. How can we make the leg walking all the time?

9. Select the left leg object and open the choreography window. Go to the Time Lines Tab. Open the time lines of the left leg and the shinbone, as well as the time lines of the included keyframe choreographs. Although the time lines cover the entire animation, there are keys only during the first 41 frames.

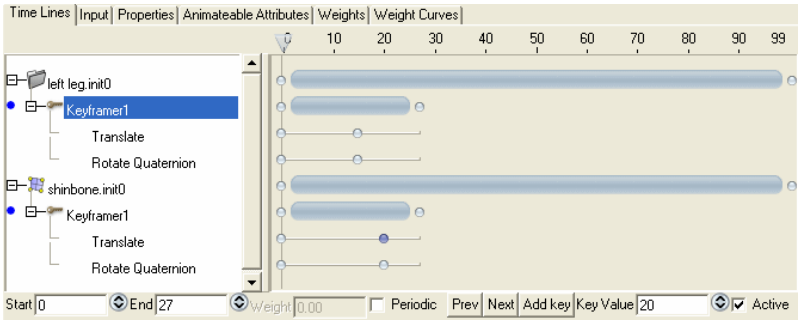


Choreographs rotating the left leg and shinbone

10. First we normalize the time lines so that they correspond the actual key range. Click the end knob of keyfr1 timeline under the left leg:init0 timeline. Press Shift key down and click the end knob of the keyfr1 timeline of the shinbone object. Now both end knobs should be selected. Press also the Alt key down; then Alt-Shift-drag the end knobs to the frame 40. Then you can release the hotkeys. Dragging with the Alt key down scales a timeline but does not scale the underlying keys.

Note that it does not matter, which choreography item is selected when you drag the time line knobs in the time line graph. However, the numeric Start and End fields at the bottom of the window show the properties of the selected

choreography, and therefore selecting a choreography before dragging its time line knob helps to make the operation accurate.



Keyframe time lines normalized and made periodic

11. Select the left leg/keyfr1 choreography and set the Periodic option from the bottom of the choreography window. Set Periodic for the shinbone/keyfr1, too. Play the animation. The leg object now repeats its motion several times.

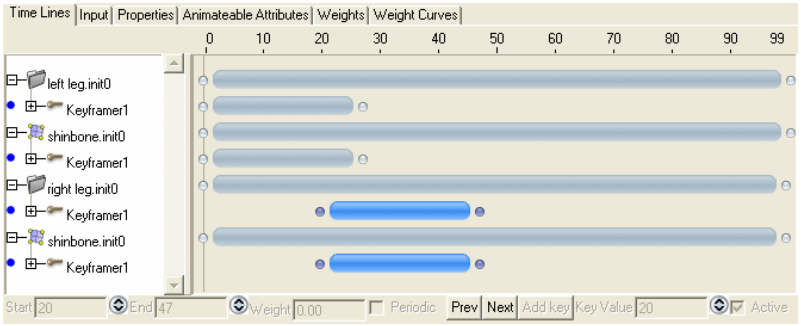
How to create another leg?

12. On the select window, select the left leg object and apply the Duplicate tool from the popup menu. Rename the copy as to 'right leg'.

Now we have a new problem. We have two identical legs, which also function identically.

13. Multi select the left leg and the right leg. Go back to the choreography window. Select the choreography right leg/keyfr1. Then click in the middle if its time line to select it. Shift click the time line of the choreography right leg/shinbone/keyfr1 to multi select it, too. Then Shift drag both time lines to the right until they start from the frame 20.

14. You can now close the choreography window. As the final step, take a top view and move the legs slightly apart from each other. Play the animation to see how the legs walk.



Time lines of the right leg shifted

Path Animations

In the Getting Started tutorial, you animated a sphere that moved roughly in a circle - this demonstrated how to create a choreography based on *keying* an object's position at various times. As you may have noticed, to refine the movement to be circular, you had to make adjustments to the animation curves in the choreography editor. Although this method is fine for certain types of movement, Realsoft 3D provides other options. One of these options is to animate an object along a path, hence the term, *path animation*.

This tutorial explores some of the ways you can create path animations, which includes attaching an object to a two dimensional object such as a line or a three dimensional object such as a surface.

Tutorial level: Medium

Example project: 'tutorprojects\animation\paths\movingSphere'

Remember the animation of a sphere traveling in a circle you created in the Getting Started with Animation tutorial? Now let's do that same animation using a path.

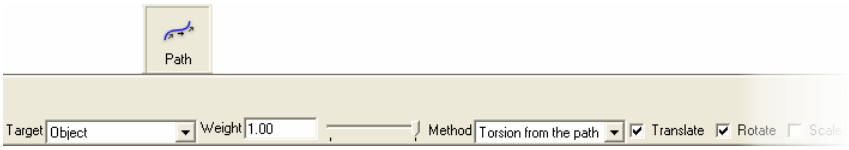
First, set up your environment. Then create an analytic sphere in the upper left corner of the view, followed by a NURBS circle centered approximately at the view's midpoint with a diameter of about 3/4 of the view's size.

Select the sphere and then the circular curve (the order is important), and then choose Tools/Lattice Mapping/Create Path Animation from the main menu (or, if you are using the startup environment, open the Lattice Mapping tab, and click the Path button. This presents a control bar with the path tool's options.



Note

It is important that you select the objects in the specified order. If you select the objects in reversed order, you would get a curve moving along the surface of the sphere.



Options for the Path tool

For this first attempt, use the default settings, so just click Accept. Now play the animation and note what is occurring. First, the sphere follows a circular path based on its initial position and its offset from the path curve. Second, the sphere rotates about the center point of the circle as if there were an invisible rod connecting the center of the circle to the center of the sphere. To contrast this rotation, double click the sphere object to open the property window, go to the Map tab and uncheck the Rotation attribute, then replay the animation.

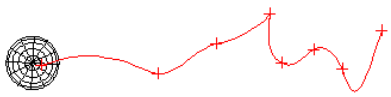
Object on an open path

Tutorial level: Medium

Tutorial project: 'tutorprojects\animation\paths\morphingPath'

Making an object follow a path defined by an open curve is essentially the same as using a closed curve. Once again, the initial positions of the two objects, in relation to each other, determines the actual movement (that is, if you are using the Translate option).

Set up a sphere and a curve as shown in the image. Select the sphere, then the curve and use Lattice Mapping/Path with the default options. Then play the animation.



A sphere and an open NURBS curve as the path

Now edit the NURBS curve, maybe straightening out the dip near the middle, and then play the animation. Try just moving points first, then edit the curve by also removing points. Next, add some new points to the curve, and reshape it again. Note that when you make these changes, the sphere speeds up and slows down as it traverses the path. This is based on the spacing of the curve points: the greater the distance between points, the faster the object moves along that line segment. No matter what type of editing you perform on the line, the object remains correctly attached to the path.

Now try editing the path curve while in animation recording mode. Select the curve and enter Edit mode. Set the time slider to frame 0 and activate animation recording. Move to frame 30 and point edit the curve. Move to frame

60 and adjust the curve some more. Continue editing the curve at frame 90, then deactivate animation recording. You now have a sphere that moves along a morphing path.

Unattaching an object from a path

As you may guess at this point there are quite a number of ways, based on the relative position between the object and the path, that you can set up a path animation. So what if you attach an object to a path, and you do not like the result? You can easily *unattach* an object from a path with the Unmap tool.

Select the object and then the path. Then, from the Lattice Mapping tab, click the Unmap button. The object and the path are no longer attached. You can now modify the positions of either object and connect the two again.



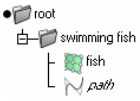
Note

You can also access the Unmap function from the main menu, Tools/Lattice Mapping/Unmap Objects.

A swimming fish

This example demonstrates how to create swimming fishes, wriggling snakes and other effects where the motion path should not only transform the object to be animated but also deform it.

1. Create a fish by using NURBS or subdivision tools. The only requirement for the fish is that it contains a sufficient number of points so that the curve can deform it.



2. Create a NURBS curve representing the path.

3. Select the 'fish' and the 'path' (in this order). Then click the Path tool. This time set the Target option to Points and click Accept.



Select Target=Points



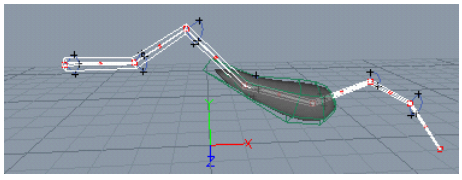
A freeform fish and a NURBS curve

Play the animation and the fish wriggles along the path. Keep this project open for use in the next section.

Reconnecting an object to a different path

Tutorial level: Medium

Tutorial project: 'tutorprojects\animation\paths\swimmingFish'



The fish moving along a skeleton

The Lattice Mapping/Reconnect tool allows you to exchange one path for another.

Using the swimming fish project from the previous section, add a skeleton object to use as the path for the swimming fish. Select the fish and then the skeleton, and then activate the Reconnect tool. Click Accept and the fish jumps to the skeleton. Move the time slider and the fish moves along the skeleton.

Example project: 'tutorprojects/animation/paths/fishOnASkeleton'

Mapping specific points to a path

Tutorial level: Medium

Tutorial project: 'tutorprojects\animation\paths\ik_path'

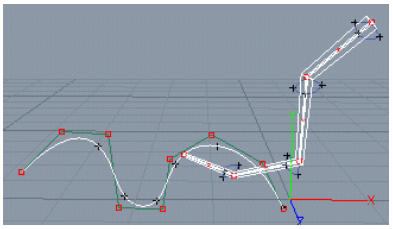
So far we have dealt with mapping entire objects to paths, but what if you only want a part of an object to move along a path, for instance, the end joint of an arm. For this you set the Path tool to use the Selected Points value for the Target option.

For example, you can use a NURBS curve to control the path of a joint of a skeleton:

1. Create a skeleton and a NURBS curve defining a path for a joint of the skeleton.

2. Multi select the skeleton and the path (in this order) and enter the Edit mode. Now, select the end point of the skeleton by clicking it.

3. Activate the Path tool and set Target to Selected Points. Click Accept.



The last joint of a skeleton attached to a NURBS curve using the path tool and the Selected Points option

You have now attached the joint to the path. Play with the time slider and see how the joint follows the path. Note that the skeleton uses inverse kinematics to follow the path, which is what you might have already expected.

Timing

In the previous tutorials we have used two key frames to set up path animations. In the first frame the object is at the start of the curve and in the last frame the object is located at the tail of the curve. The intermediate positions are interpolated for us by the animation system.

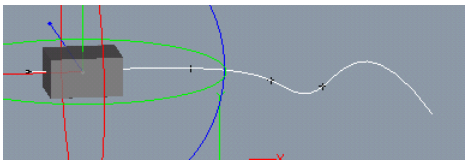
The speed at which the object travels between the two ends of the path depends on the properties of the path. Depending on the point density and parametrization of the path, the object may accelerate and decelerate while it moves along the path.

You can control the default speed and timing very easily by adding new key frames. This tutorial explains how to do that.

Tutorial project: 'tutorprojects\animation\paths\MovingAndRotatingCube'

1. Set the frame count to 200. Create an analytic cube and a NURBS curve to define a motion for it.

2. Select the cube and then the NURBS path. Apply the pull down menu Animation/Path/Move. Play the animation to verify that the cube travels to the end of the path during the playback.



A cube follows a path

Let's change the motion so that the cube reaches the end point of the path at the frame 100 and then travels back to the start of the path.

3. Open the choreography window (pull down menu Animation/Choreographs..). Go to the Properties tab. Open the sub structure of the 'cube:init0' choreography. Open also the structure of the path choreography. Select the attribute

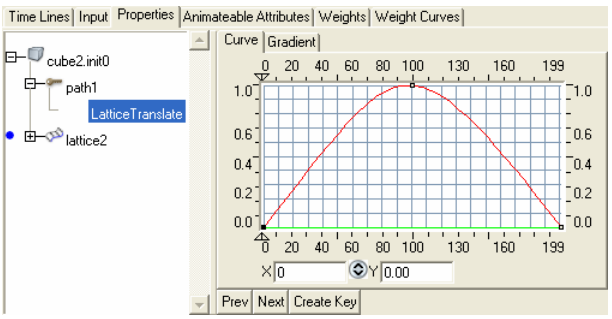
LatticeTranslate. The curve graph at the right side of the choreography window now shows a linearly ascending red curve. The green and blue curves have a constant zero value.

4. Click in the middle of the red curve and move the inserted point up to level y=1.0.

5. Move the end point of the red curve down to zero level.

Play the animation. The cube travels to the end of the path, slowly decelerates and then travels back. Deceleration at the end point is a result of the round curve shape. If you change the curve type to polyline, direction changes without deceleration.

Note that the red curve defines the position along the path. You can also create some key frames to the green and blue curves to temporarily displace the cube from the path.



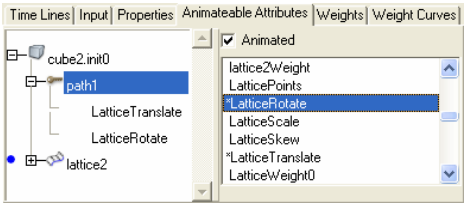
A lattice translation curve, which makes the cube to return to the start point of its path

In addition to translation, you can naturally control how the cube rotates in the path object's space. For example, to make the cube rotate while it moves on the curve:

6. Go to the Animateable Attributes tab on the choreography window. Select the path choreography.

7. Find and select LatticeRotate from the attribute list. Check Animated option for it.

8. Go to the Properties tab. Select the new LatticeRotate attribute under the path choreography. The curve graph shows no curve at this point. Select Create predefined curve/45 degree curve from the popup menu of the curve graph.



Make Lattice Rotation animated

9. Select Set Minimum and maximum values from the popup menu. Set Max Y = 360 degrees, check rescale and hit OK.

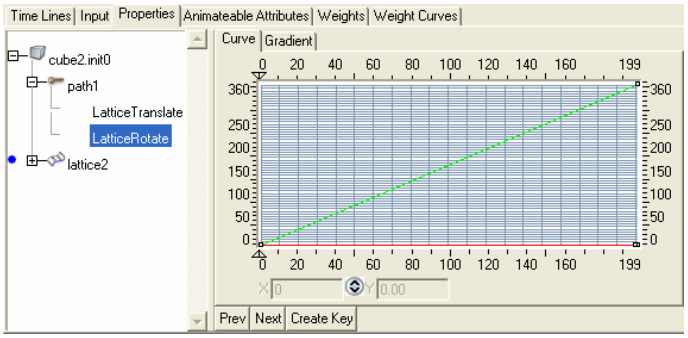
10. Move the end point of red curve and blue curve down to zero level. The green curve will grow from zero to 360 degrees, meaning that the cube will make exactly one full revolution around the path during its travel.



Tip

Use curve editor's Show Tools / Red, Green and Blue toggles to show and hide overlapping curves to make editing easier.

Play the animation. The cube moves and rotates in the parameter space of the curve.



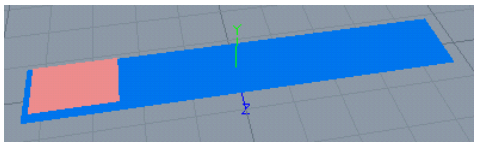
Lattice Rotation curves

Deformation paths

Let's assume we want to create a 'flying carpet' animation. This can be done by using a NURBS mesh as a path.

Tutorial level: Medium

Tutorial project: 'tutorprojects\animation\paths\meshPath'



A red 'carpet' mesh on a blue 'path' mesh.

1. Create a carpet (say a NURBS rectangle consisting of 5 x 5 control points). Open the properties window, go to Col tab and change the color of the mesh to a bright red. Rename the mesh as 'carpet'.

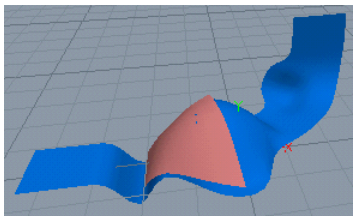
2. Create a bigger NURBS rectangle consisting of 20 x 5 control points. Open properties, Col tab and change its color to blue. This mesh represents the path of the carpet. We want the carpet to move on the surface of this mesh.

3. Select the 'carpet' and 'path' meshes in that order, activate the Lattice mapping/Path tool, set the Target option to Point and click Accept.

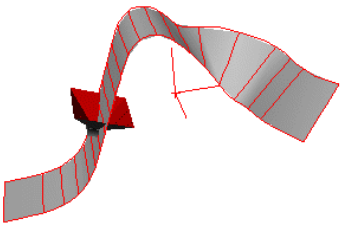
If you now move the time slider, you can see how the carpet moves along the blue mesh.

4. Modify the path mesh so that it contains bumps here and there.

Play the animation to see how the carpet moves along the curved path mesh.



The carpet follows a curved path mesh



Camera following a NURBS stripe



Note

A NURBS mesh as a camera path allows very good banking control.

Tutorial project: 'tutorprojects\animation\paths\CameraMovingAlongANurbsStripe'

Mapping objects to curved paths

Tutorial level: Medium

Tutorial project: 'tutorprojects\animation\paths\MapToCurvedPath'

Let's assume we have a very curved surface which we need to use as a motion path for an object. For example, we need to move a mesh on the surface of a NURBS head.

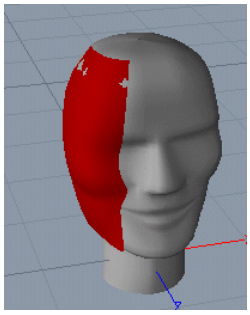
There are two problems. First, we need to deform the mesh to match the surface of the head at a certain area. If the head is very curved, this is not a trivial job. We could use collision detection or some other tools but if we have a very curved object containing tentacles and holes, matching the mesh on the path mesh can be very time consuming job.

The second problem is that lattice mapping to very curved lattice objects is not always reliable and points can get mapped to wrong lattice coordinates.

The Reconnect tool solves this problem. You can first map the object to a planar mesh to define appropriate lattice coordinates. You can even animate the object on that well behaving planar lattice object. Then, use the Reconnect tool to swap the path object to the final curved surface.

Let's see how this works.

1. Create a NURBS mesh consisting of 12x12 control points and an analytic rectangle on which we will animate the mesh. Then map the NURBS mesh to the rectangle exactly the same way as we mapped the NURBS carpet to the path mesh in the previous example.



A red mesh moving on the surface of a head

If you now play the animation, the mesh moves along the analytic rectangle.

2. Activate animation recording mode. Modify the mesh at suitable frames using move, rotate, etc. tools to create some key frames. Turn animation recording off.

3. Load in the actual object you want to be used as a motion path (for example, 'models\nurbs\simple_head'). If you use File/Load, remember to uncheck Replace, otherwise you will loose the project!

4. Select the mesh and the NURBS head and activate the Reconnect tool. Click Accept and the mesh jumps on the surface of the NURBS mesh!

Now that the mesh follows the shape of the NURBS head precisely, you can add some further key frames, fine tune motions, etc. without any problems.

Multiple motion paths

Tutorial level: Medium

Example project: 'tutorprojects\animation\paths\multiplePaths'

The lattice mapping system allows you to map an object to multiple lattice objects. For example, you can control the head of a mesh using one skeleton and its tail using another skeleton. Similarly, you can animate an object using several paths.

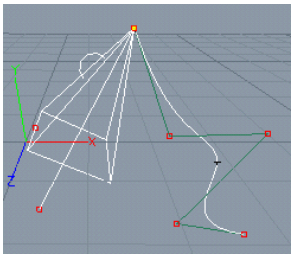
As an example, we will define an animation where the camera position follows a smooth path, but the aim point follows another polygonal path:

1. Create a camera and a smooth NURBS curve representing a motion path for the camera position.

2. This time we don't map the entire camera on the path but its position handle! So, enter the Edit mode and select the position handle of the camera. Multi select the camera and the NURBS path (again in this order).

3. Activate the Path tool and set Target to Selected Points. Then click Accept.

We have now mapped one point of the camera object to a motion path.



Map the camera position handle to a NURBS curve

4. Create a polygonal NURBS curve representing a motion path for the aim point and map the aim point handle to it exactly the same way as you mapped the camera position.

That's all. Play the animation and see how the aim point moves along one path while the position of the camera follows another path.

Animating Materials

To animate one or more materials, you follow exactly the same rules used to animate any other object type in Realsoft 3D. This tutorial presents a number of example projects that demonstrate common material animation cases. We'll start with a simple case, and move on to more interesting ones at the end of this tutorial.

Animating a texture map

Tutorial level: Medium

Example project: 'tutorprojects\animation\materials\movingTexture'

One way to animate a texture is to make the entire texture map change in some way. In general, this would do nothing to the characteristics of the texture itself - that is, attributes of the texture - but it would still produce certain useful effects.

To begin, imagine you have a scene that requires a painting on a wall, and you want to create an animation where the painting moves across the wall (maybe this is a haunted house). To do this, you do not need to animate actual material attributes. Realsoft 3D uses geometric *mapping objects* to assign materials to actual geometric objects. This type of system allows you to animate all *geometric* material properties, such as position, scale, and rotation, as easily as animating these same properties of any "regular" geometric object using the animation recording mode or choreographs.

To demonstrate this technique, start by creating an analytic cube as a wall (a front view is probably best). Now do the following:

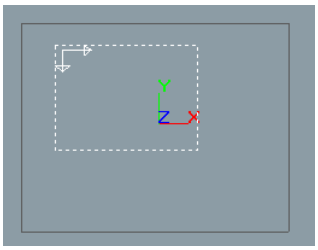
1. Go to the Select window's Materials tab. Right click, and from the popup select New/From Template.

2. The Load dialog opens up. Open the materials folder. Open the Image Maps folder and then select the Color Map file. This creates a new material called 'Color Map'. Find this material in the list, select it, and then rename it to something more suitable, such as Lake_painting, for example.

3. Double click the Lake_painting material to open the property window. Set the value of the File Name field to 'lake,' found in the Realsoft 3D 'textures' directory. Leave all the checkbox attributes unchecked. Close the property window.

4. Now highlight the Lake_painting material, then right-click and select Map/Parallel from the popup. Draw a rectangle over the wall object.

5. Ray trace render the view.



Texture map drawn inside the wall object



A material parallel mapped on the wall

6. Set the time slider to the first frame, and activate animation recording. If the "painting" (that is, the texture map object) is not placed where you want it, move it to the starting position. Go to the last frame of the animation, and move the painting to a different location, for instance, the other side of the wall. Turn off animation recording and play the animation. The parallel mapping object representing the position of the painting moves across the wall.

Essentially, that is all there is to it. You can animate a texture map object in many other ways, such as scale, rotation, skew, etc., just like any other geometric object.

You can create many interesting and useful animation effects by animating mapping objects this way. For example, you can animate the Invisible in photorealistic rendering attribute to enable and disable rendering of a certain material. In this way you can create "wink out" effects, or make a transparent portion of an object become opaque. You can even apply gravity and other simulation effects to mapping objects.

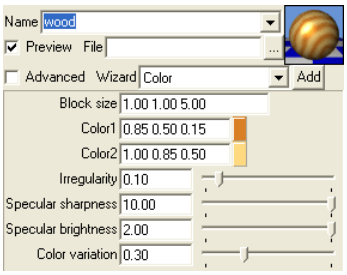
Animating texture attributes

Tutorial level: Medium

Example project: 'tutorprojects\animation\materials\animatedWood'

In the previous section we looked at animating a texture map object, but this is not the only way to animate textures. You can also animate any of the attributes of a texture.

For example, let's create a wooden tabletop and animate several characteristics of the wood texture. To do this, you animate the actual material object, not the mapping object.



Wood in the first animation frame

1. In a top view, create an analytic cube, and size it so it looks like a tabletop. Leave the cube selected.

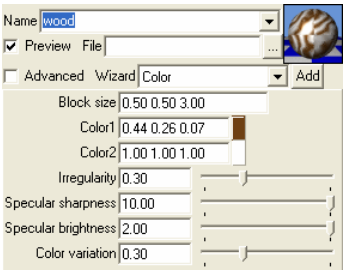
2. Go to the Materials tab of the Select window. Find the standard Wood material and double click its icon to open the property window.

We will use the default values of the wood material for its initial look, so do not change the material in the first frame.

3. Now move the time slider to frame 60 and activate animation recording. Set the Color 1 attribute to a darker brown, and the Irregularity attribute to 0.3.

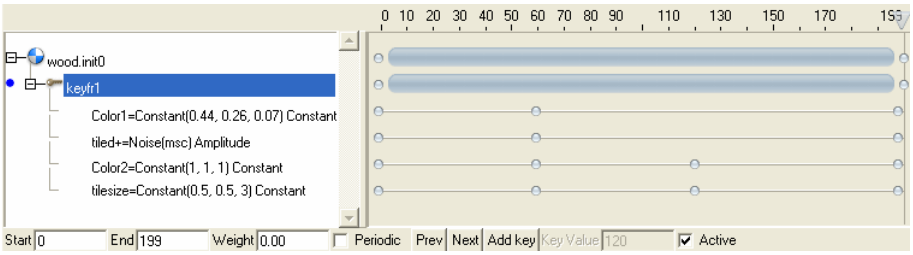
4. Next, move to frame 120 and set the Color 2 attribute to an off-white, and the Block Size attribute to (0.5 0.5 3). Finally, reset animation recording to off.

5. Reset the animation to the first frame, and drag the time slider (do not press play!) through to frame 120 while watching the attribute values change in the property window. Do you notice something? Did you want the Color 2 and Block Size attributes to start changing at the first frame or at frame 60?



Wood in the last animation frame

6. If you wanted the animation of the Color 2 and Block Size attributes to begin at frame 60, this is easy to correct. Open the choreography window (from the pull down menu Windows/Choreography Window) and select the Time Lines tab. Open the wood.init0 entry and the keyfr1 entry under it. Now you see a list of the animated attributes of the choreograph. The last two elements should be Color2= ..... and tilesize=..... Move the first key of these two attributes to frame 60. Now go back, make sure the properties window is open, and drag the time slider again. Watch the attribute values for the animated wood material change over time.



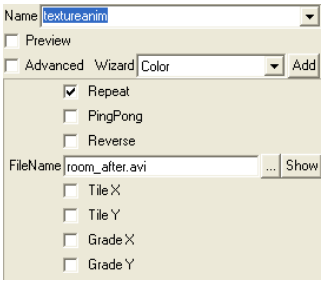
Resetting the keys to frame 60 for the Color2 and tilesize attributes

7. To finish this tutorial, close the choreography window and drag & drop the wood material from the Select window to the view window. This maps the animated wood to the tabletop.

Using animation files as textures

Another method of creating an animated texture is to use an animation file, such as an AVI, as a texture map. This allows you to use, for instance, a video sequence, or a series of images created in another program and rendered as an FLC or AVI file, as a texture.

To do this, go to the Select window's Materials tab, right-click and from the popup select New/From Template. From the Load Selected Objects dialog, open the AnimatedImageMaps folder, and then select the TextureAnim file. This places a new material called 'textureanim' in the materials list. Rename this material to something useful for your project, and then double-click this material to open the property window.



Property window and attributes for a texture animation material

Enter the name of a file in the File Name field (this should be one of the supported animation file types). Now map this material to an object.

As you can see from the attributes, you have additional options for controlling how the animation file is positioned and played. For instance, you can keep repeating the animation file from beginning to end, you can ping-pong it (beginning to end, end to beginning, and so on), or you can have it play in reverse.

Controlling the timing of an animated texture

The technique we just described works fine if the AVI (or other type) file plays the images exactly at the speed you want. But what then if that is not the case? What if you need an animated image to change first slowly and then speed up? Obviously, you can not achieve this type of control with the basic Ping Pong, Repeat, or Reverse options. The Current Frame attribute is provided for situations like this.

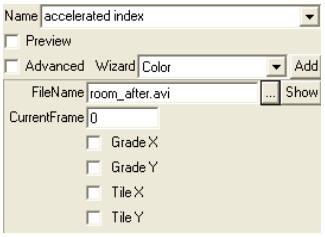
Assume you have an AVI animation file consisting of 10 frames, and that you want to control the speed of the AVI file in special way. For the first 60 frames of the animation, you only want to use the first 3 frames of the animated texture, then during the last 40 frames you want to use the remaining 7 frames of the animated texture.

You set up this kind of an animation by key framing the texture frame index (the Current Frame attribute). To do this, you can use the supplied template material in the AnimatedImageMaps folder.

1. If you have, or can easily make, a 10 frame AVI, do so and use it in this exercise (see the manual chapter about File Rendering for details).

2. Go to the Select window's Materials tab, and choose New/From Template from the popup. Load the 'AnimatedImageMaps/Keyframe Texture' material template.

3. The materials list should now contain an entry called 'keyframe texture'. Rename this material template to Accelerated Index, or something else meaningful to you, then double click it to open the property window.



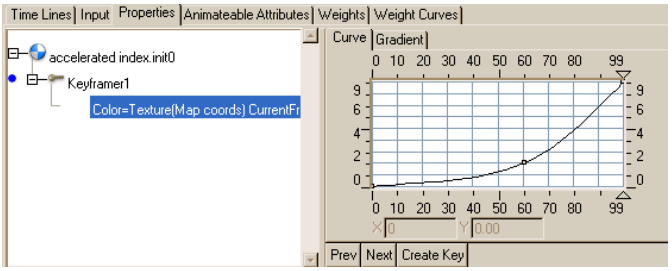
Use an .avi file as a texture

4. In the File Name field, enter the name of your AVI file.

5. Now activate animation recording and set the time slider to the first frame. The Current Frame field is, by default, zero, which is the correct start value (The frame index for AVI files typically begins at 0, so a 10 frame AVI uses index values of 0-9). Move the time slider to frame 60 and set the Current Frame field to 2. Now move to frame 99 and set the Current Frame field to 9. Turn off animation recording.

That's all there is to it: you have now keyframed the *frame index* of your AVI file.

6. Last, open the choreography window to the Properties tab. Select the CurrentFrame attribute from the choreography list to view the animation curve.



The animation curve controlling the frame number of the AVI file as a function of time

Animating Particle Effects

Just like you can attach a material object to a geometric object to define optical and other properties, you can attach 'post effect' objects to particles to render the otherwise primitive particles as more interesting objects.

If you have read the previous animation tutorials, you should already know how to animate geometric properties of any kind of objects, including particles. In this tutorial we will examine how to animate the actual particle effects.

Tutorial level: Beginner

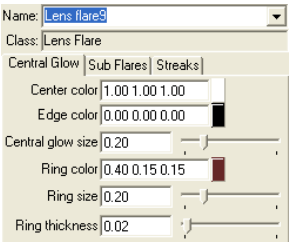
Example project: 'tutorprojects\animation\particles\animatedLensFlares'

Let us create some 1D particles, which are rendered using an animated lens flare effect. You can create spinning lens flares, star filters, and many other fancy effects this way.

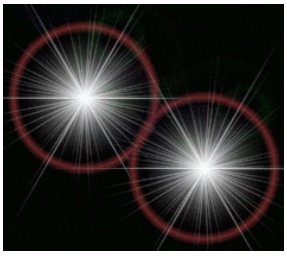
1. Use the particle tool to create some particles. If you haven't used the particle tool yet, the workflow goes as follows: Go to the Particle tab in the toolbar. Activate the particle tool, left click and drag the mouse across the view and select Accept.

2. Select the Lens flare effect from the Select window's Post Particle Effects tab and open the property window.

3. Move the time slider to the last frame. As usual, activate the animation recording mode and modify various lens flare attributes to create key frames for them.



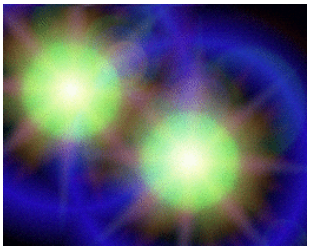
Lens flare properties



The initial state of lens flares

4. Reset animation recording and play with the animation slider to verify how the modified lens flare attributes change with time.

5. Now, map the lens flare object to the created particle object (you can do this by selecting the particle object and then drag and dropping the lens flare effect into the view window).



The final state of flares

Animating Image Effects

Realsoft 3D includes powerful tools for performing post-processing effects on rendered images. The dynamic architecture of the rendering system makes it possible for the rendering engine to output, or make available, any data it computes during the rendering stage. In addition to predefined rendering channels such as Color, Alpha, and Transparency, you can create and define any number of custom channels and use the VSL material system to process them. Thus, it is possible to achieve virtually any post processing effect you can imagine.

Global fog

Tutorial level: Medium

Example project: 'tutorprojects\animation\post images\animatedFog'

The fog post effect uses the depth information from the rendering engine and its own attributes such as fog color, saturation, and minimum and maximum distances to compute a global fog.

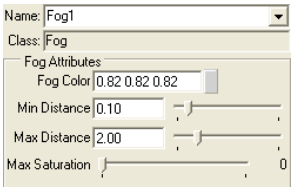
You can animate any of the fog attributes. For example, to create an animation where the saturation of the fog drops from 100% to 0%, do the following:

1. Create a fog object in the select window's Post Image Effects tab by selecting New/Fog. Double click on the new object to open the property window.

2. Turn on animation recording.

3. Move the time slider to the last frame and set the Max Saturation field to 0%. Turn off animation recording.

4. You now have two key frames for the Max Saturation attribute. Move the time slider to see how the fog saturation decreases over time.



Modify the Max Saturation attribute in animation recording mode

If you want to see how the fog looks when rendered, create some geometric objects at different distances in a view window. Next, drag & drop the animated fog object from the select window into the view window. Render the view (with Ray Trace rendering) at several different frames along the timeline to see the effect of decreasing fog saturation.

Rotations

A problem that has existed since the birth of computer graphics is one of interpolating the intermediate frames from given key frames that control rotation. For example, if you have three key frames which represent camera orientations of *front*, *top*, and *back*, you'll typically get a sudden jump as soon as the camera reaches or even passes close to the *top* position. This problem is caused by the fact that the polar coordinate system has singularities at the north and south poles. Rotation curves, which look very smooth and beautiful in the curve editor, are totally incapable of producing nice, smooth rotations in 3D space.

Realsoft 3D solves this problem by using advanced *Quaternion* interpolation for rotations.

The orientation of an object is controlled through three rotation angles called *heading*, *pitching*, and *banking*. Defining any orientation for an object through these three angles is easy and intuitive.

Let's see how well this quaternion system can solve the following well-known rotation problem:

Tutorial level: Medium

Example project: 'animation\rotation\rotatingCube'

1. Switch to the front view and create an analytic cube. Activate animation recording.

2. In frame 20, rotate the cube 90 degrees in either direction. To get accurate 90 degree rotation, hold down the Shift key while dragging the rotation handle. This activates the angle grid.

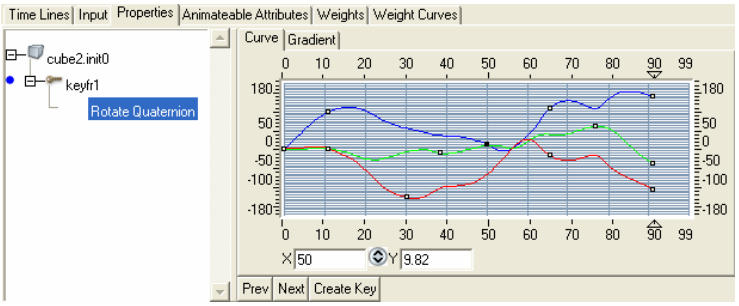
3. Switch to the side view, move to frame 40 and rotate the object 90 degrees.

4. Switch to the top view, move to frame 60 and rotate the object 90 degrees.

Now, reset animation recording and play the animation. As you can see the cube rotates nicely through all defined key frames exactly the way you would expect.

So, it looks like this quaternion interpolation system can handle rotations intuitively. However, there is one minor drawback with this system.

5. Open the choreography window, click the Poperties tab and select the Rotate Quaternion attribute. The three curves that define the heading, pitching and banking angles look very wild and may even contain sudden jumps.



Heading, pitching and banking curves generated by Quaternion interpolation

The explanation for these crazy curves is that the heading, pitching and banking angles simply must jump around to produce a natural looking rotation in 3D space! These strange curves really represent the rotation angles. The problem is that it is difficult to edit these curves in a sensible way.

Euler angles

The quaternion interpolation system produces superior rotations, and is therefore, the default method used. To use the more standard Euler angle interpolation method, check the Rotate in HPB check box in the Gen tab of the property window of your object.



Note

You have to set this option before you animate your object.

Rotating in lattice space

Another way of controlling rotations is to define an axis, and then an angle, which defines how much the object is rotated about the axis. You can do this using the lattice mapping system. The basic idea is the same as for path animations, the difference being that the object is now rotated in lattice space instead of translated.

To illustrate this, we'll create a cube that rotates around a NURBS curve.

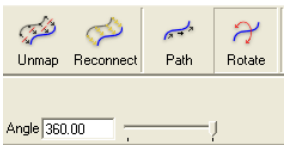
Tutorial level: Medium

Example project: 'tutorprojects\animation\rotation\rotateInLatticeSpace'

1. First create a cube, and then a NURBS curve which defines the rotation axis.

2. Select the cube and the axis (in this order) and activate Lattice Mapping/Rotate from the toolbar. Specify a rotation angle to the Angle field on the control bar and click Accept.

3. Play the animation and the cube rotates about the axis object.



Set up a rotation in lattice space

One benefit of using this method is that you can now animate the axis, which is a normal geometric object. For example:

4. Activate animation recording and move the time slider to the last frame. Rotate the axis 30 degrees (any direction is fine), and move the axis to a new position.

5. Play the animation. The cube rotates about a moving and rotating axis object.

As an alternative to using the Lattice Mapping/Rotate tool, you can also use the Lattice Mapping/Map tool and then animate the Lattice Rotation attribute manually in the choreography editor.

Texture Morphing

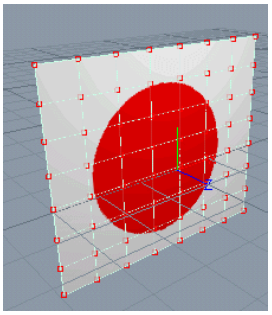
Texture morphing is the process of animating a texture, or an object's material, over time. For example, creating a scene that includes a waterfall, or any type of moving water, would use texture morphing. You have most likely seen movies in which the head of a person suddenly transforms into the head of a terrible monster, or vice versa depending on the movie. Such effects, and more, can be created using texture morphing.

Morphing the shape of a texture

Tutorial level: Medium

Example project: 'tutorprojects/animation/textures/textureMorphing'

One way to morph a texture is to change its shape. Realsoft 3D allows you to edit texture control points, just the same way you would edit geometry control points. Since many useful and interesting effects can be achieved this way, let's explore how this is done. This tutorial involves creating a texture that is shaped as a circle and changing it into a rectangle.



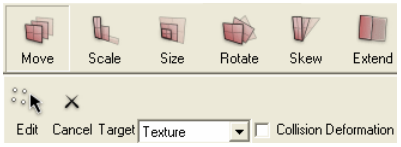
A filled circle painted on a NURBS mesh

1. Create an 8 x 8 NURBS rectangle. This is adequate for this exercise; however, you should note that the more control points the mesh has, the better control you have over texture deformations.

2. Select the mesh. The control bar now shows you the NURBS-specific tools. Click the 'Define UV' tool. This creates custom texture coordinates for the mesh that can be edited and animated.

3. Click the Materials tab in the toolbar. Click the UVimage tool and click Accept. The default settings are OK. This creates a blank texture for the mesh. Turn Shaded OpenGL on if it is not already enabled. On the Materials tab of toolbar, click the Circle tool. In the tool control bar select the Filled option and change the color to a bright red.

You can also use your favorite painting program to create the image, and then attach it to the object using the property window's 'Col' tab.

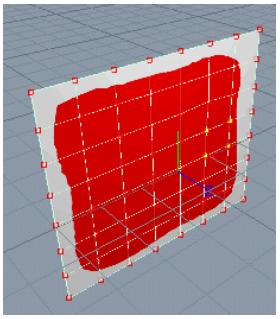


Activate the move tool and set 'Target' to 'Texture'

The next step is to deform the texture coordinates so that the circle turns to a rectangle.

4. Set animation recording on and move the time slider to the last frame.

5. Select the mesh and enter the edit mode. Select the control points whose texture coordinates you want to edit and activate the Move tool. Set the 'Target' option to the value 'Texture'.



Texture coordinates deformed.

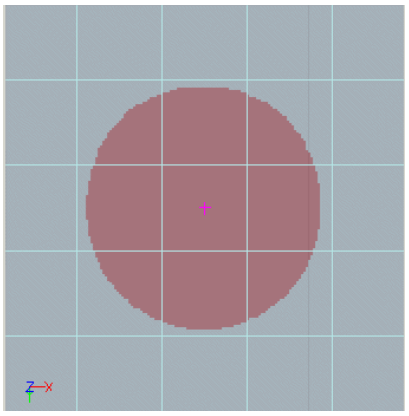
6. Enter two points through the view window to define the movement of the control points. When moving the points, make sure you click the mouse over the actual mesh. Otherwise, the program will not be able to compute how much the texture should be moved on the surface.

7. Continue selecting points and moving them, if necessary, until you have reshaped the texture into a rectangle.

8.When you have finished modifying the necessary texture control points, turn off animation recording and play the animation.

Using the UV Editor

The previous example showed how to directly edit texture object coordinates, and this technique works well for many situations. However, you also can view and edit texture coordinates in the UV Editor. You may find the UV Editor more useful for certain types of texture deformations or adjustments, or it may be that the editor is better suited to your way of viewing the texture editing process. Either way, here is how it is done.



The UV Editor

To use the UV editor to define texture deformations:

1. Open the editor by selecting 'Windows/UV Editor' from the main menu. When you select the mesh, the UV Editor shows you the texture as a backdrop image and renders the texture control points over it.

2. Move the time slider to a desired frame, set animation recording on and edit the control points.

While editing texture coordinates through the UV Editor, view windows show you the result in real time. If OpenGL acceleration hardware is not installed, rendering of large textures may slow down the editing. To avoid this, use turn shaded OpenGL rendering off for the view window (the 4th button from the bottom of the view control bar).

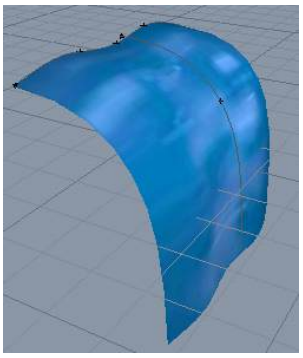
If you now move the time slider, you can see how the texture coordinates in the UV editor change with time.

Creating a waterfall

Tutorial level: Medium

Example project: 'tutorprojects\animation\texture\waterfall'

Let's move on to something more interesting. Next, we will create a simple waterfall animation where a texture that looks like falling water moves along a NURBS mesh.



A textured NURBS mesh representing a waterfall

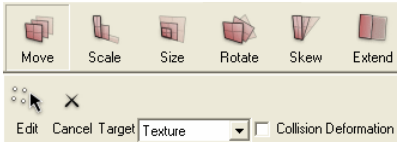
1. Create a textured NURBS mesh that represents a waterfall, see the image below. You can also use the sample project file for this tutorial. Use the property window's 'Col/Texture Map' field to attach the image directly to the surface.

2. Activate animation recording and move the time slider to the last frame.

3. Select the mesh and activate the Move tool. Set the 'Target' option to 'Texture' in the control bar.

4. Now enter two points over the NURBS mesh to define a translation for the texture downward along the mesh.

The texture animation is now ready. Turn off animation recording and move the time slider to see how the texture slides on the surface. In OpenGL rendering mode you can see how the texture moves in real time.



Activate the move tool and set 'Target' to 'Texture'

You can also animate the position of the texture through the property window. In the 'Col' tab you can see four texture transformation fields which allow you to translate, scale, rotate and skew the texture using a numerical interface.

Rendering Animations

Depending on your output requirements, there are many possible ways to render a scene. Realsoft 3D supports a variety of rendering options, including field rendering and motion blur, in addition to the various types of rendering that can be achieved, such as ray tracing, scanline, and outline.

Field rendering

Field rendering is activated from the Animation Settings window by clicking the Field Rendering check box in the Field R tab.

When the 'Field Rendering' option is checked, you get a field controller in the animation window that controls the current field.



Click the check box in the Animation Settings window

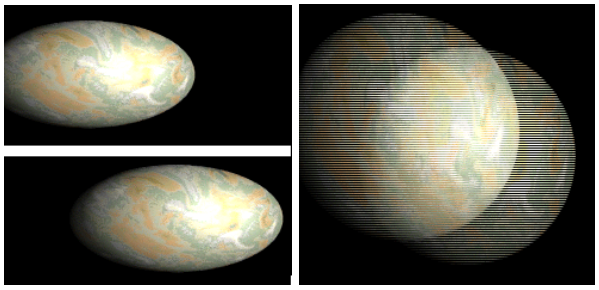
In some cases you may have to control how the fields are rendered and how they are interlaced with each other. The animation settings window contains two controls for this.



Field rendering activated, the Field control appears in the animation window

The first option, 'Shift latter field up', defines whether the latter samples are taken above or below the samples in the former field.

The second check box can be used for defining how the samples are combined.



Even and odd fields on the left. The combined frame on the right.

Motion blur

Motion blur is a rendering effect that makes objects appear to be moving rapidly through the camera's field of view. If you have ever taken a photograph of a moving car at highway speed, you have probably seen the effect.

Animation Controls

Motion Blur is activated from the animation settings window by clicking the check box.

The 'Samples' field allows you to define the number of samples per frame (or field). The higher the value, the better the quality of the motion blur.



Motion blur activated

When motion blur is activated, a new control for samples is added to the animation window.



The Sample field shows the current motion sample

Using this field, you can go to and render any of the samples in the current frame.

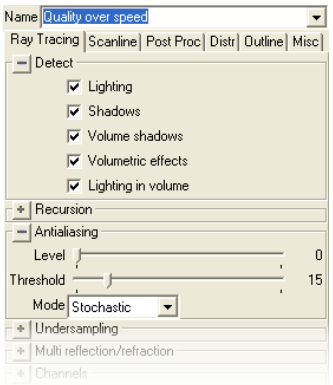
When both motion blur and field rendering are activated, each intermediate image is addressed as follows:

Minute/Second/Frame/Field/Sample

In other words, every field consists of one or more samples. Every frame consists of one or two fields.

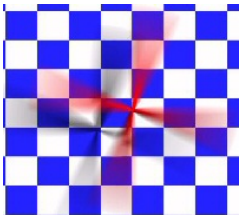
Rendering Settings

When rendering motion blur animations, it is best to use *Stochastic* anti-aliasing (defined in the rendering settings) with the antialiasing level set to its minimum value. Stochastic antialiasing operates over time and antialiases frames well even when using the lowest antialiasing level.



Use Stochastic anti-aliasing method with motion blur

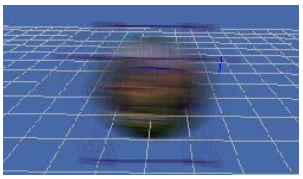
Typically 9 is a reasonable amount of motion samples when rendering animation. Because of the stochastic antialiasing method, the rendering time is, fortunately, not 9 times longer but much less depending on the scene.



A motion blurred image showing a red spinning fan. The fan casts blurred shadows on the floor.

Previewing motion blur using OpenGL

You can use OpenGL to preview field rendering and motion blur animations. To achieve that, make sure OpenGL is enabled on and activate the View Property Window / Render Tab/ Animation Preview / Combine Sub Frames option before starting the preview playback.



OpenGL rendered moving texture mapped sphere

Skeletons

Skeleton is a special purpose deformation lattice object. The idea behind skeletons is quite straightforward: a skeleton is a simple object which is connected to more complex objects. When the skeleton is modified, the attached objects are modified accordingly.

The fact that it is much easier to modify these skeletons than the actual (often quite complex) target objects makes skeletons a powerful animation tool. However, due to generality of the lattice mapping system of Realsoft 3D, skeletons can also be used in various other tasks.

In this tutorial we will demonstrate how to use skeletons to create a walking human body. We will first create a hierarchical human skeleton, then attach the actual model to the skeleton and finally animate the skeleton to take a couple of steps using the key framer.

Creating a skeleton

Tutorial level: Beginner/Medium

Example project: 'tutorprojects\animation\skeleton\body'.

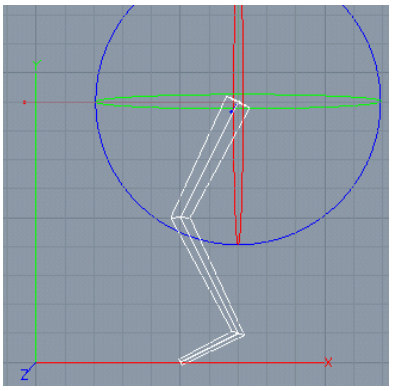
The Skeleton tool creates a skeleton object through the given joint positions. You can also define minimum and maximum constraint angles while drawing the skeleton. Each joint of a skeleton has six angle constraints, which limit rotations of the bones around the joint.

To create a skeleton representing a leg:

1. Activate the Skeleton tool from the Modifier tool tab.

2. Take a side view. Click the first point at the hip bone. Now, move the mouse towards the knee but don't fix the position for the knee yet. We will have to define proper constraints for the thighbone first. Rotate the mouse upwards so that it represents the knee in its uppermost position. Then move the mouse to the opposite direction so that it shows the position of the knee in its back position. Then move the mouse to a position between the two extreme angles and to a suitable distance (the length of a thighbone) and fix the second point (the knee) by clicking the mouse.

3. Then move the mouse towards the ankle and repeat the steps above in order to define constraints for the shinbone as well.



A skeleton representing a simple leg

4. Next define a bone from the ankle to the tip of the toes (we do not create toe bones this time) with appropriate constraints. Then select Accept from the pop-up menu to finish the skeleton creation.

Once created, skeletons can be modified in various ways.

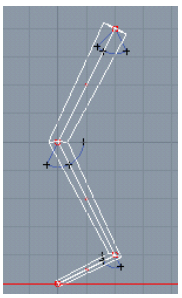
For example, to move a joint of the skeleton created:

1. Select the skeleton and enter the edit mode (as always, you can enter the edit mode either by pressing the space bar, selecting the 'Edit' compass menu, or clicking the 'Edit' tool in the tool control bar).

2. Drag a joint to move it.

While you move the mouse, not only the dragged joint but also all the other joints of the leg become modified. However, bone lengths remain unchanged. This method of modifying skeletons is called 'Inverse Kinematics'.

Inverse Kinematics is the default tool for the joints and therefore it is automatically started when you drag any joint of the skeleton.



Skeleton in the edit state. Drag a joint to apply inverse kinematics.

In addition to inverse kinematics, there are a number of other useful tools for editing skeletons. To apply 'Forward Kinematics' to a joint:

1. Select the desired joint (not the first one).

2. When a skeleton is selected, the tool control bar shows you various skeleton specific tools. Select the 'FK' tool and click the mouse twice in the view window to modify the selected joint angles.



FK tool

To modify the length of a bone:

1. Select a joint

2. Click the 'Length' tool in the tool control bar and enter two points through the view window to adjust the position of the selected joint along the previous bone.

Adding new joints

You can add new joints to a skeleton any time by using the Add Joint tool. For example, let's assume we need to add a new joint to the foot bone representing a toe of the leg:

1. Select the leg and activate the New Joint tool.



Add Joint tool

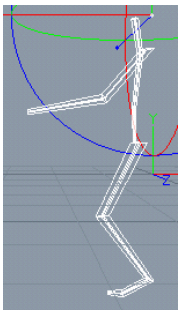
2. Drag the mouse over the skeleton. This outlines a small 'cross mark' which indicates the position of the new joint. When a desired position is found, release the mouse to create the new joint.

Creating hierarchical skeletons

A hierarchical skeleton is a skeleton, which contains one or more child skeletons. A good example of a hierarchical skeleton system is an arm and its five sub skeletons, the fingers. Another example is a spine and its two sub skeletons, the legs.

Next we will create a hierarchical skeleton roughly representing a human body.

1. Switch to the side view (so that you can see the leg from the side). Create three skeletons: a spine, an arm and a leg.



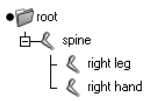
Side view: a spine and two child skeletons



Note

We only have to create one leg and one arm for the body!

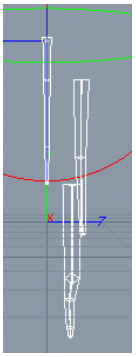
2. On the select window, drag & drop the arm and the leg skeleton inside the spine skeleton. You should also rename the skeletons so that you can recognize them later.



3. Switch to the front view and move the leg and the arm skeletons to their correct positions.

4. Select both the sub skeletons and click the Attach button in the tool control bar. This attaches the selected skeletons to the nearest joint of the parent (spine) skeleton.

Go back to the side view and try moving joints of the sub skeletons in the edit mode - the inverse kinematics affects the entire body.

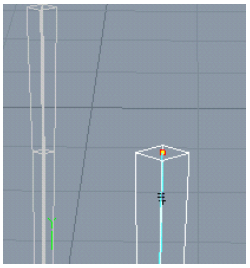


Front view: the leg and arm displaced from the spine

Editing angle constraints

If you switch to the front view and try to move one of the hand joints horizontally, you will notice that the entire skeleton moves as a rigid body. Both the arm and the leg refuse to rotate sideways. The constraint angles between the arm and the body do not allow rotations to the side.

There are two ways to edit the constraints. Either you drag the desired constraint angles through the view window, or you use the property window. For example, to modify the constraint of the root joint of the arm skeleton:



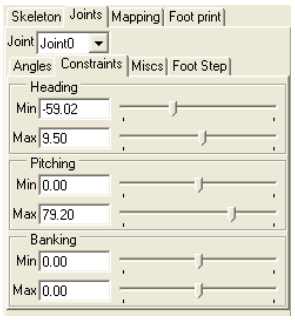
Select the root joint of the arm

1. Select the first joint of the arm skeleton by clicking it

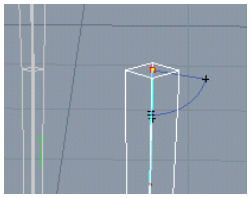
2. Open the property window. Select the 'Spec/Joints/Constraints' tab.

3. As you can see, the 'Pitching' constraint angles are zero. Increase the 'Max' value and decrease the 'Min' value.

While you move the slider, you can see how the 'pitching' handle of the skeleton moves in the view window accordingly. You can also edit the constraint angle by dragging the handle (the cross at the corner of the angle sector) directly through the view window.



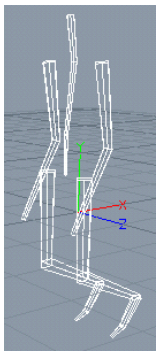
Modify the 'Pitching' angle to allow the hand to rotate sideways



The pitching constraint expanded

Edit also the constraints of the hip joint of the leg skeleton so that it allows appropriate rotations in both the heading and the pitching directions. The knee and elbow constraints are OK - they do not bend sideways.

So, we now have a body, which only has a left leg and arm. The right arm and leg can be created easily. Just select the root skeleton (spine) and apply the DupMir tool.



A human skeleton

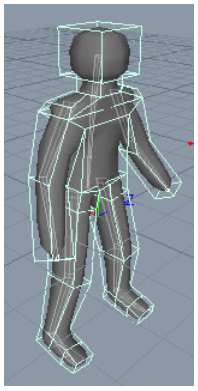


DupMir tool

This duplicates and mirrors all the sub skeletons of the spine. The skeleton is ready.

Attaching bodies to skeletons

The next step is to attach the skeleton to the actual body.



A body and a skeleton

Use the subdivision surface tool set to create a body, which roughly matches the skeleton. If you are familiar with the SDS tool set, you should be able to create a simple body for the skeleton quite rapidly. If you are too busy to model the body, you can load one from:

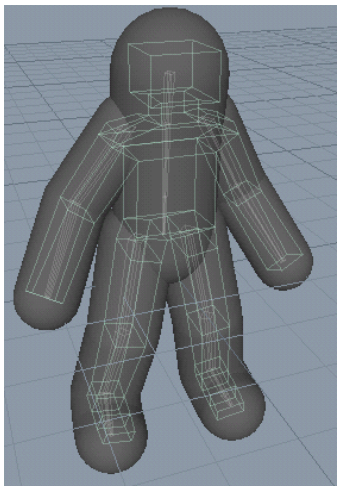
'tutorprojects\animation\skeleton\bodyAndSkeleton'

Adjust the skeleton so that it matches the body.

We will attach the body to the skeleton using the 'Multi Map' lattice mapping tool. This tool is capable of attaching a body to a hierarchical skeleton using weighted multi mapping. This means that a single point can be mapped to many skeletons with appropriate weights. The higher the weight, the stronger the skeleton in question affects the point. For example, a point between the start of two fingers can be mapped to both fingers with equal weights, which gives a realistic skin effect.

1. Select the spine and click the 'Map Volume' tool. This turns on visualization of so-called mapping radius property.

2. Click the 'Adj. Rad' tool and enter a point through the view window. If you now move the mouse up/down, you will be able to adjust the mapping volume radius so that those points you want to map to the spine are inside the mapping volume. Repeat the steps for the legs and arms.



Mapping volumes defined for the skeletons.



Note

Shaded OpenGL mode usually shows best what the appropriate mapping radius is. The wireframe of the SDS control polygon outside the mapping volume can be easily detected.

3. Once you have managed to define appropriate mapping volumes, just select the body and the root skeleton (spine) and click the 'Multi Map' lattice mapping tool. Your body is now attached to the skeleton and you can switch the mapping volumes off by selecting the skeletons and clicking the 'Map Volume' button on the tool control bar.



Note

The mapping volumes overlap where the legs and arms join to the body. Correspondingly, points near these joints get mapped to multiple skeletons!



Multi Map tool

To make sure all the points were attached to the skeleton, just modify the skeleton (drag a joint in the edit mode). The body should follow the skeleton.

Tutorial project: 'tutorprojects\animation\skeleton\attachedBody'



Note

If you defined too small mapping volumes, some points in the model may not get mapped to any skeleton. To fix the problem, select the point, then select the appropriate skeleton and use the standard lattice mapping Map tool with the 'Selected Points' option to attach the point to the skeleton manually.

If the mapping volumes are too large, some points may get mapped to multiple skeletons when they should not. You can detach a point from a skeleton by multi selecting the point and the skeleton and then applying the Unmap tool.

Animating skeletons

Animating skeletons using the key framer does not differ from animating other objects, such as spheres. Just modify the skeleton in the animation recording mode. For example, let's make the skeleton wave its hand.

1. Set animation recording on and jump to frame 20.

2. Switch to the side view and drag the end point of the right arm of the skeleton.

3. Jump to frame 40 and drag the end point of the right arm back to its original position.

4. Switch the animation recording off and play the animation. Tutorial Project: 'tutorprojects\animation\skeleton\say-Hi'

We have now gone through a basic workflow for creating and animating characters. To keep the example simple, we just attached two sub skeletons to the body to animate legs and arms. However, you can create any number of sub skeletons for controlling the body. Lips, eyes and other parts in your model can be animated this way. In the following chapters, we will go through a number of other useful tools for animating skeletons.

Key framing angles and joint positions

When you modify a skeleton in animation recording mode, the animation system animates the skeleton by key framing the bone angles, regardless of the tool you use to modify the skeleton.

The problem with angle based key framing is that it tends to create 'slipping feet' problems. It is a difficult job to key frame the bone angles so that their motions match exactly the translation of the body.

Instead of key framing bone angles, you can record joint positions. In this case, the system applies inverse kinematics to move the joints through the recorded key positions.

To do this, set the 'Preferred Anim. Method' field in the property window's 'Spec/Skeleton' tab to 'Inverse Kinematics' and then modify the skeleton in the animation recording mode as usual.



Note

When using joint position key framing, key frames are created only for the selected joints (those joints you modify directly with the mouse)!

Mixing angle and position keys

Angle and position key framing can also be mixed. For example, you can first key frame the skeleton using 'Angles'. Then you can define an exact path for a certain joint by key framing its position. This way the system first sets the skeleton to a 'nearly' correct position using angles and then inverse kinematics fine-tunes the position.

To achieve this, use Angles as the preferred animation technique and apply IK Target tool to the desired point. This tells the animation system that the selected joint should be key framed using inverse kinematics and key framed positions.



IK Target tool

Alternatively, you can use Inverse Kinematics as the preferred animation method and use the Angle tool to activate angle key framing for a desired joint.

Anchoring joints

The 'Anchor' tool fixes the joint to object space so that IK cannot move it. This tool is useful when using inverse kinematics to control the joint positions. For example, you can create a moving compass needle and use IK to make

it point to the north pole. You have to anchor the origin of the needle so that IK won't pull the entire needle to the North Pole. Let's create a walking leg to demonstrate the 'Anchor' and 'I.K Target' tools.

1. Create a simple skeleton consisting of four joints, representing a leg.

2. Set animation recording on, jump to the last frame and move the entire skeleton horizontally by dragging its 'x' axis. We have now created a simple 'moving leg' animation. Reset animation recording.

3. Enter the edit mode and apply the 'Anchor' tool for the root joint.

4. Select the last joint of the leg and apply the 'IK Target' tool.

Play the animation.

The root joint is now anchored so it is forced to move according to the translation we applied to the entire skeleton. However, the last joint acts as an 'IK Target'. Because we didn't key frame the position of the last joint, it attempts to remain in its original position.

Tutorial project: 'tutorprojects\animation\skeleton\iktarget'

IK stoppers

When you apply inverse kinematics to a joint (say, you pull a finger of a robot), the effect propagates through the entire skeleton and bends and twists it according to joint frictions and constraints.

Let's assume you want to pull the finger so that only the arm is stretched. Once the arm forms a straight line, the rest of the body is 'moved' as a rigid body.

The 'Head' and 'Tail' tools can be used for this. For example, apply the 'Head' tool to the shoulder of a robot. Then you can pull the finger of the hand. Inverse kinematics propagates towards the shoulder and when it reaches the shoulder, it stops and simply moves the joints at the other side of the joint.

 *IK Stopper tools*

Correspondingly, the Tail tool can be used for stopping inverse kinematics from propagating to the tail of the skeleton when one of the parent joints are modified. For example, if you move the robot from its spine, the arm keeps its current position if 'Tail' stopper is set for the shoulder.

Foot steps

A foot step system moves a character along a path defined by a set of footprints. The character's legs are moved using inverse kinematics. Characters with any number of legs can be attached to a footstep sequence. The system assumes only that each leg is either left or right handed.

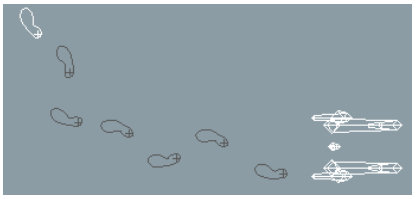
1. Model a skeleton with two legs (or load the skeleton modeled in the previous tutorials, 'tutorprojects/animation/skeleton/attachedBody').

2. Switch to the top view (so that you can see the character from top) and activate the Foot Step/Foot tool. 

3. Click the mouse to define a set of foot print objects.When a desired number of footprints has been created, select 'Accept'.

This creates a foot print object consisting of a set of left and right handed foot objects.

4. Switch to the side view. You can now see small 'step height' handles above each foot object. Move the skeleton so that its feet are positioned below these 'step height' handles.



Click the mouse on a view window to define a sequence of footprints



Note

It is important that only the joints of the feet are positioned below the line formed by step height handles. The positioning defines which joints are interpreted to be part of the feet - they will be moved using IK.

5. Select the skeleton and the foot step object (in any order) and click Foot Step/Automap tool. This attaches the skeleton into the foot step sequence.

Play the animation and the skeleton walks along the footsteps.

Modifying foot steps

Many ideas used in path animations apply directly to the footsteps too. For example, you can modify the timing by editing 'Lattice Translate' curves in the choreography window.

Each foot object inside the foot step level object defines a 'step height' handle. You can drag these handles in the edit mode to control the step height of the walking skeleton.

You can also modify positions of desired feet to get more interesting animations. Footprint objects inside the footstep object can be duplicated to get longer walking sequences.

Choreography Parametrization

Tutorial level: Intermediate to advanced

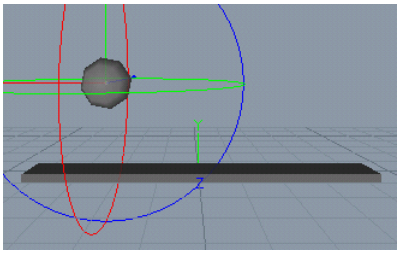
The Realsoft 3D animation system is based on the idea that you can "teach" an object to act based on some input signal. In other animation tutorials, objects were *taught* to act based on time. However, even though all actions occur over time, a time value, such as a frame number, is not the only way or the best way in all cases, to make some action occur in your scene.

This tutorial presents the basic principles for making objects perform some action based on any attribute of any object! This is actually a relatively simple notion - not nearly as complicated as it may sound. When you consider that living organisms in the *real* world respond to and take actions based on numerous types of stimuli (chemicals, light or other forms of radiation, pressure, etc.), it would seem only natural to use these techniques within an animation system.

Although it is true that you could animate anything by setting keys at specific frames, animating certain actions in this way not only can be extremely time-consuming, but prone to subtle synchronization errors. Keying actions based on the attibutes of other objects is an extremely powerful and efficient way to create a wide variety of animation effects, from plants tracking the movement of the sun to a character watching a fly buzzing around the watcher's head.

Tutorial project: 'tutorprojects\animation\input\stretchBasedOnTranslation'

Let's imagine you want to create a bouncing sphere using keyframes. You would like the sphere to bounce several times during the animation and every time it hits the floor, it should deform as if the collision between the sphere and the floor was causing the deformation.

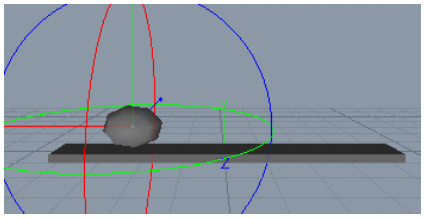


Key 1: The sphere in its initial position.

If you only want to make the sphere to bounce once, there is no problem. You can create a couple of keyframes that control the path and the deformation of the sphere. However, in making the sphere bounce numerous times, it would be great if we could make the sphere stretch automatically every time it hits the floor so that we would only have to worry about the translation of the sphere.

Let's see how this can be accomplished.

1. Create an analytic cube to represent a tabletop or a floor. Next create an analytic sphere, and in the select window, choose Make Choreographable from the popup menu. Then open the Input tab of the Choreography window. Select the 'sphere.init0' choreograph. Choose New/Keyframer from the popup menu, and then Open from the popup menu to see the created sub choreograph. Select Keyfr1 choreograph. At the right side of the choreography window, change the Input Type to Geometric Object. Now click the Change button, and scroll down the list until you see Translate. Expand the Translate folder if necessary by clicking it and select Translate.y.

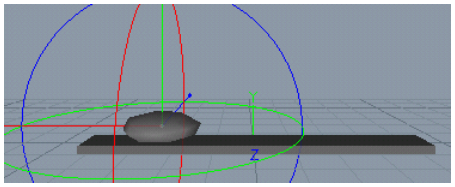


Key 2: Position where the sphere touches the floor and starts to stretch.

You are now ready to teach the sphere how it should stretch based on its Y coordinate.

2. Move the sphere downward to the position where the sphere is just about to hit the floor and starts to stretch. Activate the animation recording mode and stretch the sphere slightly in its Y direction. Then reset the animation recording mode.

3. Move the sphere further downward to the position where the sphere should reach its most deformed state. Again, set animation recording on and stretch the sphere even more. Reset the animation recording mode.



Key 3: Sphere in its lowest position with maximum stretch

That's all! Move the sphere up and down using the move tool or by dragging its Y-axis handle and the watch how the sphere stretches every time it hits the floor!

Note that if you followed the previous steps exactly, the sphere should do nothing if you move the time slider. The reason is that you have not taught the sphere to act based on time.

Let's now teach the sphere to move with time. To do this, you create a new time based choreograph, because the current choreograph is sensitive only to the sphere's translation in Y and does not react to time changes in any way.

On the choreography window, select the choreograph sphere.init0. Choose New/Keyframer from the popup menu. This creates a new keyframe choreograph for the object and makes it the current choreograph. By default, this new choreograph is time-based.

Set animation recording on and keyframe the position of the sphere so that it bounces off the floor. Note that as you move the sphere to a position at the floor, it deforms according to the attribute-based choreograph you just created. When you have keyframed several bounces of the sphere, turn off the animation recording mode and play the animation.

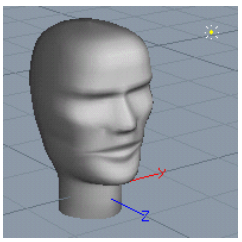
Tutorial project: 'tutorprojects\animation\teaching\StretchBasedOnTranslation'

A light sensitive fellow

Tutorial project: 'tutorprojects\animation\input\lightSensitiveHead'

Imagine that you want to create a character who does not like bright light. If the light intensity is low, the character is happy. As the light intensity gets stronger, the more unhappy our otherwise friendly character becomes.

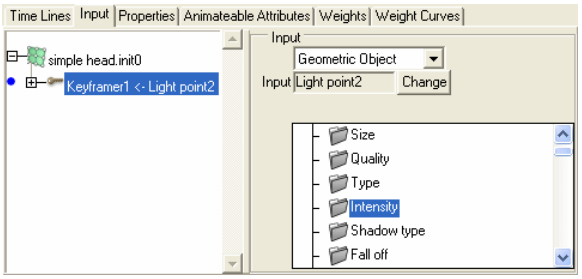
1. Create a simple head and one point light source. Position the light source so it is in front of the character and slightly above the top of the head. If you are too busy to model a head, you can load in the one from the 'models/NURBS' folder.



Create a head and a point light source

2. Select the head and in the select window, choose Make Choreographable from the popup menu.

3. Select the light source, open the choreography window and go to the Input tab. Select the choreograph head.init0. Choose New/Keyframer from the popup menu, then open the head.init0 choreography using the popup menu and click the new keyfr1 choreograph to select it. Change the input type from Time to Geometric Object and then click the Change button. You can now see all the attributes that are specific to a point light source. Select Intensity. You can now close the choreography window.



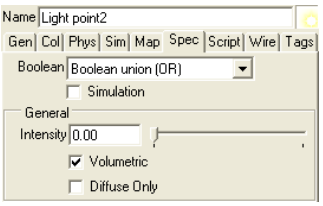
The input of the selected choreograph set to the Intensity of the light source



Note

Instead of selecting Geometric Object and clicking the Change button, you can simply drag and drop the desired geometry object from the select window over the desired keyframe choreograph to make it depend on the dropped object.

You have now set up a choreograph that is based on the intensity of the light source. From this point, all that needs to be done is to define the "key" light intensities and modify the facial expressions of the head in the animation recording mode. The only difference between this and recording a time-based choreograph is that you have to adjust the 'intensity slider' instead of the time slider.

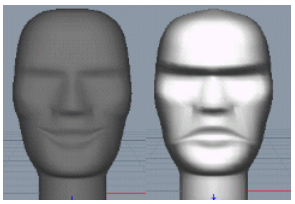


Set the Intensity of the light source to zero

4. Open the property window for the point light. Go to the Spec tab and set the 'Intensity' attribute to a low value (in this case, use zero). Then, set animation recording on and deform the face so that it looks relatively happy. When done, reset animation recording.

5. Now increase the intensity of the light source by moving the slider all the way to the right. Again, enter the animation recording mode. Now edit the character's facial features so that it looks very unhappy. Reset the animation recording mode.

You have now created two 'key' light intensities for the character. Now select the light source and play with the intensity slider in the property window. The stronger the light intensity, the more unhappy the character looks!



The face looks happy under low lighting and unhappy under bright light.

So, we have created a character who does indeed appear sensitive to light. What you actually did was morph the head's facial features as a function of the intensity of the light source. But why would anyone want to define morphing based on the intensity of a light source or some other attribute? There are many reasons, as we will show in the following tutorial examples. Keep reading and be ready for a big surprise.

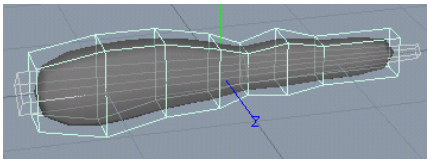
Teaching muscles

Tutorial project: 'tutorprojects\animation\input\bicep'

Imagine that you need to create an arm whose bicep muscle acts based on the angle between the two arm bones, so that the smaller the bone angle, the bigger the bicep. Think about how tedious this task would be if you had to keyframe it based on time. Not only would this be tedious, but extremely limited in application. This is where the real power of attribute based animating is exhibited.

By using attributes other than time, you can single point edit the bicep at two different bone angles to show the muscle how to deform. In this way, you now have total control over all muscle deformations. Then, you just animate the character's skeleton, making it move as you want, and all the muscles deform automatically according to your definitions.

1. To make such an arm, start by creating and shaping an SDS mesh to be your arm. Then create a skeleton consisting of two bones - make sure the first joint of the skeleton represents the shoulder joint. Remember to define a proper angle constraint for the elbow while you draw the skeleton.



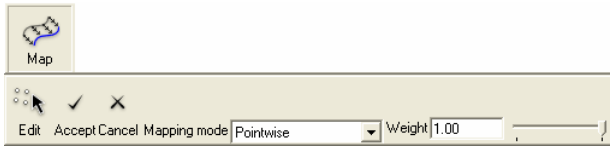
A SDS mesh and a skeleton

Position the skeleton, and then:

2. Multi select the arm and then the skeleton.

3. Activate the Lattice Mapping->Map tool to map the mesh to the skeleton.

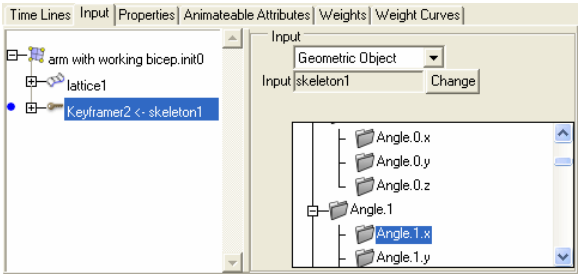
4. Set the mapping mode to the Pointwise option, and click Accept.



Map the mesh on the skeleton using the Map tool and the Pointwise option

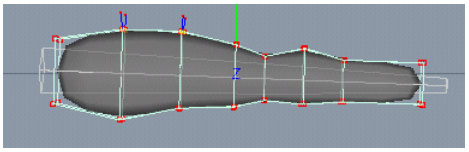
5. Select the skeleton and turn the edit mode on by hitting the spacebar. If you drag the end point of the skeleton so that the arm bends, you can see that bicep area does not react as a real bicep would. Therefore, you need to teach the bicep to get bigger (and contract) based on the bone angle.

6. To do this, select the arm and open the choreography window at the Input tab. Select the bicep.init0 choreography. Select Open from the popup menu to see its sub choreographs; there is already a lattice item. Choose New/Keyframer from the popup menu, then select the created keyfr1 object. Change the Input type to Geometric Object and select the the skeleton in the select window. Click the Change button. Then select the Angle1.x attribute of the skeleton as the input source. Now keyfr1 is a choreograph that is based on the angle of Joint1 of the skeleton.



Set the input of the choreography to skeleton's 'Angle1.x'

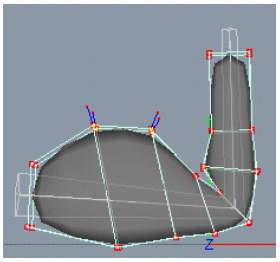
7. Select the skeleton and edit its position so that the arm is straight. Now select the arm mesh and activate the animation recording mode. Use single point editing to make the bicep area look the way you want when the arm is stretched out straight. When finished, deactivate animation recording. Now the muscle knows what to do when the skeleton is opened to its maximum bone angle.



Bicep corresponding the 180 degree bone angle.

8. Select the skeleton and move 'Joint2' so that the forearm is bent at approximately a 90 degree angle. Select the mesh again and activate the animation recording mode. Repeat the procedure described above to point edit the bicep area to make it look like the muscle is contracting and bulging. Exit the animation recording mode when done.

That's all you have to do! Now the bicep compresses and stretches based, quite naturally, on the angle between the two arm bones. Move Joint2 and see how the arm looks. There are more refinements you could do to make the arm bend look more natural. If you want to have parts of the arm animate based on other joints or joint angles, use the same technique to create additional choreographs.



Bend the arm and make the bicep bigger

Deformation based on acceleration

Tutorial projects:

- 'tutorprojects\animation\input\deformBasedOnAcceleration'
- 'tutorprojects\animation\input\bendBasedOnAcceleration'
- 'tutorprojects\animation\input\bendWithAngularAcceleration'

You have probably seen some fancy animations created by advanced simulation systems. If you have used some of these systems, you know that in order to take full advantage of them, you have to know something about physics. Another problem is that simulations are controlled by a rich set of parameters, and often it is not clear how strong an impact a change to any particular parameter value will have on the final result.

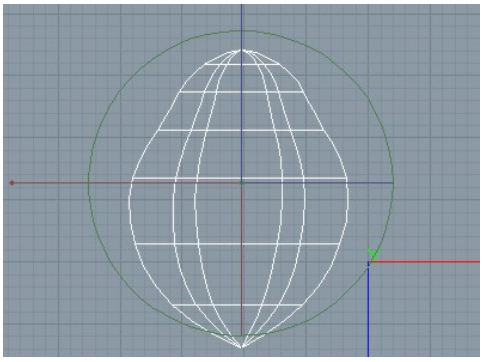
Although Realsoft 3D does include many simulation features, it would, in general, be easier in many cases to keyframe some of these fancier visual effects. Once again, this is what attribute-based choreographs allow you to do.

Imagine that you want to make some character or creature whose body needs to jiggle in a realistic way, based on the body's acceleration (for instance, jumping up and down). In the same way you made the bicep muscle bulge based on the angle of the elbow, you can apply the same principle to make a character's midsection jiggle (or deform in some way) in response to physical properties like the velocity and acceleration of the object.

1. To create this type of animation, start by creating and shaping a cubic NURBS sphere so that it looks similar to the image on the right.

2. Next, from the select window's Geometric Objects tab, right click and choose 'New->Field Evaluator' from the popup menu. This creates a field evaluator object. This object is needed for one reason only: it is capable of evaluating acceleration.

3. Select the field evaluator object and then the sphere object, and then choose Drop to a level from the pop-up menu. Your acceleration-sensitive model is now ready to be animated.



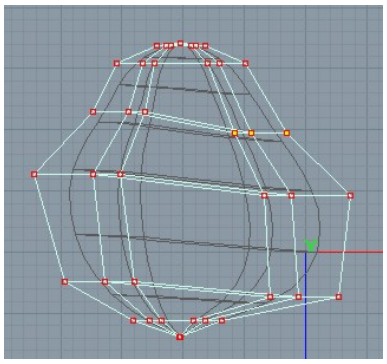
A NURBS sphere reshaped to allow for 'torso' jiggle

4. Select the mesh object, located inside the newly created level object, and choose Make Choreographable from the popup menu. Open the Choreography window to the Input tab. Select the mesh's choreograph and choose New/Keyframer from the popup menu. Open the init choreograph and select the keyfr1 object under it. Then drag & drop the Field Evaluator object from the select window into the keyframe object in the choreography list. Choose the Acceleration.y attribute of the field evaluator object.

5. Now go back to the select window. Select the field evaluator object and open the property window to the Spec tab. Set the acceleration field to the following values:

0 10 0

6. Next, activate animation recording, select the NURBS sphere and enter the point editing mode. Edit the model so that its shape is what you want it to look like at this acceleration value. Bend the torso region downward as if it was the high acceleration that deformed the object. Then deactivate animation recording.



Shape of the sphere character when the acceleration value is 0 10 0

7. Since this character will also have to decelerate (that is, respond to negative acceleration), you should also set the character's look under the opposite condition. Change the Field Evaluator object's Acceleration attribute to the following values:

0 -10 0

Activate animation recording, edit the mesh, and then deactivate animation recording.

You have now taught the mesh to 'bend' based on true acceleration. The only thing we need to do now is to animate the position of the level object containing the mesh and the field evaluator object and see how the mesh deforms according to the acceleration.

8. Make sure animation recording is still on, and modify the vertical position of the level object up/down in various time frames. When done, turn off animation recording and play the animation.

Batch Rendering

Realsoft 3D supports number of startup options to support batch rendering. This feature allows you to, for example, render multiple animations over night.

There are three startup options which are useful when using the program as a 'batch renderer'.

-play

Automatically plays the animation using the current file rendering settings.

-firstframexxx

-lastframe yyy

Specify the range to be played. If not given, uses the range specified by the project.

To use batch rendering, you have to select one of the 'rendering specification' objects in the select window (second tab on the right) and save the project as usual. Batch rendering uses the selected rendering configuration to render the project.

When -play startup option is given, the program automatically quits as soon as it gets the given animation rendered. This allows you to create script files to render any number of animations. For example, to render two animations you might execute the following script file:

```
realsoft3d -file myproject.r3d -play -firstframe 10 -lastframe 20

realsoft3d -file anotherproject.r3d -play
```



Note

The -firstframe or -lastframe switch the program to 'play range' mode and override any range settings defined by the project. If you don't pass these options, the range settings defined by the project will be used.



Chapter 10. Simulations

Simulations Overview

In the Animation chapter, you learned how to add motion to your scenes using the techniques of keyframing and motion paths, using so-called *traditional* animation tools. Although these techniques can produce impressive results, they still require the animator to define every object movement or attribute change over time. For some types of motion, this can be quite painstaking, and still may not produce results that look realistic. For example, imagine trying to keyframe the motion of a baseball after it has been hit by a batter, or the flight of a golf ball. Even more ordinary, what about a scene where two children throw balls in the air—one is a beach ball and the other a soccer ball. Do you think you could accurately or realistically keyframe the motions of these two balls? Maybe you could, but Realsoft 3D offers tools to make these types of animation much simpler: namely, the *Simulation* tools.

The simulation tools introduce a completely different approach for producing motion. Instead of defining the changes in an object's position over time, you can actually define a "physical" system (simulated of course) that can be used to control an object's motions. Think of some of the physical properties that exist in the real world: there is gravity, there are electromagnetic fields, there is friction, there are forces exerted on objects by wind and water, and there are properties of objects such as their mass, elasticity, rigidity, surface friction, etc.

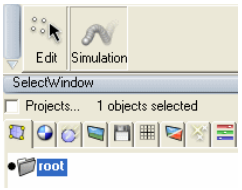
When you use simulation, Realsoft 3D computes all motions automatically, and realistically, based on the laws of physics that produce those motions. Through a combination of setting the Physical and Simulation attributes for objects (Phys and Sim tabs of the Property window), you create the type of *world* in which objects exist; for instance, an Earth-like or Moon-like environment. One advantage is that by using simulation, you can affect (animate) a large number of objects just as easily as a single object.

Our first simulation project is to create an object that emits spheres, which a fan then blows toward a wall. When the spheres hit the wall, they collide and rebound back.

Level: Intermediate

Example project: 'tutorprojects/simulation/firstsimulation'

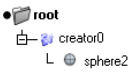
Start a new project. Select the 'root' object and click the 'Simulation' button shown in the tool control bar. This activates simulation for all objects placed under the root level.



Simulation activated for the root level

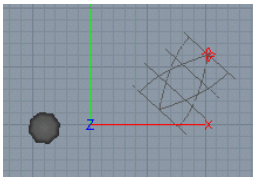
Add a *creator* object to the scene by selecting 'New/Creator' from the Select window's popup menu. Now create an analytic sphere—place it anywhere you like in the view—and drop it inside the creator object. Your hierarchy should look as follows.

Click the Animation play button. The creator object emits spheres which move in the same direction. This happens, because of the default attribute settings for initial velocity and object lifetime of the creator object.



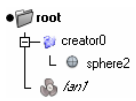
Creator object including one analytic sphere

Click the 'Simulation' tab on the toolbar, and activate the Fan tool. The Fan tool requires that you enter three points in the view window. The first point sets the anchor position of the fan, the second point sets the length of the fan object, and the third point the width. The width does not matter in this example. Enter the points as shown in the following picture to define the fan object.



A fan created

The object hierarchy should now appear as follows:

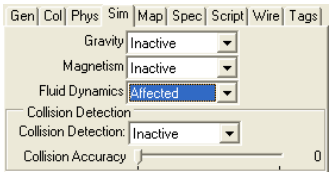


By default, all simulation effects are switched off; therefore, the fan will not yet affect the spheres. To turn the fan on so it affects the spheres, select the sphere object and open the Property window. Go to the 'Sim' (simulation) tab and set the 'Fluid Dynamics' property to 'Affected'. Now all spheres emitted by the creator object are duplicates of this *sample* sphere. All duplicates have the same fluid dynamics properties enabled, and will be affected by the airflow from the fan.



Note

The fan object should affect the spheres via fluid dynamics. We don't want the spheres to generate any fluid flows. Therefore, we set the 'Fluid Dynamics' field to 'Affected'.

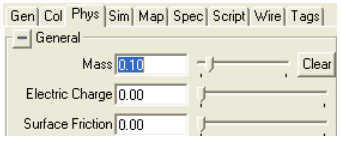


Fluid Dynamics enabled for the sphere used by the Creator object

Now play the animation. If you have followed the instructions exactly to this point, you most likely are not seeing any effect on the spheres; that is, they continue in a straight line as if the fan was not working. This is due to the default values for the mass of the sphere (1.0 kilogram) and the wind velocity (10 meters/second) produced by the fan.

Select the sphere, and on the 'Phys' tab, set its 'Mass' attribute to 0.1.

The default airflow of the fan may also require some adjusting. Select the fan, and on the 'Phys' tab set the 'Fluid Velocity' attribute to 100.0. Now play the animation again. This time, the fan blows the created spheres downward.



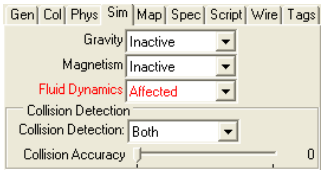
The 'Mass' property set to 0.1 for the sphere.

Note that when the animation reaches the last frame, it is automatically rewound to the first frame. If you cancel the animation before it reaches the last frame, you must manually rewind the animation back to the first frame (use the Rewind button).

The final part of this scene is to create a wall into which the spheres collide. We want the collisions to be detected so that the spheres bounce back after hitting the wall.

Create an analytic cube object, and position it so that it is in the way of the spheres blown by the fan. Play the animation and note where the spheres *disappear*, and place the wall inside of this position. Check from the other orthogonal views to verify that the spheres will hit the cube. If not, adjust the size of the cube.

To make the spheres collide with the wall, you must activate the collision detection effect for both the sphere and the wall. First, multi-select the sphere and the cube, then go to the Property window's Sim tab and set the Collision Detection attribute to 'Both'. (Multi-selecting objects allows you to set one or more attributes to the same value for all selected objects.)



Collision detection enabled for the sphere and the cube

Play the animation and note what happens - the wall is being knocked backward by the collisions with the spheres. There are two reasons for this: first, the mass of the wall is not great enough compared to the mass of the spheres, and second, the wall was set to a Collision Detection value of 'Both', meaning that it is both a *cause* of collisions, and it is *affected* by collisions.

To change the behavior of the wall so that it is not displaced by the collisions, first try setting the wall's Collision Detection attribute to 'Cause'. Play the animation. The wall is no longer moved by the spheres. However, it is still a lightweight object, which cannot bounce the objects that much. This combination makes the wall act strangely in the simulation: the locked position does not match its low mass. Reset wall's collision property back to both, and on the 'Phys' tab, set the mass of the wall to 1000.0. When you play the animation again, you'll note that the wall still moves, but is affected far less by the sphere collisions. This is because we still have a *frictionless* environment; for all practical purposes, the scene we have set up is being treated, physically, as if it were in outer space. To simulate a truly heavy (ground-anchored) wall, change the mass to for example 1000000 kilograms and set the Collision Detection attribute to Cause. Then the wall will not move and has collision behavior of a massive object.

The Collision Detection attribute is set to 'Both' for the sphere. This causes the spheres to collide with both the wall and each other. Change the value for the sphere to 'Affected' and play the animation to note the difference in behavior.

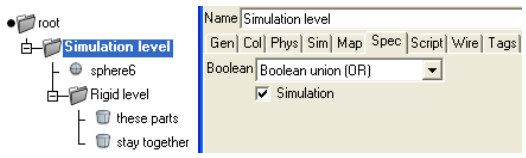
Since you now have a working simulation environment, try testing different values for both the Physical and Simulation attributes of the various objects.

In addition to the attributes available on the 'Phys' and 'Sim' tabs, each object type has *type-specific* attributes, which are located on the 'Spec' tab. For example, the Fan object has a 'Turbulence' attribute for making the airflow turbulent. This is purely a Fan-specific attribute, which is why it is found on the Spec tab.

Simulations and the object hierarchy

The simulation system has an important relationship with the object hierarchy. The 'Simulation' property, which was turned on in the previous example, is *not* a global setting that makes all objects in the scene collide and bounce around. Rather, the Simulation property affects hierarchy levels: that is, *level objects*. The Simulation property controls only the direct sub-objects of the level, and is not propagated into the sub-hierarchies of the level. This situation is shown in the following diagram.

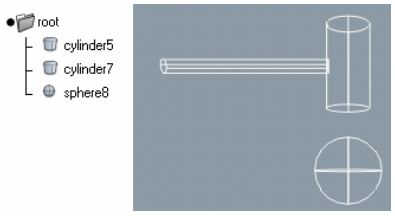
Setting up the Realsoft 3D simulation system to work this way results in an extremely flexible system with very fine-grained control. To demonstrate this, consider a scene where a hammer strikes a ball.



'Simulation' is a level attribute. Simulation will not affect the objects inside the sub-object called 'rigid level'.

Start a new project, and create two analytic cylinders to form a simple hammer, and an analytic sphere for the ball. Set up the scene as shown in the following illustration. Make sure that the hammer head lines up with the sphere in all directions.

Multi-select all three objects, and then choose 'Animations/Simulations/Collision Detection' from the menu bar. Collision detection is now activated for all three objects. Note that choosing this function also sets the Simulation property on in the parent level of the objects, which in this case is 'root'. To verify what the settings are for the three objects, open the Property window to the 'Sim' tab, and check the value of the Collision Detection property for each objec - note that it is set to 'Both' for each object.



A hammer ready to hit the ball

Now select the two parts of the hammer, click the Simulation tab on the toolbar to open it, and activate the Velocity tool. Draw a vertical line in the view window by placing the first point to the right of the hammer's head approximately at its center, and the second point about one major grid unit below the sphere. This line represents how much the hammer moves in one second.

Now play the animation. When the hammer strikes the ball, the hammer breaks. This happens because the parts of the hammer are treated as individual objects in the simulation, and no force is holding the parts together. To change the behavior, you need to change the object hierarchy.

First, stop animation playback and rewind it to the first frame. In the select window, right click and choose 'New/Level' from the popup. Rename this new level as 'hammer'.

Select both cylinders and drag & drop them into the "hammer" level. The hierarchy should now look as follows.



A new hierarchy for the simulation

Play the animation. This time the hammer drops, but the ball is not affected in any way. This is because the cylinders are now enclosed in a new hierarchy level that has no simulation attributes set yet. We will have to redefine the physical properties for the new level object called 'hammer'.

Select the 'hammer' level and then choose 'Animation/Simulations/Collision Detection' from the main menu. Open the 'Simulation' tab on the toolbar, and activate the Velocity tool. Draw a velocity line like you did previously.

Play the animation. Now the parts of the hammer remain firmly attached, and the ball moves away after being struck. However, note that the hammer is still affected by hitting the ball - it spins as if no one were holding it. Of course, no one is holding the hammer, but what if you wanted it to behave that way? A good question, but that is a topic for later. If you like, experiment with the value of the Collision Detection attribute of the objects, or the mass values of the objects and see how these changes affect the simulation.

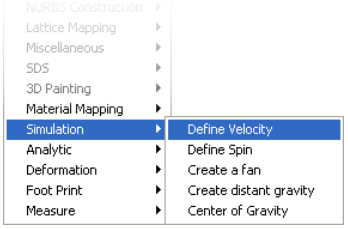
Controlling Velocity and Spin

The simulation system of Realsoft 3D controls objects using two properties: velocity and spin.

Force fields, such as gravity and electro-magnetic fields, modify velocity and spin properties according to the laws of physics. In collisions, velocities and spins change according to rebound energy, mass momentum and other physical properties of the colliding objects so that the objects will rebound correctly.

In addition to force fields, the user can also control the spin and velocity. There is nothing exceptional in these attributes. You can define and even key frame them as you like.

In many situations, it is necessary to define initial velocity and spin for animated objects. For example, when building a bowling simulation, the bowling ball needs an initial velocity.



To define a constant velocity for a sphere:

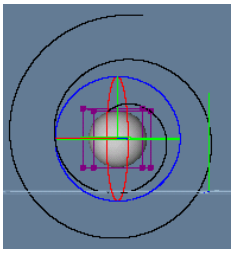
1. Make sure the root level is 'Simulated' and create a sphere.

2. Select 'Tools/Simulation/Define Velocity' from the menu bar.

3. Enter two points through a view window. The two points define the speed for the object - in other words - how much the sphere will move in one second.

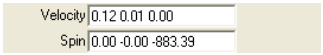
You can also make the sphere spin: select the sphere and the menu 'Tools/Simulation/Define Spin'. Enter three points through the view window: center, start angle and end angle.

The spin tool renders a 'helix' to indicate the angular velocity. The object will rotate this much in one second. (Note - one can set multiple rotations.)



Defining spin

You can also define the velocity and spin attributes numerically. To do this, select the sphere and open the property window. Then go to the 'Phys' tab and enter the desired values to the 'Velocity' and 'Spin' fields.



Defining velocity and spin numerically using the property window

The Creator Object

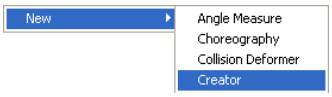
The *Creator* object is a level-object construct that generates duplicate copies of its sub-objects during an animation. However, for the Creator object to work, the level that *contains* the Creator must have the Simulation property turned on. There are many uses for creator objects: bullets being shot from a gun, bees leaving a hive, water dripping from a faucet, just to offer a few examples. Creator objects can be animated using keyframes or paths, just like many other object types.

In this tutorial, we set up a simple simulation where a Creator object throws spheres in all directions.

Level: Intermediate

Sample project file: 'tutorprojects/simulation/emitter'

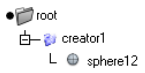
Right click in the Select window and choose 'New/Creator from the popup.



Insert a Creator object

Add a small analytic sphere inside the hierarchy of the Creator object.

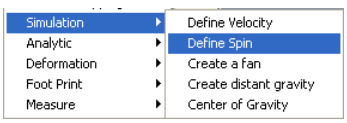
Select the creator object and then choose 'Tools/Simulation/Define Spin' from the main menu to define a spin for the creator level. You need to enter three points through the View window to define the spin. Click once near the left center of the view. Click the second point 5 cm (2 inches) horizontally to the right. Click the third point about 5 cm vertically up from the second point. This creates a counterclockwise spin.



A sphere added into the hierarchy of the Creator object

Select the *root* level and activate the Simulation property for it from the tool control bar. Now play the animation. The creator object emits spheres as it spins. However, note that only the creator object itself is spinning - the spheres it emits do not spin. Now it is time to modify some of the default values.

Select the Creator object, open the Property window's Spec tab, and examine the values of the attributes.

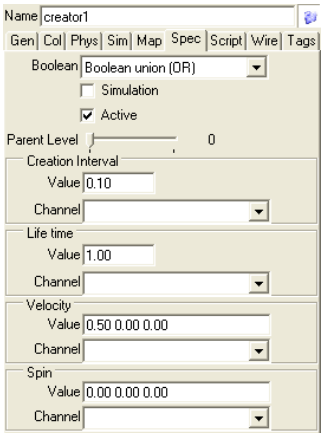


Activate the Define Spin tool, and click three points in the View window to define a spin for the Creator object.

The Parent Level attribute specifies how many hierarchy levels above the creator the emitted objects are inserted. The default is 0, which causes objects to be inserted at the same hierarchy level that contains the Creator object.

The Creation Interval attribute defines the time interval between the creation of emitted objects; that is, it defines how rapidly the Creator emits objects. The smaller the value, the more objects are created. The time interval is specified in the current time units (defined through the Options window). So, for instance, if your time units are frames, setting the creation interval to 15 will cause a sphere to be emitted every 15 frames, or two times per second for an NTSC video system.

The Life Time attribute defines the time interval that an emitted object is active before being destroyed (i.e. its lifespan). The smaller the value, the shorter the object's life. The time interval is specified in the current time units (defined through the Options window). So, for instance, if your time units are frames, setting the Life Time to 15 means a sphere is active for 15 frames (0.5 seconds in NTSC video) before being destroyed.



Attributes of the Creator object

The Velocity attribute defines the rate at which a sphere moves, in meters per second. Since this attribute is a vector value, you define velocities along the three axes. To get motion only along the X axis, you would enter a value such as "0.5 0 0". Velocity is defined in the Creator object's space.

The Spin attribute defines the rate at which an emitted object spins. Since this attribute is a vector value, you define the spin along the three axes.

Set the following attribute values (make sure the Time system is set to 'frames' in the Options window/animation tab):

- Creation Interval = 10
- Life Time = 30
- Velocity = 0.75 0 0
- Spin = 1 1 1

Play the animation. Not bad, but it could still use some fine-tuning. How about adding some gravity to make it more interesting.

Click the Simulation tab on the main toolbar, and activate the Distant Gravity tool. In a View window, enter two points so that you define a vertical line pointing down. You now have a Distant Gravity object in the hierarchy (it should be placed at the same level as the Creator object). Select the Distant Gravity object, open the Property window's Spec tab, and set the Strength attribute to 0.2.

Now select the sphere inside the Creator object level. Open the Property window, and on the Sim tab set Gravity = Affected. On the Phys tab, set the sphere's mass to 0.5. Finally, select the Creator object and on the Spec tab set Creation Interval = 5, Life Time = 90, and Velocity = 0.25 0 0. Play the animation.

Fighter plane shooting bullets

In this tutorial we create a simple FA-18 fighter flying around and shooting.

Level: Advanced

Sample project file: 'Tutorprojects/simulation/fighter'

Model a simple fighter. Make sure all fighter parts are in one level, called 'fighter'.

Insert two Creator objects into the fighter level, one for each wing. Since we are going to use path animation to move the fighter, the guns should naturally move with the fighter.

Create a small sphere that represents a bullet, duplicate it, and move one sphere inside the hierarchy of each Creator object. Position the spheres, so they are in the barrel of a gun; one sphere on the left side left, one on the right.

Select both Creator objects and open the property window's 'Spec' tab. Set the Velocity attribute to 1.0 0 0. In other words, the "bullets" will move at 1.0 m/s (real world bullets move much faster, but this way we have a chance to see them).

You can now play the animation to make sure that the pilot shoots the bullets into the correct direction. If not, either modify the Velocity attribute (try negative values), or simply rotate the creator levels. To rotate a creator level object, select it in the select window and use the transformation handles to rotate. Press and hold the shift key to rotate the creator level object with 45 degrees increments.

Click on the root level to make it the current level. Next create a NURBS curve to define a motion path for the fighter. Use the 'Tools/Lattice Mapping/Create Path Animation' tool to map the fighter to the path.

Select the parent level of the Creator objects, and activate the Simulation property. This makes the simulation system calculate the velocities of the bullets that are created.

Play the animation. The fighter shoots "bullets" as it flies.

Material controlled creation

Level: Advanced

Sample project file: 'tutorprojects/simulation/bubbles'

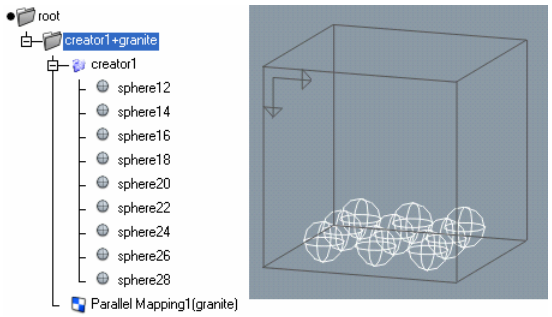
The attributes of the objects which a Creator object duplicates during animation playback can be controlled using materials. The Channel gadget below each attribute in Creator object's 'Spec' tab is used for this purpose.

The following "bubble emitter" example demonstrates how you can use this feature.

First, click the Creation toolbar and add a Creator object into the scene. Now create an array of nine spheres, arranged as shown in the following illustration, and place them all inside the Creator object level.

Next, select the Creator object, and drag & drop the "granite" material from the material library onto the View window. The hierarchy should look like that shown above.

With the Creator object selected, open the property window, go to the 'Spec' tab, and set the Velocity to the direction perpendicular to the sphere array (this would be '0 1 0' in the sample scene).

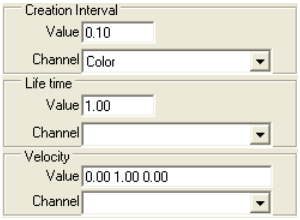


A bubble emitter system

Select Channel = Color from the drop-down list below the Creation Interval attribute. The color defined by the Granite material will define how frequently bubbles appear from each sample sphere. The actual creation frequency is the product of the material channel value and the value set in the Creation Interval field. The latter can be used to scale the values defined by the material.

Select the Parallel Mapping object. Go to the 'Simulation' tab of the main toolbar and activate the Velocity tool. Then draw a vertical velocity line downward.

Select the parent level of the Creator object and activate the Simulation property from the tool control bar.



*Creation Interval is 0.1 * color of a sample object*

Play the animation. The creation frequency is different for each sample sphere. In addition, note that because the granite mapping moves, the creation frequencies also change continuously.

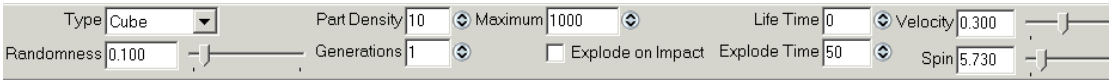
Explosions

In this example we will set up a simple simulation where a sphere explodes after a certain period of time.

1. Create a sphere. Animate it as you like, for example define a velocity value for it.

2. The sphere still selected, go to the Simulation tab of the toolbar and click the Explode tool. A new Explosion modifier appears to the hierachy.

3. The tool control bar shows the properties of the explosion modifier. Set Type to Cubes, Part Density to 10 and Explode Time to frame 50.

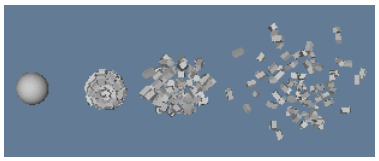


Explosion options

4. Play the animation. Sphere will explode after 50 frames.

This effect breaks the exploding object to a number of pieces. The number of explosion parts can be controlled through the Part Density field.

The Life Time field can be used for controlling how long the exploded parts remain alive.



Exploding sphere

Explode at impact

In addition to a specified time, objects can be exploded by collisions. For example, you can create a gun, which shoots exploding bullets.

To create a simulation where a bullet explodes its target object:

1. Create two spheres: one representing a bullet and one representing a target.

2. Use the Simulation/Velocity tool to define an initial velocity for the bullet so that it will finally hit the target sphere.

3. Select both spheres and activate the Collision tool in the Simulation tooltab. The default options are suitable, so simply click Accept.

4. Select the target and apply the Explode tool.

5. The toolbar shows the option of the Explode modifier. Set Explode Time to zero to switch off time based explosion. Activate the Explode on Impact option. Then play the animation.

Explode at a specified time

Explosion is controlled by an Explode attribute. As soon as this attribute is set, the object explodes. Explode Time and Explode on Impact options simply set this attribute when a certain time is elapsed or when a collision is detected.

The Explode attribute can be animated in a way similar to animating other attributes. For example, if you want a sphere to explode at frame 30:

1. Create a sphere and make it choreographable by applying the menu Make Choreographable from the select window's popup.

2. Open the Choreography window. In the Time Lines tab, select the Sphere.Init0 choreography. Then select New/Keyframer from the popup menu. Open the sub structure of the Init choreography to see the inserted keyframe object. Click the keyframer object to select it.

3. Go to the Animateable Attributes tab. Select the Explode attribute and set the Animated option.

4. In the Properties tab, open the keyframer object and click on the Explode attribute under it. Right click in the graph window and select Set Minimum and maximum values from the popup menu. Make sure that Min x and Max x values match the length of your animation: Min X=0, : Max X = the number of frames in the animation. Make sure that the maximum value for Y = 1. Check the Rescale box.

5. Select Create predefined curve/Constant curve from the popup menu. Add one point to the created curve by dragging the curve. Because the type of the Explode attribute is Boolean, also the type of the created curve is Boolean. Drag the added point to the value 1.0 at frame 30 (X = 30).

6. The sphere selected, go to the Simulation tab of the toolbar and click the Explode tool. Set its Explode Time to zero to eliminate the basic modifier control. A target specific value, defined by the keyframer, will be used instead.

7. Play the animation. Now, when time reaches frame 30, the explode attribute value goes up to 1 and the sphere explodes.

Explode at other events

As you already know, the animation system allows you to animate 'anything as a function of anything'. This is also true for explosions. However, it usually does not make much sense to key frame the explode attribute, because as soon as the explode key frame curve goes up, the object dies taking all the key frame curves with it.

There are numerous cases where the explode option should be controlled by another object attribute. For example, you might want to explode an object as soon as its speed exceeds a certain limit. To achieve this, animate Explode as a function of Speed. Or, you might want to explode an object as soon as its 'y' coordinate reaches a certain upper limit. In this case, animate the Explode attribute by the Translation.Y attribute.

Let's create an animation where an object explodes as soon as it moves below the ground plane, in other words, when its y coordinate gets smaller than zero. The Explode attribute will not be animated in time but by the y coordinate of the object.

1. Create a sphere and apply Make Choreographable to it from the select window's popup.

2. Open the Choreography window and select the sphere's base choreography 'sphere.init0'.

3. Select New/Keyframer from the popup menu of the choreography list. Open the sub structure of the Sphere.init0 choreography and activate the new keyframer choreography.

4. Go to the Input tab of the choreography window. Drag the sphere object from the select window and drop it on the keyframer object in the choreography window. This instructs the choreography to use one of the sphere's attributes for input. Select Center.y from the Attribute list.

We are now ready to animate the Explode attribute as a function of the sphere's y coordinate.

5. Go to the Animateable Attributes tab. Set the Animated option for the Explode property. Then go to the Properties tab and click on the Explode attribute under the Keyframer choreography.

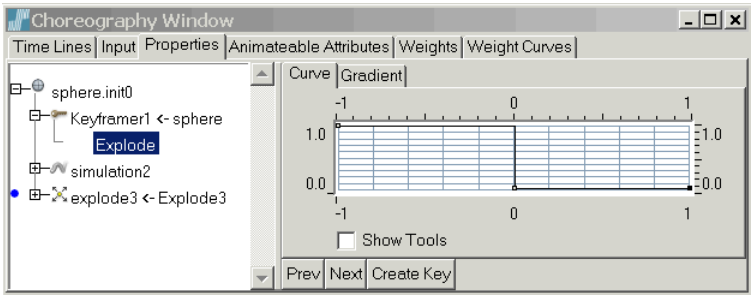
6. Now select Create predefined curve/Constant curve from the curve gadget's popup menu at the right side of the Choreography window. Select Set Minimum and Maximum values from the curve gadget's popup. Define 'Min X' = -1, 'Max X' = 1 (this is the input parameter range i.e. the y coordinate of the sphere center). Define 'Min Y' = 0,

'Max Y' = 1. The vertical Y-axis controls the state of Explode attribute. Check the Rescale box and close the scaling dialog from the OK button. Then define the curve shown below.

7. Use the Simulation/Velocity tool to define an initial velocity for the sphere, making it fall downwards.

8. Add an Explode modifier controlling the sphere. Set its Explode Time to zero.

Play the animation. The sphere moves down and when it reaches the ground plane, it explodes.



Explode is 1 when Center.y is below 0



Note

Explode modifier should usually be the last choreography applied to an object, so that explosion takes all motions and animation details into account. So, if you add new animation effects after setting an object to explode, remember to move explosion to the end of the choreography list.

Gravity

Gravity is always an attractive force, which affects the objects by trying to pull them towards each other. According to Newton's formula:

$$F = g * \frac{m1 * m2}{r^2}$$

Above 'g' is the gravity constant, 'm1' and 'm2' are the masses of the bodies and 'r' is the distance between the bodies. In other words, the higher the mass, the stronger the force. The shorter the distance, the stronger the force. The force is inversely proportional to the square of the distance.

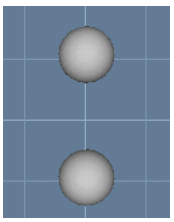
Because the gravity simulation system follows Newton's laws of motion, one can simulate planet systems and other physical phenomena.

Planets orbiting each other

1. Create two spheres: one near the top edge of the view and another one near the bottom edge of the view.

2. Open the Property Window and set the mass for the both spheres to 100000000 kg. The default 1 kg mass is way too light to generate a noticeable gravity field.

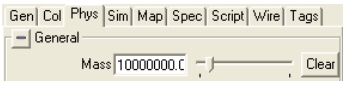
3. Further down on the Phys tab, set the Velocity of the first sphere to '-0.1 0 0' and that of the second sphere to '0.1 0 0'.



Two spheres representing planets

4. Go to the Sim tab and set Gravity to 'Both' for both spheres.

As always, make sure the parent level has its 'Simulation' option set and play the animation.



Set the mass through the 'Phys' tab of the property window



Note

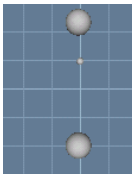
If the velocities of the spheres are too high, the spheres fly away from each other. If velocities are too low, planets approach each other along a spiral orbit and eventually collide.

Example project: 'tutorprojects/simulation/orbiting_planets'

A three planet system

Let's add a moon to the 'orbiting planet' example created above.

Example project: 'tutorprojects/simulation/three_planets'



A three planet system. The moon orbits the upper sphere.

1. Rewind the animation back to frame 0. Move the spheres further apart to make room for the moon.

2. Create a new sphere, a bit smaller than the two planets, which will act as Moon. Open its properties at the Spec tab and set Mass to 100. Set the velocity of the moon to [-0.35 0 0]. This will prevent the moon from falling into the upper planet.

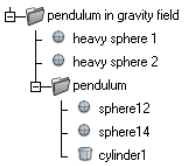
3. Go to the Sim tab and make sure that for all spheres gravity is set to 'Both'.

Play the animation. If you defined the masses and velocities correctly, the moon should orbit the upper sphere. Note how both planets affect the orbit of the moon.

Pendulum

Gravity treats all individual objects, such as spheres, as if their entire mass were collapsed to a single point called 'center of gravity'. Because of this, gravity cannot affect the 'spin' property of single objects. Two orbiting spheres will never start to rotate.

You can define non-homogenous mass distributions by constructing objects from several sub objects. For example, you can create a long pendulum object, which will start to oscillate in space because the gravity pulls its ends with different forces.



The pendulum simulation consists of three objects: two heavy spheres and the pendulum.

Let's see how this works:

1. Drag & drop a 0.1 meter grid into the view window to activate grid snapping.

2. Create two heavy spheres with equal masses (100000000 kg, for example)

3. Create a level object consisting of three sub objects: two spheres at the ends and a cylinder connecting the spheres. This is our 'pendulum' object. The pendulum object must be positioned exactly in the center of the two heavy spheres.

4. Rotate the pendulum about its center point slightly (say 30 degrees).



The rotated pendulum

5. Select the two heavy spheres and set Gravity to Cause through the Sim tab of the property window. This way the two heavy spheres affect only the pendulum, but they don't attract each other.

6. Set Gravity to Both for the pendulum.

Turn on Simulation option for the parent level and play the animation. The Pendulum starts to oscillate.



Note

If you wish to treat an object, which consists of several parts, as a single entity in gravity computations, just define a mass for the whole composite object using the property window's Mass control. The mass value defined by the level overrules its children's mass values. This naturally speeds up the simulation,

because only one single point, the center of gravity, is considered. If you have set a mass value for the complete level and later wish to restore the use of child-defined mass distribution, click the Clear button behind the mass control.

Example: 'tutorprojects/simulation/pendulum'

Elastic planet

The previous tutorial demonstrated how gravity affects the spin of hierarchical objects. Gravity can also deform objects (a poor astronaut falling into a black hole can turn to spaghetti).

Example project: 'tutorprojects/simulation/elastic_planet'

1. Create an analytical sphere representing a heavy planet. Set the mass of the sphere to 100000000 kilograms.

2. Create another NURBS or SDS sphere (i.e. a freeform object that allows deformations). Go to the 'Phys' tab and define the following physical properties for it:

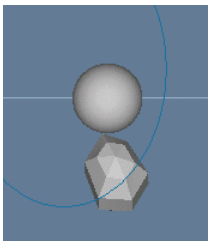
- Rebound Energy: 0.1
- Elasticity: 4
- Rigidity: 0
- Velocity: [0.5 0 0]

As usual, we have to define some initial velocity, which allows the elastic sphere to orbit the heavy sphere. In this case the velocity should be such that it gives a very elliptic orbit for the elastic sphere.

The 'Elasticity' property means that the elastic sphere has internal string forces, which attempt to restore the initial shape. The higher the value, the stronger the string forces are. The result is a 'jelly' like material.

'Rigidity' 0 means that the object can be deformed. For example, warm butter is a good example of a material which can be deformed but which is not elastic. Jelly is an example of an elastic and non-rigid material.

'Rebound Energy' determines how much energy is lost in deformations. If the rebound energy is 1, no energy is lost and the elastic planet would keep jiggling forever. A rebound energy of 0.1 means that jiggling will fade rapidly (due to friction between material particles).



An elastic object deformed by gravity

3. Select both spheres and set 'Gravity' to 'Both' through the 'Sim' tab. Then activate 'Simulation' for the parent level.

Play the animation. The elastic sphere deforms significantly when passing by the heavy sphere.

Birth of a spiral galaxy

Let's create a galaxy, which has a massive black hole in its center.

Example project: 'tutorprojects/simulation/galaxy'

1. Create a small but heavy (100000000 kg) sphere representing the black hole.

2. Use the particle tool to create lots of stars (1D particles) around the black hole.

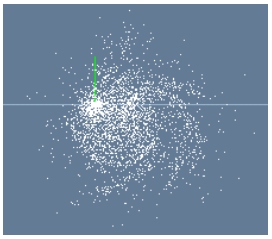
3. Set gravity for both objects to 'Both'.

4. Set the following physical properties for the particle object:

- Rigidity: 0
- Spin: [0 0 1].

5. Set the Simulation option for the parent level and play the animation.

The particle object is made completely non-rigid so that gravity can affect each particle as a separate object. Some initial spin was added for the particles so that they orbit around the black hole instead of falling into it straight away.



A spiral galaxy



Note

Use a high frame count to preview this animation (say, 1500 frames). After some seconds the particles should form a spiral galaxy. You also may have to increase the Spin value to see the effect. Zoom out the view window to see the whole galaxy appearing.

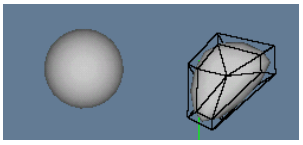
Collision Detection

The collision detection system of Realsoft 3D can detect collisions between any objects. You are not required to define any type of bounding box for an object - just activate collision detection, define suitable physical properties, and play the animation.

Level**:** Intermediate

Sample project file: 'tutorprojects/simulation/collisions'

Create an analytic sphere near the left edge of the view and one subdivision sphere in the center of the view. Select the subdivision sphere, enter the point editing mode, and move a couple of points to deform the second sphere.



Two spheres to be used for a collision

Select the analytic sphere and choose Tools/Simulation/Define Velocity from the main menu. Enter two points in the View window: enter the first point in the middle of the the analytic sphere, and the second point on the right side of the sphere. The analytic sphere will move the length of this line in one second.

Select both spheres, and then choose Animation/Simulations/Collision Detection from the main menu. This menu function turns on collision detection for the selected objects, and also enables the Simulation property for the parent level.

Now play the animation to see how the spheres bounce after colliding.

Note that collision with the SDS sphere is based on the true surface boundary, not the control polygon. If the shape of a colliding object is irregular and the object moves very fast, it may occasionally penetrate other objects. This can be controlled using the Collision Accuracy attribute (located on the Property window's Sim tab). The default value of zero is usually sufficient. Higher values improve accuracy, but at the cost of animation playback speed. If the Collision Accuracy attribute value is not the same for the two colliding objects, the highest value is used. Try adjusting the value of this attribute and compare the results.

Collision deformations

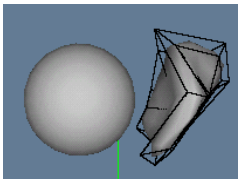
Level**:** Intermediate

Sample project file: tutorprojects/simulation/elastic_collision

A collision can also deform the colliding objects. Let's make the previous example more interesting by making the SDS sphere elastic.

Select the SDS sphere, open the Property window to the Phys tab, and set the following attribute values:

- Elasticity = 4.0
- Rigidity = 0.1



A SDS sphere deformed by a collision

These attribute settings should make the SDS sphere respond as if it were made of a semi-soft, elastic rubber compound. Play the animation. The collision with the other sphere causes it to jiggle (or oscillate).



Note

Elasticity is a valid attribute only for freeform objects, such as SDS and NURBS objects. Analytic spheres and cubes, for example, cannot be deformed because their geometry does not contain enough points.

Selective collisions

Level**:** Intermediate

Sample project file: tutorprojects/simulation/selective_collision

In the previous two examples, collision detection was activated by selecting the objects and using the Animation/Simulations/Collision Detection function from the main menu. This tool causes collisions to be detected between all of the selected objects. It also sets all the necessary collision attributes for the selected objects. You may have noticed that in the previous examples the Collision Detection attribute was set to the value "Both" for the two spheres, meaning that each object both causes and is affected by collisions.

However, it may not always be the case that you want every object to both cause and be affected by collisions, even though in a realistic universe, this is always the case. One reason to be more selective in setting the value of the Collision Detection attribute is animation playback speed. The computational requirements grow much more rapidly than the number of colliding objects—in fact, the demand is proportional to the square of the object count. For example, if you increase the object count by a factor of 10, the number of object pairs (potentially colliding) grows by a factor of 50.

Manually defining which objects cause collisions and which are only affected by collisions can be used to decrease the computational work for collision detection, often without affecting the realism of the scene. The following example demonstrates this situation.

Create one big analytical sphere (a planet) in the middle of the view window. Then create a dozen smaller analytical spheres (meteorites) around it.

Select the planet sphere and open the Property window. On the Phys tab, set the mass to 1 000 000 000 kg, so that it creates a strong gravity field.

Switch to the Sim tab, and set Gravity=Cause and Collision Detection=Cause.

Multi-select all the meteorite spheres, and set Gravity = Affected and Collision Detection = Affected. Close or minimize the Property window.

Select the parent level of all the spheres, and turn on the Simulation property by clicking the button on the tool control bar.

Set the total frame count in the Animation window to, say, 600 frames. A message pops us asking to scale the time lines. Click No and then and play the animation.

Inertia

Level**:** Intermediate

Sample project file: tutorprojects/simulation/inertia

The Inertia attribute specifies how the mass of an object is distributed inside its volume. If an object's inertia is low, then the mass is located near the center of gravity point; this, in turn causes strong spins during collisions. The higher the inertia, the more the mass is distributed throughout the object's volume, and the stronger the object resists spinning. By default, the program estimates the initial inertia values for an object based on its mass by assuming that the volume is filled with homogeneous matter.

The Inertia attribute is located on the Property window's Phys tab. It is best to use the default inertia as a starting point for experiments. It is easily possible to define inertial values that are physically impossible (at least in this Universe). The result will be that the object exhibits unnatural behavior in collisions, such as wild spinning.

So let's try an experiment with inertial values.

First create three analytical spheres to use as test objects. Stretch two of the spheres so that they become elliptical.

Select the round sphere, open the Property window to the Phys tab, and set the mass to 1 million kg.

Select one of the elliptical objects, and on the Phys tab examine the value of the Inertia attribute. The program sets the default mass of an object to one kilogram; therefore, if the radius of the object is about 10 centimeters, the inertial values will be quite small. Decrease the inertial values along each axis by a factor of 10 by adding a zero after the decimal point. For example, values of 0.05 0.05 0.05 would become 0.005 0.005 0.005. Such values will make this object have an unusually low inertia, making it behave like a heavy lead ball thickly coated with lightweight plastic foam. Make sure that you have defined a suitable number of decimals in the Options Window/Metrics tab.

Select the second elliptical object, and again examine the value of the Inertia attribute. This time increase the value by a factor of ten by removing one zero after each decimal point. For example, values of 0.05 0.05 0.05 become 0.5 0.5 0.5. These values make the second elliptical object have a high inertia, making it behave like a hollow lead sphere having all the mass close to the outer surface.

Close or minimize the Property window, then select all three objects and choose Animation/Simulations/Collision Detection from the main menu.

Define a suitable velocity for both the elliptical objects using the Velocity tool, located on the Simulation tab. The velocity vectors should be drawn in such a manner that each object will hit the heavy sphere during the animation.

Set the total frame count in the Animation window to, say, 500 frames (do not scale time lines when doing this). Play the animation and compare how each of elliptical objects spins after its collision with the heavy sphere.

After experimenting with Inertia values, you can use the Clear button to the right of the Inertia field to restore the original default values, which usually produces quite suitable and natural spinning.

Surface friction

Level**:** Intermediate

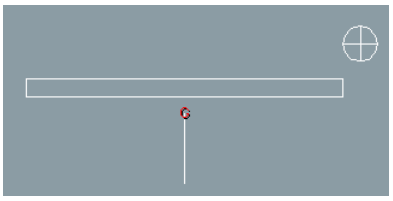
Sample project file: tutorprojects/simulation/surface_friction

An object's surface friction plays an important role in collisions. The following example, a sphere rolling on a table, demonstrates this property. Note that a collision is not always a catastrophic event. A ball rolling on a floor or table is still a collision, albeit, a mild one.

Create a cube to represent a table, and a NURBS sphere slightly above and to the right of the table as shown in the image below. Remember to verify from the side view and top view that the sphere is really aligned with the table.

From the Simulation tab of the toolbar, activate the Distant Gravity tool and draw a vertical line pointing downward. This defines the direction of gravity, not its strength. Select the Distant Gravity object in the select window, double click it to open its properties. At the Spec tab set the Strength value to 2.0.

Select the sphere, and from the Simulation tab activate the Velocity tool. Draw a horizontal velocity line from the center of the sphere to about the middle of the table.



Objects for surface friction simulation

Select the sphere, open the property window at the Sim tab and define the following parameters:

- Gravity = Affected
- Collision Detection = Affected

Select the cube object representing the table and define the following properties:

- On the Sim tab, set Collision Detection = Cause
- On the Phys tab, set Mass = 1000000 kg

Although real world tables are not this heavy, they stand on a firm floor, which in turn is anchored to a house, and hence to the Earth. Therefore, the mass resisting the collision is quite large. Note that even though the table will not move when Collision Detection = Cause, it needs a proper mass to create an impact with the sphere.

Select the parent level of your objects, and make sure the Simulation property is on. Play the animation. The sphere should bounce across the table. It may rotate somewhat due to its initial collision with the table and limited computation accuracy. Try also playing with the Collision Accuracy value. Let's see what happens when we give both the table and the ball a non-zero Surface Friction value.

Select both the cube and the sphere. Open the property window to the Phys tab and set Surface Friction = 0.1.

Play the animation again. The sphere stops bouncing rather quickly, and begins to roll. The Surface Friction attribute has the following properties.

- It is defined by the minimum value of the attribute of the two colliding objects. (Even the best tires cannot help a driver on a wet icy road.)
- The value range is from zero to one. At a value of 1.0, no slipping will occur on the surface.

To test this last property of the Surface Friction attribute, set the value for the table and ball to 1.0 and replay the animation.

Rebound energy

Level**:** Intermediate

Sample project file: tutorprojects/simulation/decreasing_rebound

The Rebound Energy attribute defines how much of an object's kinetic energy is lost in each collision. If an object's rebound energy value equals 1.0, this produces a fully elastic collision; that is, the object does not lose any kinetic energy. For example, if a rubber ball hits a concrete wall, it bounces away with exactly the same speed as it had before the collision. If an object's rebound energy value equals zero, this produces a fully non-elastic collision - the object loses all its kinetic energy, such as an egg hitting a concrete wall. When a collision is non-elastic, the majority of kinetic energy is lost (it is converted to heat). If two objects with zero rebound energy collide, they stick together.

To experiment with rebound energy, set up a scene as follows.

Create two NURBS spheres. Multi-select them both, and from the main menu choose Animation/Simulations/Gravity, followed by Animation/Simulations/Collision Detection. Then set the mass of each sphere to 200 million kg.

Play the animation. The spheres should attract each other until they collide, then continue hitting each other and bouncing apart. If the simulation proceeds too slowly, stop the playback, rewind the animation, then increase the mass of both objects. If you look at the attribute values on the Phys tab for the two spheres, you'll see that the Rebound Energy attribute is set to 1.0 (the default). Try setting the Rebound Energy value of both spheres progressively lower (0.75, 0.5, 0.25), and replay the animation. As the value is lowered, rebounding decreases and finally stops.

Electro Magnetism

Electro magnetic force is generated by the electric charge of an object. Unlike gravity, electromagnetic force can be both attractive and repulsive.

A compass is a good example of this force type. The needle points to the North Pole, because the sign of the force field at the North Pole is opposite to the sign of the needle of the compass.

A compass

1. Create two small analytic spheres representing the north and the south poles.

2. Select the north pole sphere and go to the Phys tab in the property window. Set Electric Charge to 10. Then set the electric charge of the south pole to -10.

3. Create a level object consisting of two sub objects representing the two ends of the compass needle. Set electric charge of the 'north' end sub object to -1 and electric charge of the south end to 1.

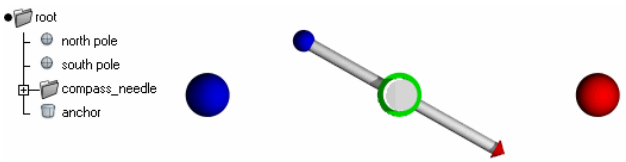
4. Select the 'pole' spheres. In the Sim tab, set Magnetism to Cause.

5. Select the needle level object. In the Sim tab, set Magnetism to Affected. In other words, the needle does not generate a magnetic field, it is just sensitive to the field generated by the north and south poles.

6. Remember to turn on Simulation of the parent level of the compass needle.

In the real world, even tiny compass needles generate a magnetic field. The field is just so weak that most magnetic objects near the compass do not notice it. If you want the needle to generate an electro magnetic field, select Magnetism = Both option. If you have lots of compasses and the field generated by the needle is irrelevant, you might want to switch it off - this simplifies the simulation and increases animation playback speed.

The sample project 'tutorprojects/simulation/compass' includes a slightly more interesting setup: the needle contains two connecting rods and a cylindrical ring in the middle. A 'heavy' pole goes through the ring, anchoring the needle using collision detection. The ring and pole have some friction and their rebound energy is less than 1. This stabilizes oscillations of the needle when it orientates to the magnetic field.



A compass simulation

Fluid Dynamics

With the help of the fluid dynamics simulation tools of Realsoft 3D, you can easily animate various fluid or air flow related phenomena:

- A falling object accelerating slowly towards the terminal velocity
- A fan that blows a bunch of objects away
- Aerodynamic properties of a moving dart turn its peak forward

The first example demonstrates how atmosphere resists rapid motions. Because of this, the velocity of a falling object does not increase forever but approaches the so called terminal velocity.

Tutorial level: Medium

Tutorial example: 'tutorprojects/simulation/falling_objects'

1. Create two analytic spheres, one big and one small.

2. Activate the Distant Gravity tool from the Simulation tab of the toolbar. Draw a vertical line showing the direction of gravity on a view window.

3. Select the two spheres. Open the property window, go to the Sim tab and set Gravity = Affected for the spheres.

4. Select the parent level and turn Simulation on from the tool control bar.

5. Set the frame count to 500 and play the animation.

The natural earth-like gravity accelerates the spheres very rapidly and they disappear from the view window. Stop the playback after about 50 frames. If you zoom out to see the spheres, they have fallen equally much. The velocity does not depend on the size of spheres.

6. Rewind the animation.



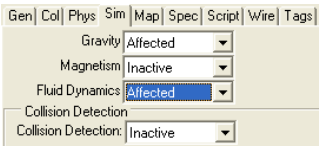
The fan tool selected

7. Select the Fan tool from the Simulation tool tab. Draw a fan object with three mouse clicks on a view window: 1. fan position, 2. fan direction, 3. fan radius. The size and orientation of fan is not important this time.

8. Go to the Phys tab of the fan's property window. Set Fluid Velocity of the fan to zero (default value is 10 m/s). The reason is that we do not want any wind blowing in this simulation. The fan only adds a non-flowing atmosphere which resists motions.

9. Increase the Density value (it is also in the Phys tab) of the fan from 1.0 to 10.0 kg/cubic meters. The denser the media, the better it resists motions.

10. Select the two spheres. Switch to the Sim tab and set Fluid Dynamics = Affected.



Turn 'Fluid Dynamics' on

11. Play the animation. The spheres start falling again. Now the smaller sphere falls faster. The spheres have an equal default mass of 1 kg. However, the small sphere has a smaller area causing air friction, and therefore it can fall quicker. This is a correct result - a small stone falls quickly, but if you put it inside a balloon it falls slowly.

You may stop the playback again after 50 frames. The property window shows that velocities are now smaller, probably less than 20 m/s. Rewind the animation.

12. As a final step, select the two spheres and decrease the Mass value to 0.1 kg using the Phys tab of the property window. Play the animation again. Now the air resistance is quite significant because of the low weight of the spheres. Both reach their terminal velocity quickly during the first 50 frames. The terminal velocity of the bigger sphere is quite low, less than 5 meters per second.

A funny fan

You may have already noticed that the Fluid Dynamics gadget includes the same list of alternatives as the Collision Detection and force fields: Cause - Affected - Both - Inactive. This means that any object can generate the fluid medium and a flow to it. However, there is one special purpose object available for this purpose, namely the fan object.

A fluid flow generated by an ordinary object, for example a NURBS line, is constant everywhere. A fan can generate non-uniform and turbulent flows. The following example demonstrates this.

Tutorial level: Medium

Example project: 'tutorprojects/simulation/fan'

1. Create a Creator object by using the select window's popup menu New/Creator.

2. Create a small sphere to the left edge of the view window. Open the property window, go to the Sim tab and define Fluid Dynamics = Affected. Even a strong wind cannot move small round objects that weigh 1 kilogram (probably made of lead). Therefore, switch to the Phys tab and change Mass to 0.001 (= one gram). Then drop the sphere into the sub hierarchy of the creator object.

3. Activate the Fan tool from the Simulation tool tab. Draw a fan in the middle of the view window. The third mouse click defines the fan diameter; it should be about 10-20 % of the view window's width (see the image below).

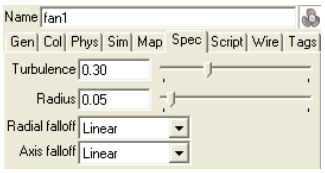
4. Go to the Phys tab of the fan's property window. Set the Spin value of the fan to '4 0 0'. This will make the fan rotate. Increase the Density to 10 and the Fluid Velocity value to 200 m/s. This is a truly powerful fan, which can move the spheres easily!



The fan in action

5. Switch to the Spec tab containing the fan specific options. First set Turbulence to 0.3 - this will produce slightly turbulent airflow, like all real world fans do.

6. Both Radial falloff and Axis falloff gadgets are Constant by default. This means that the fan is really strong - it creates a wind that blows through the whole universe. Change both Radial falloff and Axis falloff to Linear. Now the airflow will drop to zero outside the radius of the fan, and get weaker as the distance along the fan axis grows (flow is inversely proportional to the distance). The third alternative Quadric also produces a local airflow, but the falloff is more rapid.



Properties of the fan

7. Close the property window. Turn on Simulation of the parent level and play the animation.

A flying dart

So far, we have used simple symmetric objects for testing fluid dynamics. We have already seen some examples of force fields that spin objects having a suitable sub structure:

- A pendulum in a gravity field
- A compass needle in a electro magnetic force field

By using compound objects consisting of several parts, the same kind of fluid effects can be achieved. The fluid simulation system examines sub hierarchies of objects and each sub component generates its own contribution to the total result. If the mass/area distribution is not symmetric, the result force rotates the object.

The next example, a flying dart, shows this kind of situation. The dart is thrown and it starts falling in a gravity field. The shape of the dart is such that airflow automatically turns the sharp peak of the dart to point to the direction where the dart is moving.

Tutorial level: Medium

Example project: 'tutorprojects/simulation/dart'

1. Model a dart using suitable components. For example, model the two wings at the tail by using two spheres, which are stretched flat in one dimension. The body of the dart can be a cylinder with two spheres at the ends. An analytic cone forms the peak. Enclose all the components in a hierarchy level called 'dart'.



A dart

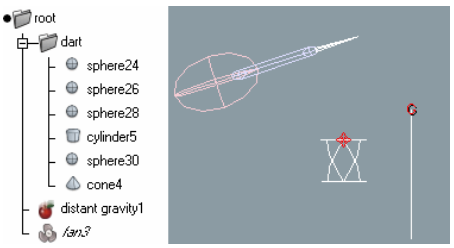
2. Place the dart to the left edge of the view. Direct it to the right and slightly upwards, which is a good starting position for the dart.

3. Go to the Simulation tab of the toolbar. Make sure the dart is selected and start the Velocity tool. Draw a velocity line halfway through the view, to the direction of the dart. The line shows how you 'throw' the dart.

4. Select the root level, then click the Distant Gravity tool of the Simulation tab and draw a gravity line pointing downwards (the length is not important, only the direction).

5. Activate the Fan tool and draw a fan in a view window. Then select the parent level of created objects (root) and turn Simulation on from the tool control bar.

All the required components of the simulation are now included, but we have to adjust some simulation and physical properties. Let's start from the dart itself:



The components of the dart simulation

6. Select the dart level and open the property window. Go to the Sim tab. Set Gravity = Affected and Fluid Dynamics = Affected.

Now the simulation already works, but the default properties are quite poor. For example, the default weight of the small dart is several kilograms!

7. Go to the Phys tab. Select the tail components of the dart and set Mass field to 0.001 (one gram). Remember you can set the number of decimals in the Options/Window/Metrics tab from the main menu. Then select the body components and set their mass to 0.003 (three grams). Finally, select the peak and the front part of the body, and set the mass to 0.01 kilograms. Note that with these values, the center of gravity will be in the front part of the dart. If you have tried playing with 'real' darts, you know that they are designed this way.

Now the dart works better. However, in earth's gravity a dart falls very rapidly and hence it must also fly very rapidly to reach its goal. In order to see properly how the dart flies, let's define a 'slow motion' simulation setup.

8. Select the Distant Gravity object from the hierarchy. Go to the Spec tab and decrease the Strength from 9.82 (earth gravity) to 0.1. Now the dart will fall more slowly.

9. Select the Fan and go to the Phys tab. Set Fluid Velocity to zero (no wind).

10. A normal atmosphere does not affect the 'slow motion' dart much. Therefore, increase the Density value from 1 to 10.

You can now close the property window. Set the fame count to 150-250 frames, turn on Simulation for the Root level and play the animation to see the result.



Note

The dart will turn easier if you decrease the default Inertia value. The program estimates the default inertia from the mass and object dimensions. The estimated value can be inaccurate. real dart's mass

is heavily concentrated in the front part, which is made of heavy metal, whereas the plastic tail does not weight much. Therefore, the correct inertia value is quite low. For example, divide all the three default inertia components by five.

Scripting in Simulations

You can control various simulation specific attributes by using scripts. This helps to control the behavior of objects acting under influence of force fields.

There are several ways to use scripting in simulations. Firstly, you can attach a script directly to an object. You can also create a special *constraint* scripting object and add it to a hierarchy level in order to control all objects in the same hierarchy level. A constraint object allows you to link objects anywhere in the object hierarchy with each other and control them using scripts.

Attaching a script to an object

You can attach scripts to objects through the property window's Script tab. For example, to control the color of an object using its translation attribute:

1. Create a sphere, set the Command Language in the property window's Script tab to JavaScript and enter the following script:

```
translate = Self.GetTotalTranslate();
Self.SetColor(translate);
```

The script reads the translation attribute of the object and assigns it to the color attribute.

2. Key frame the sphere by moving it in the 'x' direction. Play the animation to see how the color changes when the sphere moves.

Constraining positions

Let's assume you want to keep a sphere falling in a gravity field above the ground plane. Attaching the following script to the sphere does this:

1. Select the sphere and set the Command Language in the property window's Script tab to JavaScript. Enter the following script:

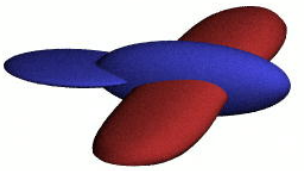
```
center = Self.GetCenter();
if(center.y < 0) {
     center.y = 0;
     self.SetCenter(center);
}
```

Airplane

The following tutorial example shows how to use a constraint object in a fluid dynamics simulation. We build a little airplane which hopefully manages to take off the ground. This is not an easy task, because designing airplanes is demanding work!

Tutorial level: Advanced

Example project: 'Tutorprojects/Simulation/Plane'

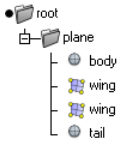


A simple airplane

1. Model a simple miniature airplane (about 10 cm long) as shown in the image. It consists of four components: a body, two wings and a tail. Our sample plane consists of two elliptic spheres (body and tail) and two SDS cubes, which have been edited to a wing-like shape.

2. Select the plane level object. Open the property window, go to the Sim tab and define Gravity = Affected, Fluid Dynamics = Affected.

We will set up a simulation where things happen slower than in a real world. This makes it easier to track the plane when it flies.



3. Select the body and two wings of the plane. Go to the Phys tab of the property window and set their mass to '0.01' (10 grams).

Select the tail and set its mass to '0.002'. The large but light tail will stabilize the plane.

4. Select the body of the plane. Make sure that the Handles/Rotate is set in the view window's popup. Now look at the three move handle lines; the red is object's x-axis, green denotes the y-axis and the blue line is the z-axis. Enter three values to the Fluid Friction field of the Phys tab. The first value is friction in the x-axis direction. If the red line points along the body of the plane, x-friction is very small, 0.01, because the object is most aerodynamic in this direction. If the red axis is not the most aerodynamic axis (but the one between wings or the vertical axis), x friction is slightly larger, 0.03. The other two friction factors are defined the same way. In our sample plane the red axis is the length axis, so the suitable Fluid Friction values are '0.01 0.03 0.03'.

5.Select the wings and define the Fluid Friction the same way as you did for the body. A wing has small fluid friction in two directions, and high friction against the flat wing plane. The orientation of the wings may be different from the body, so check again the red-green-blue orientation of the wing's rotate handle when defining the three fluid friction factors. In the sample plane, the green axis is the vertical axis. Hence the suitable friction factors are '0.01 0.1 0.01'.

6. Select the tail and define Fluid Friction for it ('0.01 0.01 0.1' for the sample plane).

7. Next we define symmetry of airflow around the wings. Asymmetric airflow will create a lifting force. Fluid Symmetry field controls the desired property. A factor '0 0 0' means that air flows around the object in a perfectly symmetric way. In our sample plane, the green object axis is the vertical axis i.e. the direction in which the wings lift the plane. Therefore, we define a Fluid Symmetry value '0 0.7 0' for both wings.

8. Select the complete plane object. Go to the Simulation tab of the toolbar, activate the Velocity tool and draw a short horizontal velocity line for the plane.

9. Activate the Distant Gravity tool and draw a vertical line pointing downwards. This adds gravity to our simulation. The plane has to fight against the gravity to take off the ground. Since our animation happens slowly and in small scale, go to the Spec tab and decrease the Strength field to '0.02'. Now objects fall slowly, and the plane does not have to travel 500 km/h to take off.

10. Activate the Fan tool and draw a fan object in the view window. The fan adds fluid friction to the simulation. To make the simulation interesting, let's use turbulent wind: go to the Spec tab of the property window and set Turbulence of the fan to the full level of 1.0. The direction of wind now changes randomly in space. Go to the Phys tab and decrease Fluid Velocity to 0.2 m/s. Slow enough turbulent wind gives our plane a change to stay in the air for a while. Increase the Density value to 10. Thick air helps to stabilize the plane.

At this point, the plane already tries to take off the ground thanks to its initial velocity. However, it will fall down quickly, because air resists its motion and it has no engine pushing it forward. We will use scripting to add a suitable force, which pushes the plane forward.

11. Activate the Distant Gravity tool again and add another gravity field to the scene. Rename the force object as 'push'.

12. Go to the Spec tab of he property window and adjust the strength of the new gravity object to 0.2 (0.2 N means that the engine is very weak, but the plane weighs only some tens of grams and the actual gravity is also small).

Now the problem is that the new force has a fixed direction. It should always push to the direction of the plane.

13. The push object selected, go to the Script tab of the property window. Set Command Language to JavaScript. Type the following script (remember to hit Enter after the last line):

```
body = Self.FINDBYNAME("body");
center = body.GetCenter();
to = body.GetA();
to.fadd(center);

Self.SetPosition(center);
Self.SetDirection(to);
```

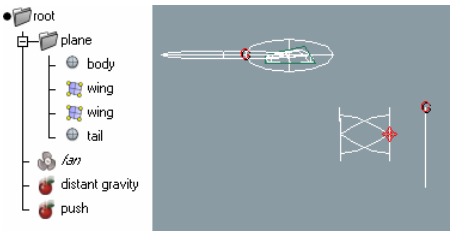


Note

The JavaScript language and the attribute names are case sensitive.

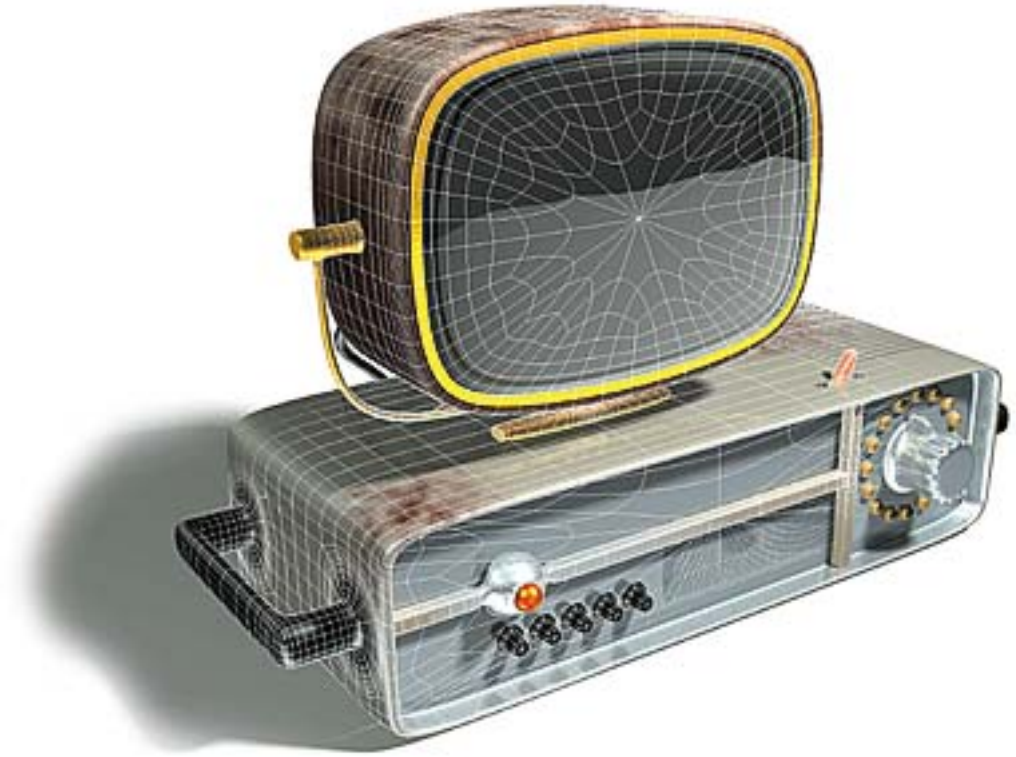
The script above reads the position of the elliptic sphere and writes it to the end point of the push force direction line. Then it sums the 'A' axis of the sphere to the center and assigns to sum to the start point of the force. Note that the orientation of the plane body may be different in your scene: if a green translation handle points the length axis of plane body, then use 'B' instead of 'A' (and 'C' if blue axis shows the right direction). The force orientation may also become opposite to the plane direction, but this can be fixed e.g. by rotating the 'body' ellipsoid 180 degrees (or by changing 'F+' to 'F-' in the script above).

You can also use other attributes of the plane for defining the force direction. The important idea is that the script copies a suitable direction from the plane to the 'push' force. You can see the list of available attributes e.g. by using the 'Input' tab of the choreography window (see the animation documentation for further details).



The plane simulation scene finished

The scene is now ready. Set the frame count to 200-300 frames and play the animation. With some luck, the plane manages to take off and make a beautiful flight. If not, you can decrease gravity and turbidity, increase the size of the tail, balance the plane better, etc. Since the plane has no pilot, it is doomed to crash sooner or later.



Chapter 11. User Interface

Customizing the Toolbar

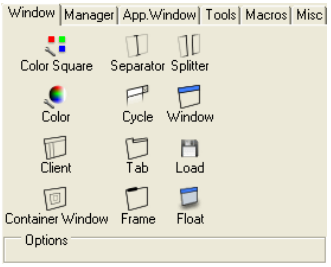
The toolbar in Realsoft 3D is user configurable. You can add new tools into the toolbar, change the order in which the tools appear and delete existing tools.

1. Select the pull down menu Customization/Available Objects Window.

Note that the contents of this window may vary depending on the version and installed modules.

2. Go to the Tools tab and open some tool frames to find the actual tool icons.

3. Drag & drop the desired tools into suitable locations of your toolbar.

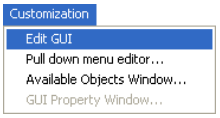


Available Objects Window

Removing tools from the toolbar

You can remove any tool from the toolbar as follows:

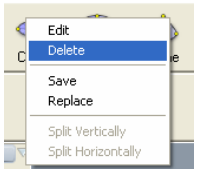
1. Select the pull down menu Customization/Edit GUI.



Activate the user interface editing mode

2. Move the mouse over the desired tool and click the right mouse button to open the pop-up menu. Select Delete from the menu.

3. When ready, exit the GUI editing mode by selecting the menu Customization/Edit GUI again.



Delete a tool from the toolbar

Changing the order of tools

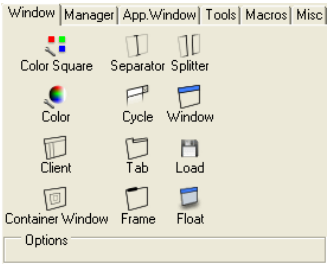
In the GUI editing mode, you can use drag & drop to move a certain tool of the toolbar into another position. However, you can't move or relocate tools away from the toolbar, i.e. tools can exist only in a toolbar.

Customizing the User Interface

Realsoft 3D contains an advanced drag & drop based GUI development system. It allows customization of existing user interfaces, as well as creation of completely new ones.

The 'Customization/Available Objects Window' pull-down menu opens a window, which contains all installed user interface elements.

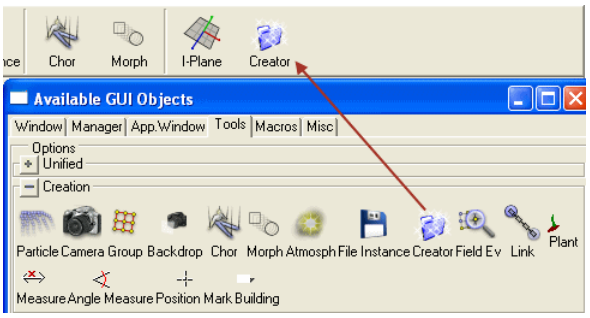
Note that the contents of this window may vary depending on the version and installed plug-in modules.



Available GUI Objects

Adding tools to toolbars

The 'Tools' tab in this window includes all the installed tools. Simply drag & drop the desired tools into your toolbar.

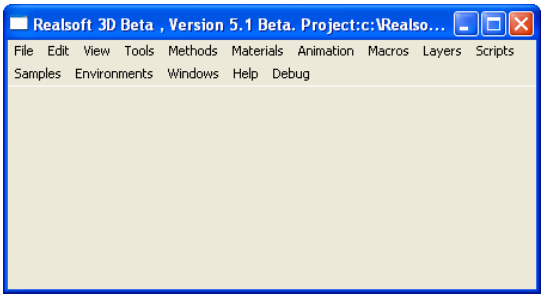


Drag & drop new tools into your application toolbar

Creating new user interfaces

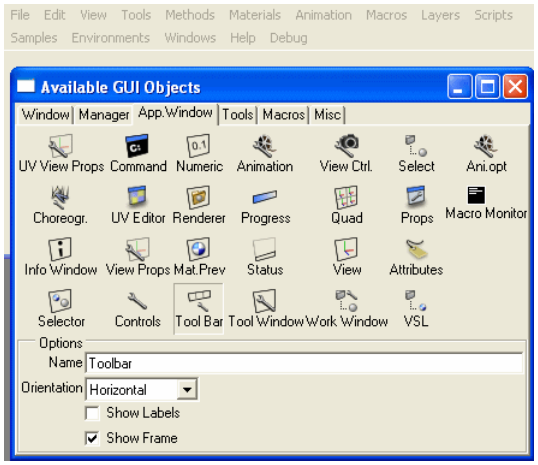
1. Start the program with a new 'blank' project by clicking the 'Blank' project icon of the 'environments' folder, or replace the current environment using the pull down menu 'Environments/Blank'.

2. Select the pull down menu 'Customization/Available Objects Window'. Some of the objects in this window, such as the 'Animation' object, do not define any options and can be directly dropped into the actual main window. Some of the objects, such as the 'Macro' button, require definition of further options, such as the 'macro name', before they can be dragged into the main window.



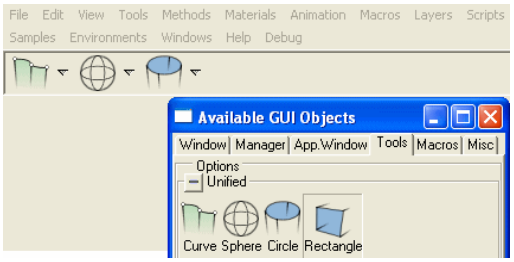
An empty GUI

3. Go to the 'App.Window' tab and click the 'Tool Bar' icon. The lower part of the window now shows you some GUI object specific options. Set the 'Show Frame' option and drag the 'Tool Bar' icon into the 'blank' main window.



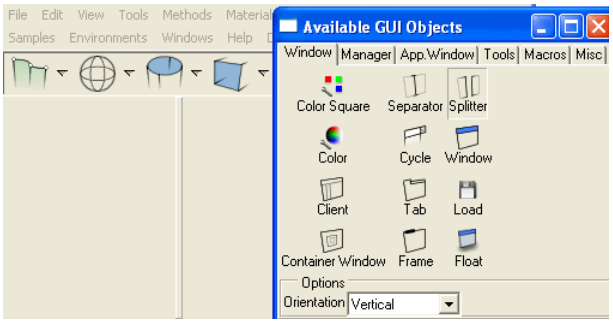
Inserting a toolbar to the GUI

4. Go to the 'Tools' tab of the Available Objects window and open the 'Unified' frame. Drag the 'Sphere' icon into the toolbar. Continue dropping tools until the toolbar contains all the tools you need.



Adding a tool into the toolbar

5. Click the 'Splitter' icon of the 'Windows' tab of the Available Objects window. Again, the lower part of the window is changed to reflect the splitter specific options. Set 'Orientation' to 'Vertical' and drag & drop the splitter into the main window.

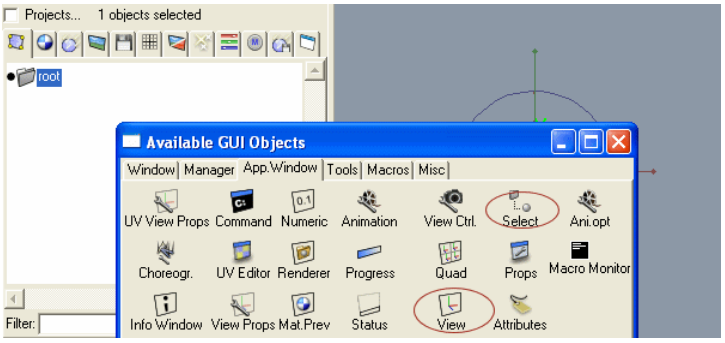


Adding a splitter

6. Go to the App.Window tab and drag the Select object to the left side of the splitter and the View object to the right side of the splitter.

7. When the GUI is ready, close the Available Objects window. To save the created user interface, select the Environments/Save Environment menu and enter a desired name for the file.

You have now created a new fully working user interface and you can use it right away! For example, to create a sphere, simply click the sphere tool.



Select and View windows added

Geometry managers

When creating new windows, the first thing you have to do is to drop some sort of *geometry manager* into the window. If you do not do this, one is created for you automatically, as soon as the first GUI object is dropped into the window. Geometry managers are responsible for managing controls in the window, keeping them correctly positioned, aligned and sized according to gadget sizes, the current font and other requirements.

There are currently three geometry managers available:

1. **Packer**. This packs all dropped objects either horizontally or vertically. This is a frequently needed geometry manager.

2. **Row Column**. This arranges its child objects into rows and columns.

3. **Placer**. This manager positions children using relative coordinates.

Once you have selected the desired geometry manager and defined appropriate attributes for it (horizontal/vertical, etc.), you can drag & drop it into the application window.

Then you can drop the actual GUI objects in, and they will be aligned automatically by the geometry manager of the window.

Editing user interfaces

In the previous example we created a customized user interface from scratch. However, we also have to be able to delete GUI objects from the user interface, as well as modify their attributes (such as the orientation of a toolbar).

This can be done in the 'Edit' mode.



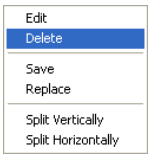
To enter the GUI editing mode, select the pull down menu 'Customization/Edit GUI'. You will no longer be able to use the interface. Instead, you will be able to rearrange the controls in the interface by dragging them.



Drag & drop an icon to a new position

To delete a GUI object from the interface, move the cursor over the object (the frame will be highlighted) and click the right mouse button. This opens a GUI object specific pop-up menu. Select 'Delete' to delete the object.

Select the pull down menu 'Customization/Edit GUI' again to leave the edit mode and return to normal.



Deleting a GUI object

Binding Macros to Keys

The default startup project (startup.r3d) defines a set of default key bindings. For example, when you hit the 'm' key, the move tool is activated.

The key binding system of Realsoft 3D implements this behavior. This system contains a list of nodes, each defining two things: a key combination and a macro. Whenever you hit a key, the key binding list is scanned. If a node matching the pressed key is found, the corresponding macro is executed.

Macros can perform various useful tasks. They can activate a tool with the desired options. They can include a sequence of actions. They can rotate the view window, etc.

Defining key bindings using drag & drop

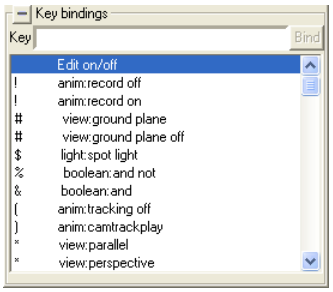
Assume that we want to create lots of NURBS curves using the 'Freehand' option. Activating the curve tool and setting the freehand option is not the fastest way to get the curves created. To solve the problem, you can record a macro, which activates the curve tool with the specified options, and bind the macro to a desired key.

Let's see how this can be achieved.



Note

We suggest that when creating new key shortcuts in the following tutorials, always use the Alt key as a modifier to avoid confusion with the default key bindings. The Alt modifier has been reserved for the user's own shortcuts.



The 'Key bindings' frame of the select window's Named Macros tab

1. Go to the select window's Named Macros tab and open the 'Key Bindings' frame.

2. Click the 'Key' field and enter the desired key combination, for example 'Alt f'.



Key 'Alt f' entered into the 'Key' field

3. Activate the curve tool and set the 'Freehand' option. Then press the Ctrl key down and drag & drop the curve tool icon into the binding list of the 'Key Bindings' frame.

A new named macro with the name of the dragged tool is automatically created and bound to the specified key. Whenever you hit the key, the curve tool with the freehand option becomes activated.

Defining key bindings manually

In addition to the drag & drop interface, you can use a manual method.

For example, to define the 'freehand curve' binding manually:

1. Set 'Macro/Record' on

2. Click the NURBS/Curve tool and set the 'Freehand' option.

3. Reset macro recording and cancel the NURBS/Curve tool.

You have now recorded a macro. You can verify that the macro works as expected by hitting the 'Ctrl + g' key (which executes the current macro). The NURBS/Curve tool becomes active. Cancel the tool after testing that it works as expected.

4. Go to the Named Macros tab of the select window. Open the 'Named Macros' frame. The 'Name' field contains a string 'current'. All new macros are named as 'current' by default. Activate the 'Name' field and rename the recorded macro to a suitable new name, such as 'freehand curve' and press the enter key. The name of the macro is added to the list (alphabetically sorted) of named macros.

The next step is to bind the new named macro to a suitable key. Make sure you don't have any geometric objects selected. If you have, deselect them.

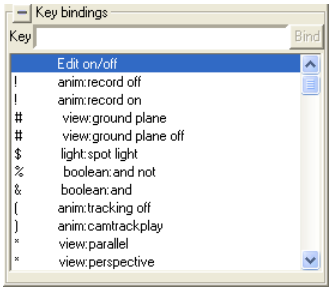
5. Open the 'Key bindings' frame and click the 'Key' field, enter a suitable key combination (in this case the ALT + f key) and then click the 'Bind' button. The key combination is added to the list, which is alphabetically sorted by key command.

This is exactly what happens when defining bindings using drag & drop.

Context sensitivity

Realsoft 3D key binding system supports context sensitivity. This is implemented using *modifiers*. For example, if you hit the 'f' key, a named macro called 'forward one frame' is executed. However, if you hold down the Alt key while hitting the 'f' key, another macro (say 'freehand curve') can be executed.

In addition to the standard keyboard modifiers, such as 'Ctrl' or 'Shift', Realsoft 3D automatically defines modifiers such as 'the type of selected objects' or 'name of the current tool'. This increases the flexibility of the binding system greatly: You can define key bindings, which are executed only when a certain tool is activated or a certain type of object is selected.



Key bindings

You can define a context sensitive macro in the following way:

1. Go to the select window's Named Macros tab and open the 'Key Bindings' frame.

2. Make sure there are no geometric objects selected. If you have activated a tool, cancel it.

3. Activate the Key field in the Key Bindings frame by clicking it with the mouse and hit the Alt and the f key. The Key field shows you '<alt>f'.

4. Now hold down the Ctrl key and hit the 'f' again. The 'Key' field now shows you a string '<ctrl>f'. In other words, <ctrl> is a modifier for the 'f' key.



'Alt f' entered into the Key field

5. In addition to the standard keyboard modifiers, such as Ctrl, Shift and Alt, Realsoft 3D allows you to use any key as a modifier for another key. For example, hold down the 'q' key while hitting the 'f' key. The result is a string '<q>f'.

6. Now activate the Move tool of the toolbar (it's in the Transformation tab), click the 'Key' field and hit the 'f' key. The result is a string '[Move]f'. A macro bound to this key string will be executed by the 'f' key only when the Move tool is activated.

7. Finally, select the 'root' object in the Geometric Objects tab of the Select window and return back to the Named Macros tab. Now click the 'Key' field, hold down the Ctrl key and hit the 'f' key. This produces a combination '[Move][Level]<ctrl>f'. A macro bound to this combination is executed only when the Move tool is activated, a level object is selected, Ctrl is pressed and the 'f' key is hit.

The actual key binding system is simple to describe. When you hit a key, the system attempts to find a key binding, which matches the key string produced by currently set modifiers (such as '<ctrl>f'). If found, the associated named macro is executed.

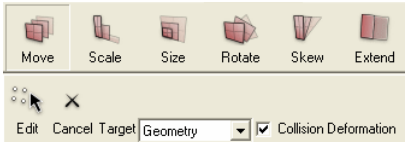
Note that keyboard modifiers are closed inside angle brackets '<>' whereas *current tool* and *selected object* modifiers are enclosed in normal brackets '[]'. Modifiers enclosed with '[]' are optional. If no binding is found, optional modifiers are removed one by one to check matches with more general bindings. For example, if you hit the 'p' key while a 'sphere' object is selected, the binding '[sphere]p' is searched first. If not found, a plain 'p' binding is searched. This may sound complicated, but it allows you to define very powerful and flexible context sensitive key bindings. Let's see how this all works in practice.

Move with collision detection

Let's imagine that you need the Move tool with the 'Collision Detection' option very often, so that you want to define a key binding which allows you to activate the tool efficiently.

1. Activate macro recording.

2. Click the Move tool and set the 'Collision Deformation' option.

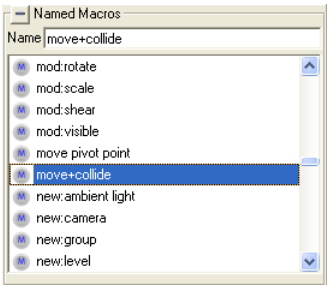


Record a macro, which activates the Move tool and sets its Collision Deformation option

3. Disable macro recording and cancel the Move tool. You have now recorded a macro, which activates the Move tool and sets its Collision Deformation option.

4. Go to the Named Macros tab in the Select window, open the 'Named Macros' frame and rename the 'current' macro as 'move+collide' and press the Enter key.

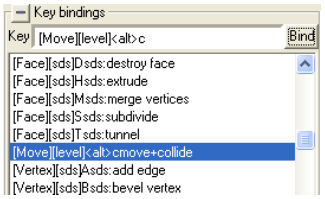
5. Make sure you don't have any objects selected (we don't want to create an Object sensitive binding this time). Also, make sure the Move tool is canceled (we don't want to create a Tool sensitive binding). Then click the 'Key' field of the 'Key bindings' frame, press Alt down and hit the 'c' key. Then click the 'Bind' button.



Rename 'current' as 'move+collide'

You have now bound the 'move+collide' macro to the '<Alt>c' key, and you should see the string '<Alt>c move+collide' in the key bindings list.

6. Create a couple of SDS spheres, select one of them and hit the '<Alt>c' key to activate the move tool with collision deformations.



'Move+collide' macro bound to '<alt>c' key combination

Tool sensitive bindings

Let's assume that you create curves a lot and you often need to use the 'Closed' option of the NURBS Curve tool. So, let's define a key binding which allows you to activate the 'closed' option for the NURBS curve simply by hitting the key '<Alt>c'.



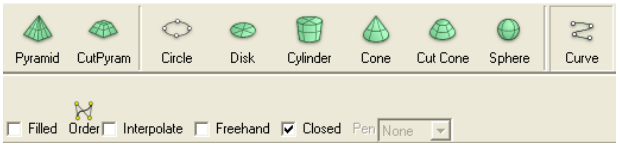
Note

The idea here is that '<Alt>c' should set the 'closed' option only when the curve tool is activated. Otherwise it should activate the move tool with collision detection as defined in the previous example.

1. Activate the NURBS Curve tool. The control bar now shows you the curve specific options.

2. Set macro recording on. Note that macro recording is activated AFTER starting the curve tool.

3. Set the 'Closed' check box.

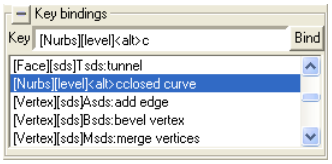


Record a macro which sets the 'Closed' option for the NURBS curve tool

4. End macro recording. You have now recorded a macro, which just sets the 'Closed' option.

5. Go to the Named Macros tab of the Select window. Open the 'Named Macros' frame and rename the recorded current macro as 'closed curve' in the 'Name' field and press the Enter key.

6. Without canceling the NURBS curve tool, open the 'Key bindings' frame and click the 'Key' field. Now enter the combination '<Alt>c'. You should now see the string '*[Nurbs]<alt>c*' string in the 'Key' field. Click the 'Bind' button.



'Closed curve' macro bound to [Nurbs]<alt>c key

Now, cancel the curve tool and activate it again. The closed option is not set by default. Use the '<Alt>c' key combination to get the 'Closed' option set.

Note that the 'Move+collide' binding still works provided that the curve tool is not selected.

Context sensitivity and edit mode

The key binding system is also sensitive to the editing mode of the selected objects. For example, you can bind the 'b' key to the 'Break' tool whenever a NURBS curve is selected and is in the Edit mode. If the selected curve is not in the Edit mode, the 'b' key can execute another tool, for example the Bend tool.

The context sensitivity is implemented by concatenating the 'edit mode' to the key string as a modifier.

Some objects can define several edit modes. For example, SDS objects have three edit modes: point, edge and face. Each mode can have its own key bindings. For example, 's' can activate 'subdivide' in the Face editing mode. In Edge mode, the 's' can activate the 'Split edge' tool.

Another example is a set of bindings to the Tab key, which cycles through all edit modes of an SDS object. This can be achieved by defining the following four bindings:

	[sds]	***Tab***	***sds:edit points***
[Point]	*[sds]*	*Tab*	*sds:edit edges*
[Edge]	*[sds]*	*Tab*	*sds:edit faces*
[Face]	*[sds]*	*Tab*	*sds:object*

In other words, when you hit the 'Tab' key when a SDS object in 'object' mode is selected, the Edit Points mode is entered. If you hit the Tab key again, the macro 'sds:edit edges' is executed, and so on.

Multiple key bindings

You can bind more than one macro to the same key combination. In this case, Realsoft 3D cycles through the macros.

Let's assume you need to switch between smooth and polygonal SDS objects and you would like the '<Alt>p' key to toggle the polygonal option on and off. This is naturally a SDS specific key binding: '<Alt>p' should only toggle the polygonal option when a SDS object is selected.

The default startup project defines these macros already so we don't repeat the recording steps here.

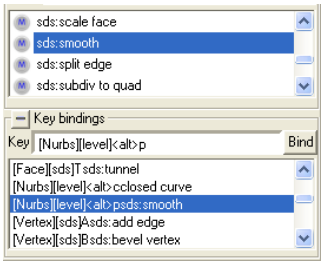
1. Create a SDS sphere and select it (we want to define a SDS specific binding).

2. Go to the Named Macros tab of the Select window and open the 'Named Macros' and the 'Key bindings' frames.

3. Select the 'sds:polygonal' named macro, click the 'Key' field and hit the key 'p' while holding the Alt key down. Then click the 'Bind' button.

4. Select the sds:smooth' named macro and click the 'Bind' button again.

You have now bound two macros to a single key.



Two macros bound to a single key

5. Hit the key '<Alt>p' a couple of times while the SDS sphere is selected. The sphere surface toggles between polygonal and smooth shapes.

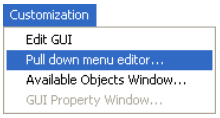
Any number of macros can be bound to a single key this way. The key binding system remembers the last executed binding and next time you hit the same key, the next macro in the macro list is executed.

Saving/loading key bindings

Key bindings are saved with projects. They are included in the Macros section of Realsoft 3D files. You can use the pull down menu File/Save as Startup to save the current project with the modified key bindings to your startup file.

If you want to save only the macros and key bindings, select File/Save As and set the Macros section. This saves both the named macros and the key bindings.

Binding Macros to Pull Down Menus

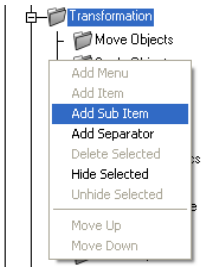


Open the pull-down menu editor

The user can customize the pull down menu strip. Any number of new menu items can be inserted to it. When a menu is selected, the associated macro is executed.

Let's practice this by adding a pull down menu, which activates the Move tool.

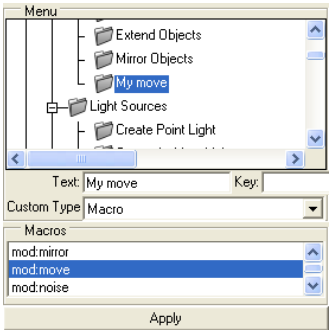
1. Select the pull down menu Environments/Customization/Pull down menu editor.



Add a new menu item into the Tools/Transformation menu

2. We will insert the new menu item under the Tools/Transformation menu. So, locate and select the Transformation menu item from the opened menu tree view. Select 'Add Sub Item' from the menu editor's popup menu.

This adds a new node 'New Sub Item' into the end of the menu list of the Transformation node.

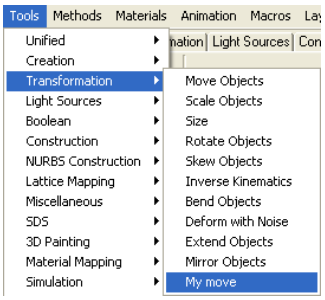


A macro 'mod:move' bound to a 'my move' pull down menu

3. Locate the node and rename it to 'My move' using the 'Text' field. Then select the 'mod:move' macro of the named macro list in the bottom of the menu window. You have now created a new menu item and bound it to a macro.

4. Click the 'Apply' button in the menu window. The specified menu is now inserted to the actual menu strip.

Close the menu editor and try selecting the new menu - the macro 'mod:move' is executed, activating the move tool.



A new pull down menu 'My move' inserted to the Transformation menu



Note

The menu strip is saved with the project. Save the environment to a suitable file to keep the customized menu permanently. (Use the 'File/Save as Startup' pull down menu to save the menu strip to your startup file. Note that this overwrites the startup file.)

Binding Macros to Buttons

Some of the Tools in Realsoft 3D are very general. For example, the Analytic tool creates all kinds of analytical objects.

The good thing is that you can easily find the tool: no matter if you are creating an analytic sphere, a cube, a cut pyramid or a cylinder, just click the analytic tool and the program makes all analytic specific options available for you.

However, if you just want to create certain types of objects, you may prefer a simple button, which selects suitable tool options with one click.

You can construct custom tool palettes using macro buttons, which activate tools with suitable predefined options.

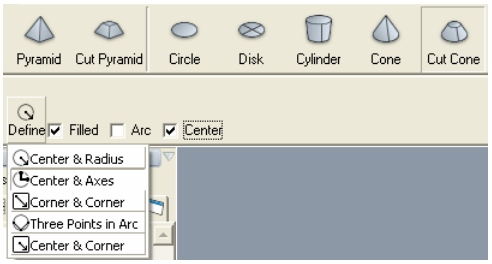
Using drag & drop to create macro buttons

The drag & drop interface provides you with an easy way to insert macro buttons into the user interface.

1. Activate the desired tool and specify desired options.

2. Press the Ctrl key down and drag & drop the tool to a separate window, into another position in the toolbar, etc. to create a 'short-cut' to the tool.

Whenever you click this 'shortcut' button, the original tool is activated with the specified options.



Activate the desired tool with desired tool options

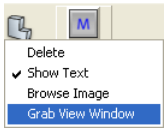
The created macro button uses simple image which shows red 'M' letter. You can find this default image from the 'icons' folder.



Macro button created from the CutCone tool

3. Render nice image for the tool button using the View window and select the Macro button's pop-up menu 'Grab View Window'. This saves the contents of the current view window and attaches the saved image to the macro button.

You may also select any existing image for the macro button by selecting the 'Browse' menu item.



Grab View tool

Creating macro buttons manually

You can drag & drop any number of macro buttons from the Available Objects window into the current user interface. Each macro button can define an icon, a label, a tool tip and a named macro to be executed when the button is clicked.

1. Select the Customization/Available Objects Window pull down menu.

2. Go to the Macros tab of the opened window and click the Macro icon. The macro button specific options are shown.



Macro icon

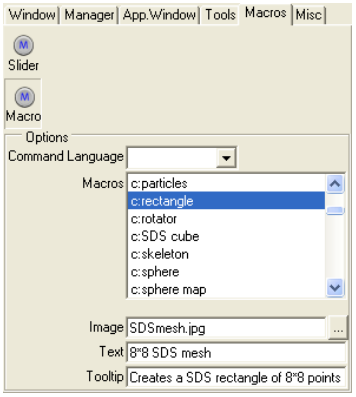
3. Select a macro to be executed from the 'Macro' list. Whenever the button is clicked, the selected macro is executed. You can also Click 'Browse' to locate a desired image for the button. If you don't specify an icon, a default icon will be used. Any image format that Realsoft 3D supports will work (such as JPG, BMP, TGA, GIF, etc.). You can also enter the desired 'Text' and 'Tool tip' strings for the button.

4. When done, drag & drop the macro icon from the Available Objects window into the desired position in the actual GUI (such as a toolbar).



Note

Use File/Save As or Environments/Save environment to save the new customized GUI.



Macro Button specific options defined

Binding Macros to Compass Menus

You can create any number of compass menus.

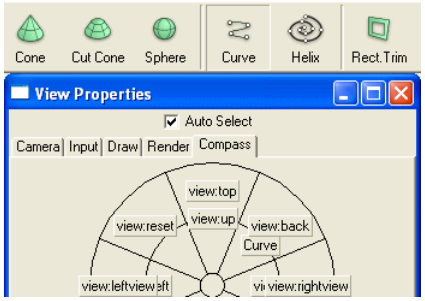
Compass menus are context sensitive in the same manner as key bindings: the contents of the current compass depends on the selected object, current tool and keyboard modifiers. For example, you can create a compass, which is activated when you hold down a certain key while dragging with the right mouse button.

Each menu item in the compass is bound to a named macro, which is executed when the item is selected.

These tutorials show you how to create context sensitive compass menus and how to bind macros to them.

You can easily set up a new compass menu by simply dragging desired tools into it. For example, to add a tool 'create freehand closed NURBS curve' to the default compass menu.

1. Open the view property window and go to the Compass tab. Select the first compass from the Compasses list (Compass).



Select a tool, set specific options and drag & drop the tool into the compass editor

2. Activate the NURBS/Curve tool and set the Closed and Freehand options. Then press the Ctrl key down and drag & drop the Curve tool button into the desired position of the compass menu.

A sphere specific compass

Let's assume you are working in a company designing ball bearings and you frequently need to duplicate a sphere and move it in the 'x' direction.

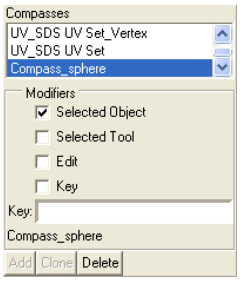
You can increase your productivity by creating a compass menu, which is automatically used when an analytic sphere is selected. It allows you to duplicate and move the selected spheres by using one single operation.

We start by recording a named macro for this purpose.

1. Create an analytic sphere and record a named macro 'mymacro' which duplicates and moves the sphere (If you don't remember how to record macros, check out the macro tutorial in the Modeling chapter).

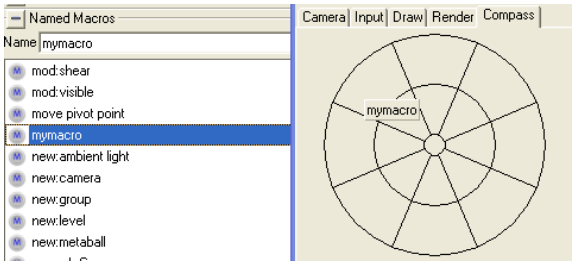
2. Select View Properties from the view's pop-up menu and go into the Compass tab. In this tab you can manage compass menus.

3. Make sure the analytic sphere you created is selected and set the 'Selected Object' check box. Then click the 'Add' button. A new sphere specific compass is created.



A sphere specific compass created

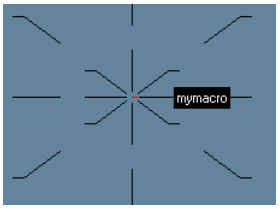
The new compass is shown whenever you drag with the right mouse button and asphere is selected. You can test this in the view window immediately (just drag with the right mouse button in the view window to see the empty compass menu). 4. Drag & drop the recorded 'mymacro' from the 'Named Macros' list of the select window's 'Named Macros' tab into a desired position of the compass.



A compass menu consisting of one menu item

The compass is now ready. You can close the view property window.

5. Select an analytic sphere and drag with the right mouse button in the view window to open the sphere specific compass menu. Select 'mymacro' in the compass menu and the macro is executed.



Select a sphere and drag with the right mouse button in the view window

A key sensitive compass

In addition to selected objects, you can also create keyboard, tool and edit state specific compasses.

For example, you can create a compass, which is opened whenever you drag while holding down the 'F1' function key.

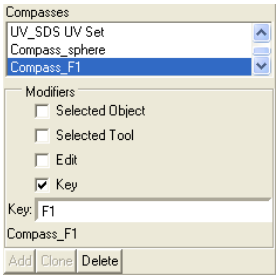
To do this:

1. Go to the 'Compass' tab of the view property window as described in the previous example.

2. Set the 'Key' check box, click the 'Key' string fieldbelow it and hit the 'F1' key. You can now see the string 'F1' in the key field.

3. Click the 'Add' button. This creates a compass called 'Compass_F1'.

4. Drag & drop the suitable macros (or tool icons) to the compass.



F1 Compass created

Deleting and editing compass menus

You can change and delete compass items using the 'Compass' tab of the view property window.

If you want to change a macro binding of a certain compass area, simply drag & drop a new macro into the sector in question.

To delete a named macro, click the right mouse button over the desired compass sector. This opens a pop-up menu, which includes a tool for deleting the macro binding (the actual macro is not deleted).

The Clone and Delete buttons near the bottom edge of the Compass tab duplicate and delete entire compasses.

Localization

The user interface of Realsoft 3D is completely dynamic. Any part of if can be removed, new windows can be inserted.

The dynamic implementation also includes strings and labels associated with graphical controls: such as sliders, buttons and check boxes. All strings can be replaced, translated to another language, abbreviated, etc. and the interface automatically adapts to the new string lengths.

The 'locale' directory contains all strings of the program. The folder includes one file for each Realsoft 3D module. There is a sub folder for each language. The folder named 'english' contains all the built-in strings in the program.

To support new languages, say swedish, find this 'locale' folder in the home directory of Realsoft 3D. Create a new directory under the 'locale' directory. For example 'swedish'.

3. Copy the default string files from the 'english' directory into the newly created directory. Translate the strings in the copied .ini files to the new language. Translate only the strings after the '=' character on each line. The strings before the '=' character are used for identifying the strings and should not be changed.

4. Open the StartOpts utility program and set language to the new language, say swedish.



Note

The string files in the 'locale\english' directory are automatically generated from the source code and should not be modified.

You can also select the localization with a startup parameter as follows:

```
realsoft3d -lang swedish
```

Localizing user defined strings

In addition to built-in strings, the end user may create new user interfaces, such as windows and tool bars. Strings in these user defined display elements can be translated as well.

Localizing Tab labels

Tab labels can be defined in the r3wid2.ini, under class [r3guitab].

For example:

```
[r3guitab]

SDS=Subdivision
```

will change the 'SDS tool tab to 'Subdivision'.

Localizing Window names

Window names (strings you see in the Windows pull-down menu), can be localized in r3wid2.ini, under the class [r3drfwto].

For example, to change the 'Property Window' to 'Properties':

```
[r3drfwto]

Property Window=Properties
```

Localizing Docked Window names

Docked windows, such as 'Advanced Search' or 'Animation Settings' can be localized through the r3wid.ini file, under the class [r3clnwin].

For example, to change the 'Animation Settings' title to 'Anim. Options':

```
[r3clnwin]

Animation Settings=Anim. Options
```



Chapter 12. Scripting

Introduction to Scripting Languages

Scripting languages (or macro languages) can be used for the following tasks:

- Create and edit geometric and other objects through a command line interface.
- Expand and customize the user interface. For example, you can create a slider, which controls the bone angle of a skeleton.
- Write custom material effects for photorealistic rendering.
- Define *constraints* between objects. For example, the position of an object can be defined as a function of two other objects.
- Create intelligent behavioral animations in which procedures written in macro languages control how objects behave.
- Expression evaluation. The numeric interface for creating and modifying objects, for example, allows you to use expressions to define properties for objects. When creating a sphere, you can fetch information from other objects using expressions like (*sphere1.center.x + sphere2.center.y)/2.0*.

The structure of Realsoft 3D is 'object oriented'. Correspondingly, scripts written by the user are always associated with some sort of an object.

Controlling Geometric Objects

Let's create a sphere whose position in the 'y' direction is fixed to 0.0. In other words, the sphere is not capable of moving in the up/down direction.

1. Create an analytic sphere.

2. Select the sphere, open the property window and go to the Script tab.

3. Select JavaScript from the Command Language control and enter the following text into the text field:

```
center=Self.GetCenter();
center.y = 0;
Self.SetCenter(center);
```

Now, try to move the sphere with the mouse. You can move it horizontally but not up or down!

Let's create a simple animation. Try the following script:

```
center=Self.GetCenter();
center.y = Time;
Self.SetCenter(center);
```

On other words, the 'y' position of the sphere depends on the time. Play the animation to verify this.

Binding Scripts to GUI Objects

Let's assume you have to create animations so that there is a key frame in every 5fth frame.

1. Open the Environments/Customization/Available Objects window.

2. Go to the Macros tab. Click the Macro button. The lower part of the window shows you now the macro button specific options.

3. Set the Command Language field to JavaScript. You can now enter the desired program into the text control. Enter the following text:

```
layerlist = GetJS("");
currentproject = layerlist.GetCurrentLayer();
animator = currentproject.GetAnimator();
frame = animator.GetCurrentFrame();
frame = frame + 5;
animator.SetCurrentFrame(frame);
```

4. Enter an appropriate name and a tool tip for the button (such as 'Jump 5'). You might also want to paint a nice icon for the button using your favorite painting program. Then select it using the Image browse button of the macro button controls.

5. Drag & drop the macro button icon into the toolbar.

Every time you click the macro button, the animation time is increased by 5 frames.

JavaScript

Introduction

JavaScript provides you with a sophisticated, object oriented scripting interface to Realsoft 3D.

These tutorials describe how to use JavaScript to access Realsoft 3D functionality. They are intended for programmers who are already experienced at using JavaScript or some other object oriented programming language.

JavaScript is linked to Realsoft 3D through set of wrapper objects. There is a JavaScript object for each Realsoft 3D object, allowing you to access the functionality of Realsoft 3D through native JavaScript interface.

It should be emphasized that JavaScript interface was designed to provide you with total control over Realsoft 3D. It allows you to customize and expand the functionality of the program. Therefore, the structure of the JavaScript interface corresponds to the structure of Realsoft 3D application, as defined in the Realsoft 3D Software Development Tool Kit.

It is also important to understand that JavaScript objects and Realsoft 3D objects are not the same thing. Realsoft 3D objects can exist with or without JavaScript.

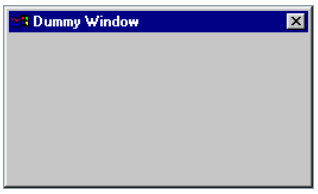
Creating User Interface Objects

Creating a window

You can create a window by calling:

```
include("oops/r3window.js");
win = new r3Window(R3WGA_Parent, _r3gui)";
win.REALIZE();
```

This creates an empty window with default size, position and other attributes.



Empty floating window

The first call.

```
include("oops/r3window.js");
```

loads in JavaScript interface to Realsoft 3D Window objects, if not already loaded.

The second call:

```
win = new r3Window(R3WGA_Parent, _r3gui);
```

creates a window object. Parameters for the construction function specify properties for the window to be created. Each property is defined by a tag-value pair. For example:

```
R3WGA_Parent, _r3gui
```

defines the parent window.

All properties for the r3Window class can be found in 'scripts/js/oops/r3window.js header file, or one of the files from which the r3window object is derived.

There are two ways for setting desired attributes: by passing them to the constructor function as a tag list, or by setting them by calling the appropriate Set method.

For example, r3window.js defines attribute id 'R3WA_Title'. This attribute can be set by calling the 'SetTitle' function (just replace the 'R3WA_' prefix with 'Set' to get the name of the function.

```
include("oops/r3window.js");

win = new r3Window(R3WGA_Parent, _r3gui);
win.SetTitle("My Window");
win.REALIZE();
```

Another way to create a window with desired title is by passing the attribute id, with associated value to the constructor function:

```
include("oops/r3window.js");

win = new r3Window(R3WGA_Parent, _r3gui, R3WA_Title, "My Window");
win.REALIZE();
```



Note

Realsoft 3D defines variable '_r3gui'. This variable refers to the Realsoft 3D main window. The R3WGA_Parent tag must always be given to the constructor function because it is not possible to create a window without specifying the parent for it. This is true of all other user interface specific objects, such as sliders and buttons, to name a few.

The third call:

```
win.REALIZE();
```

makes the window visible to the user. You have to call this method whenever you create a new window object. This is also true for all objects derived from the r3Window object.

Geometric managers

Realsoft 3D provides you with an advanced toolset for designing user interfaces. The system uses *geometric managers* to take care of managing layouts. By letting geometric managers to do the job has many advantages. For example, your layouts will work on any platform. Also, the more complex interfaces you design, the easier it will be to maintain them using geometric managers. Geometric managers allow you to create truly reusable user interface code.

You will be able to change gadget texts, icon sizes, insert new controls into your windows etc. and the layout is automatically updated for you.

Packer geometry manager allows us to design horizontal and vertical layouts i.e. align controls horizontally or vertically. This is the most often needed geometry manager.

To create a packer:

```
include("oops/r3packer.js");
packer = new r3Packer(R3PA_Orientation, R3PAOF_HORIZONTAL);
```

Then you can ask the window to use the created geometry manager by calling:

```
win.SetGmanager(packer);
```

Anything you insert into this geometry manager gets automatically aligned.

Just like the r3Window class, geometric managers are also derived from the widget base class. However, geometric managers do not have any visible geometric properties.

Creating gadgets

Properties for the button object are defined in the 'oops/r3button.js' header file. You can create a button by calling:

```
include("oops/r3button.js");

myButton = new r3Button(R3WGA_Parent, myWindow,
                        R3GA_Text, "My Button",
                        R3GA_ToolTip, "Click this button");
```



Note

Just like with windows, you must specify the parent window for the button using the R3WGA_Parent tag. In fact, all user interface objects require that you pass the parent object to the constructor function. The underlying operating system is not able to create the control if no parent is specified.

Below you can find some examples for creating other commonly needed controls.

```
include("oops/r3checkb.js");

gad = new r3Checkbox(R3WGA_Parent, myWindow,
                     R3GA_Text, "A checkbox",
                     R3GCBA_Checked, TRUE);

include("oops/r3cycle.js");

gad = new r3Cycle(R3WGA_Parent, window,
                  R3GA_Text, "Quality",
                  R3MXILTGA_Labels, ["Preview","Medium","Good","Best"],
                  R3GA_ToolTip, "Select desired rendering quality");

include("oops/r3radiob.js");

gad = new r3Radiobutton(R3WGA_Parent, window,
              R3GRBA_Labels, ["Circle","Rectangle","Triangle"]);

include("oops/r3slider.js");

slider =  new r3Slider(R3WGA_Parent, window,
                       R3GA_Text, "Level of detail",
                       R3GSLA_Min, 0,       // minimum level
                       R3GSLA_Max, 100,     // maximum level
                       R3GSLA_Level, 20);   // current level
```

Making objects talk to an application

If we executed the above code, it would create a dummy button object. Clicking the button would do nothing. For all practical applications, we need to make GUI objects to talk to us so that we can link functionality to our user interface. This can be done through *callback functions.*

Let's imagine we want to implement a close button i.e. when the user clicks any of the buttons, the entire application is shut down.

To do this, we define a callback function which calls r3Exit() function:

```
function myhook(window, event, value)
{
    r3Exit();
}
```

Then we pass this function to the button constructor by using the R3RA_Hook tag.

```
button = new r3Button(R3RA_Hook, myhook,
                      R3WGA_Parent, window,
                      R3GA_Text, "My button");
```

Whenever the user clicks the above button, the 'myhook' function gets called, which shuts down the program.

You can use callbacks with all the basic Realsoft 3D GUI objects, such as slider gadgets, string gadgets, windows, just to name a few.

Putting it all together

Now that we know how to create windows, packers and buttons, let us create a window with horizontally aligned buttons.

```
include("oops/r3button.js");
include("oops/r3window.js");
include("oops/r3packer.js");

function myhook(window, event, value)
{
    r3Exit();
    return 1;
}

window = new r3Window(R3WGA_Parent, _r3gui,
                      R3WA_Title, "My Window",
                      R3WA_ReportCloseWindow, TRUE,
                      R3WA_ReportNewSize, TRUE);

packer = new r3Packer(R3PA_Orientation, R3PAOF_HORIZONTAL);
window.SetGmanager(packer);

for(i = 0; i < 5; i++) {
  button = new r3Button(R3RA_Hook, myhook,
                        R3WGA_Parent, window,
                        R3GA_Text, "Exit " + i);
  packer.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, button);
}

window.REALIZE();
```

This creates the following window:

In the 'for' loop, the program creates five button objects and inserts them into the packer by calling the 'ADD' method. The parameters for this method specify how the packer will manage the control in question.



A window with horizontal packer.

R3PAPF_EXPAND asks the object in question to request as much space as possible from the packer. If there are multiple controls defining expand option, the packer will distribute the space evenly to all requesters. The R3PAPF_FILLX and R3PAPF_FILLY flags then control whether the object should fill the given screen space horizontally, vertically or both.

If you only specify R3PAPF_EXPAND, but you do not specify R3PAPF_FILL options, the buttons will be distributed over the entire window (i.e. the packer will distribute them more space) but the buttons will not be stretched to fill the extra screen space.

The program also calls the FIT method. By passing the R3WFP_BESTFIT parameter to this method, we ask the window to compute the size needed to make all the created sub controls to fit into the window. The size needed depends on the current font, operating system, and many other properties.

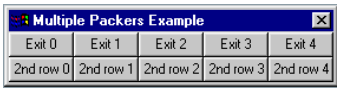
You can find the above sample program from:

scripting/intro/

Creating more complex layouts

Each window can have only one geometric manager associated with it. However, geometric managers can be grouped hierarchically to construct more complex layouts.

For example, we can create a tool bar window consisting of two rows of tools by making the top level packer vertical and then creating two horizontal sub packers for it. The actual tool buttons are then inserted into these sub packers.



A vertical packer two horizontal sub packers.

And here is the code:

```
// create the top level packer
top = new r3Packer(R3PA_Orientation, R3PAOF_VERTICAL);

// create two sub packers
sub1 = new r3Packer(R3PA_Orientation, R3PAOF_HORIZONTAL);
sub2 = new r3Packer(R3PA_Orientation, R3PAOF_HORIZONTAL);

// and insert the sub packers into the top level packer.
top.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, sub1);
top.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, sub2);
```

Then we create and insert desired controls into the sub1 and sub2 packers.

```
// create buttons for the first sub packer
for(i = 0; i < 5; i++) {
    button = new r3Button(R3RA_Hook, myhook,
                          R3WGA_Parent, window,
                          R3GA_Text, "Exit " + i);
    sub1.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, button);
}

// buttons for the second sub packer
for(i = 0; i < 4; i++) {
    button = new r3Button(R3RA_Hook, myhook,
```

```
                            R3WGA_Parent, window,
                            R3GA_Text, "2nd row " + i);
        sub1.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, button);
    }
```

The geometric manager hierarchy can be as deep as you like, providing you with total control over layouts.

Anchoring

In the previous chapters we demonstrated the usage of pack flags (R3PAPF_EXPAND, R3PAPF_FILLX and R3PAPF_FILLY).

The second parameter for the ADD() method specifies so called 'anchor' flags. Whereas R3PAPF_FILLX and R3PAPF_FILLY flags control the size, the Anchor flags control how the widget is positioned within the space it gets from the packer.

Below you can find the list of available anchor codes.

Code	Description
R3PAAF_CENTER	Align the control at the center
R3PAAF_N	North (up)
R3PAAF_NE	North-East
R3PAAF_E	East (right)
R3PAAF_SE	South-East
R3PAAF_S	South (left)
R3PAAF_SW	South-West
R3PAAF_W	West (down)
R3PAAF_NW	North-West
R3PAAF_ALIGN	Align based on alignment points of controls

Unlike pack options, these are mutually exclusive i.e. you can define only one of them.

Because the anchor codes control how the object in question is managed within the space it gets from the packer, it usually makes sense to use these only with R3PAPF_EXPAND option. If you don't define EXPAND, the inserted control may not have a room to move anywhere.

For example, if you insert a control to a packer by calling:

```
    mypacker.ADD(R3PAPF_EXPAND, R3PAAF_CENTER, mybutton);
```

the inserted button will be centered within the space it gets from the geometry manager.

Perhaps one of the most often needed anchor options is R3PAAF_ALIGN, so let's check this out in more detail.

Let's create a window that consists of a number of vertically stacked sliders. The example looks pretty much the same as the first button example we created in the 'Putting it all together' chapter. The only difference is that we now use sliders instead of buttons and use vertical packing rather than horizontal packing.

As usual, we need to include the slider's header file. By studying the header file we can find out the name of the constructor as well as the attributes the slider defines. That's all we need to know in order to create a slider.

Here is the code:

```
include("oops/r3slider.js");
include("oops/r3window.js");
include("oops/r3packer.js");

window = new r3Window(R3WGA_Parent, _r3gui,
                      R3WA_Title, "Align Sliders",
                      R3WA_ReportCloseWindow, TRUE,
                      R3WA_ReportNewSize, TRUE);

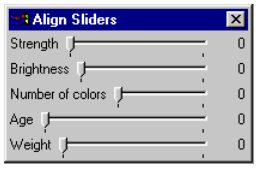
packer = new r3Packer(R3PA_Orientation, R3PAOF_VERTICAL);
window.SetGmanager(packer);

// create five sliders
var sliderLabels = ["Strength", "Brightness", "Number of colors",
                    "Age", "Weight"];

for(i = 0; i < sliderLabels.length; i++) {
    slider = new r3Slider(R3WGA_Parent, cbWindow,
                          R3GA_Text, sliderLabels[i],
                          R3GSLA_Min, 0,
                          R3GSLA_Max, 100);
    packer.ADD(R3PAPF_EXPAND | R3PAPF_FILLX, 0, slider);
}

window.REALIZE();
```

This creates the window shown on the right.



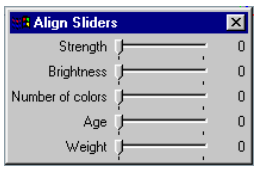
A window with vertically packed sliders.

By inserting the sliders with the R3PAAF_ALIGN, we can align the sliders based on their 'alignment point'.

Let's change the ADD method to:

```
packer.ADD(R3PAPF_EXPAND | R3PAPF_FILLX, R3PAAF_ALIGN, slider);
```

Run the script. The resulting window is shown on the right.



Sliders inserted with R3PAAF_ALIGN

Creating dynamic interfaces

When designing user interfaces, you should hide unnecessary complexity from the end user. For example, if you select a certain type of geometric object, your tool window canA automatically show only those tools which can be applied to the selected objects.

Geometric managers define the 'Stealth' attribute, which allows you to hide the controls associated with the geometric manager in question. Setting 'Stealth' to 'TRUE' makes all the managed controls invisible and collapses them to zero size. The effect is that the contents of the geometric manager can not be seen and do not require any screen space.

We strongly encourage you to take advantage of this powerful feature so let's go through an example demonstrating this.

Let us create a window which consists of a bunch of sliders. Let's imagine that some of these sliders should be shown only when the user clicks an 'Advanced' check box.

This is really trivial to implement. The only thing we need to do is to put those advanced controls into a separate geometry manager, which we can then either show or hide easily.

Here is the code creating the geometry managers and the sliders:

```
// create a window and top level packer as usual
window = ... [snip]
packer = ...[snip]

// insert couple of controls into the packer, as usual
... [snip]

// create the 'Advanced' check box.
checkb = new r3Checkbox(R3RA_Hook, showAdvanced,
                        R3WGA_Parent, window,
                        R3GA_Text, "Advanced");
packer.ADD(R3PAPF_EXPAND | R3PAPF_FILLX | R3PAPF_FILLY, 0, checkb);

// create and insert advanced controls into a separate packer.
packAdvanced = new r3Packer(R3PA_Orientation, R3PAOF_VERTICAL);
packer.ADD(R3PAPF_EXPAND | R3PAPF_FILLX, 0, packAdvanced);

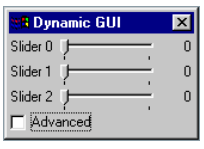
for(i = 0; i < 2; i++) {
    slider = new r3Slider(R3WGA_Parent, window,
                          R3GA_Text, "Advanced " + i);
    packAdvanced.ADD(R3PAPF_EXPAND | R3PAPF_FILLX, 0, slider);
}

// Let's hide the advanced controls by default
packAdvanced.SetStealth(TRUE);

// tell the check box where to find the 'advanced'packer and
// the parent window so that check boxes callback can access them.
checkb.packer = packAdvanced;
checkb.window = window;

// realize, as usual
[snip]
```

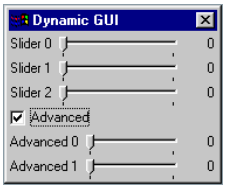
And here is the callback:




```
function showAdvanced(gadget, event, checked)
{
    if(checked)
        gadget.packer.SetStealth(FALSE);
    else
        gadget.packer.SetStealth(TRUE);
    gadget.window.FIT(R3WFP_BESTFIT);
}
```

When the user checks the Advanced check box, the advanced packer is shown. When the user resets the check box, packer is hidden.

Whenever the layout of a window is changed, we need to ask the window to update its layout by calling the FIT method.



Window layout controlled by 'Advanced' check box.

Managing Geometric Objects

Creating Geometric Objects

In the previous tutorial we learned to create various user interface objects, such as windows and buttons. Creating geometric objects is equally easy. For example, in order to create sphere objects, you first need to include the header file defining properties for the sphere class. This can be done by calling:

```
include("real/objects/r3sphere.js");
```



Note

You can find all built-in geometric objects from the 'scripts/js/real/objects' folder. Plug-in libraries may define geometric objects too, in which case their header files can be found from 'scripts/js/plugins' folder.

Once you have loaded the header file, you can create any number of sphere instances. For example, the following program creates two sphere objects.

```
sphere1 = new r3Sphere(0);
sphere1.SetRadius(0.1);

sphere2 = new r3Sphere(0);
sphere2.SetRadius(0.01);
sphere2.SetCenter(new r3Vect(0.1, 0.1, 0));
```

The above code should be self explanatory.

Inserting into the Current Project

Once you have created geometric objects, you also need to decide what to do with them. The most common operation is to insert them into the current project.

To do this, you need to find the JavaScript object managing geometric objects in the current project. This can be done by calling:

```
primLayer = GetJS("CurrentProject.Geometrics");
```



Note

The GetJS() function corresponds to the Get() function. However, the difference is that Get() returns you a Realsoft 3D object, whereas the GetJS() function creates and returns JavaScript wrapper object.

Then you can insert the two spheres into the current project by calling:

```
primLayer.LOCKEXCLUSIVE();
primLayer.INSERT(0, 0, sphere1);
primLayer.INSERT(0, 0, sphere2);
primLayer.RELEASE();
```

When modifying the contents of a layer, you must always lock the layer by calling either LOCKEXCLUSIVE() or LOCKSHARED() methods. The reason for this is that Realsoft 3D is multi threaded application and two or more threads can access the contents of a layer simultaneously.

If you are going to change the contents of a layer, a mutually exclusive 'write' lock must be obtained by calling the LOCKEXCLUSIVE() method. For example, when you insert a new object into a layer, mutually exclusively lock must be used.

In case you only want to read the contents of a layer, shared read lock is sufficient. This can be accomplished by calling the LOCKSHARED() method.

Both type of locks can be released by calling the RELEASE() method.

Creating Levels

To create a level object:

```
include("real/objects/r3level.js");

level = new r3Level(0);
```

Let's imagine we want to create couple of sub objects for the newly created level object, say two cubes.

```
include("real/objects/r3cube.js");

cube1 = new r3Cube(0);
cube1.SetParent(level);
cube1.SetP0(new r3Vect(0.0, 0.0, 0.0);
cube1.SetP1(new r3Vect(0.2, 0.0, 0.0);
cube1.SetP2(new r3Vect(0.0, 0.2, 0.0);
cube1.SetP3(new r3Vect(0.0, 0.0, 0.2);

cube2 = new r3Cube(0);
cube2.SetParent(level);
cube2.SetP0(new r3Vect(0.1, 0.1, 0.0);
cube2.SetP1(new r3Vect(0.25, 0.0, 0.0);
cube2.SetP2(new r3Vect(0.0, 0.25, 0.0);
cube2.SetP3(new r3Vect(0.0, 0.0, 0.25);
```

Note that analytic cubes are defined by four points: one corner point plus three associated 'edge' points. The remaining points for the cube are defined internally, so you don't have to (actually, can't) define them.

A level object also defines the 'Boolean' attribute for boolean operations. Let's turn the level object to a boolean operation, so that the two sub cubes gets boolean operated with each other.

```
level.SetBoolean(R3BOOL_AND);
```

Our hierarchical level object is now ready so we can insert it into the current project by calling:

```
primLayer.LOCKEXCLUSIVE();
primLayer.INSERT(0, 0, level);
primLayer.RELEASE();
```

Transforming Objects

All geometric objects define an attribute called 'Matrix', which defines so called local object space for the object. For example, an object can be moved by constructing the desired transformation matrix and concantenating it to the object's matrix.

The Matrix class defines four methods for setting up transformation matrices: translate(), scale(), rotate() and skew().

Let's imagine we need to translate the selected objects by 0.1 meters on the world 'x' axis. To construct a transformation matrix which translates in world x-axis, call:

```
m = new r3Matrix();
m.translate(0.1, 0, 0);
```

Then you can apply the transformation matrix to the selected objects by calling:

```
primLayer.LOCKEXCLUSIVE();
primLayer.TRANSFORM(m);
primLayer.RELEASE();
```

You may concatenate any number of matrix operations into a single matrix. In this case, the last concatenated operation is executed first. For example, if you want to move and scale the selected objects, call:

```
m = new r3Matrix();
m.scale(0.5 0.5, 0.5);
m.translate(0.1, 0, 0);
```

in which case the selected objects are first translated and then scaled.

Any transformation effect, such as scaling, rotation or skew, or any combination of them can be constructed and applied this way.

Single Point Editing

The TRANSFORM() method allows you to apply any linear transformation to an object. However, it does not allow you to deform the geometry of an object.

The geometry base class ('scripts/js/real/objects/r3prim.js) defines the SETPOINT() method, allowing you to set any of the geometric attributes defined by the object in question.

Let's create a nurbs curve consisting of four control points.

```
include("real/objects/r3nurbs.js");

nurbs = new r3Nurbs(0);
```

```
nurbs.SetOrder(4);
nurbs.SetCount(4);
nurbs.SetKnots([0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0]);
for(i = 0; i < 4; i++)
    nurbs.SETPOINT(i, new r3Vect(i * 0.1, 0, 0));
```

Here we used SETPOINT() method to define initial geometry for the nurbs curve.

Another useful method is GETPOINT(), which allows you to fetch the value of any desired geometry point. By using GETPOINT and SETPOINT methods, you can apply deformation effects to geometric objects. First you call GETPOINT() to fetch the current value, then you modify the value using some non linear function and finally you assign the modified value back to the object by calling SETPOINT().

For example, the following uses fractal noise to deform all geometric points of the given object.

```
function applyNoise(obj)
{
    // create new vector object
    p = new r3Vect();

    // fetch the number of geometry points in the given object
    pcount = obj.GetPointCount();

    for(i = 0; i < pcount; i++) {
        // fetch the current value of the point 'i'
        obj.GETPOINT(p, i);

        // displace the point using fractal noise
        p2 = p.noise(3, 4);
        p2.fmul(0.1);
        p.fadd(p2);

        // set displaced point back to the object
        obj.SETPOINT(i, p);
    }
}
```

Enumerating Selected Objects

Now that we managed to implement a somewhat interesting deformation effect, let's see how it could be plugged in as a new deformation tool.

The layer base class defines a method which is very useful when implementing custom tools. The method is called ENUMSELECTLIST(). This method scans through the currently selected objects and passes them to the given hook function, such as the deformation function we wrote above.

There is only one complication. Because the JavaScript objects are 'wrapper' objects attached to actual Realsoft 3D objects, our hook function will be called with a Realsoft 3D object rather than a JavaScript object. The low level Realsoft 3D functionality does not know if the enumeration request was generated by JavaScript wrapper object. Therefore, we must create a JavaScript interface to the given Realsoft 3D object before we can access its properties through JavaScript methods.

Here is the code:

```
function deformByNoiseCallBack(r3obj)
```

```
{
    // create JavaScript interface to 'r3obj'
    jsobj = R3ToJS(r3obj);

    // call the applyNoise() function
    if(obj)
        return applyNoise(jsobj);
    return 1;
}
```

Now you can deform the selected objects by calling:

```
primLayer.LOCKEXCLUSIVE();
primLayer.ENUMSELECTLIST([R3RA_Hook, mycallback]);
primLayer.RELEASE();
```

The 'R3RA_Hook' parameter is used for specifying the callback function to be called.

You can find this example from the 'scripts/js/myclasses/toolbars/mydeform.js'. To test it, create couple of SDS or NURBS objects, select them and apply 'Scripts/Toolbars/mydeform.js' pull-down menu.

Vectors and Matrices

Vectors

In the Managing Geometric Objects tutorial we used vector objects in various ways. Most geometric objects define vector attributes and in order to manage geometric objects, you need to know something about vectors.

Vector is one of the built-in classes in Realsoft 3D, so you don't have to include a header file to get the class defined.

A vector object defines three attributes: x, y and z. It can be used for representing geometry points, direction vectors, and many other things of 3D nature.

You can create a vector by calling:

```
v = new r3Vect();
```

which creates so called null vector. All the three attributes are initialized to zero. You can initialize a vector to a desired value by calling the set() method.

```
v.set(1, 0, 0);
```

or, simply by setting the attributes using an assignment operator:

```
v.x = 1.0;
v.y = 0;
v.z = 0;
```

You may also create a vector by defining initial x, y and z values to the constructor:

```
v = new r3Vect(1, 0, 0);
```

Vector class also defines methods for vector addition and other vector operations. For example, let's imagine we have two vectors:

```
a = new r3Vect(1, 0, 0);
b = new r3Vect(0, 1, 0);
```

Then we want to perform a vector operation c = a + b. This can be accomplished as follows:

```
c = a.add(b);
```

This corresponds to the following code:

```
c.x = a.x + b.x;
c.y = a.y + b.y;
c.z = a.z + b.z;
```

Let's take another commonly needed operation like 'scalar multiplication'. For example, if you want to perform operation 'b = a * 2.0' i.e. to double the length of the vector, call:

```
b = a.mul(2.0);
```

This corresponds to the following code:

```
b.x = a.x * 2.0;
b.y = a.y * 2.0;
b.z = a.z * 2.0;
```

There are two methods for applying most vector operations: those which result in a new vector object and those which apply the operation to the vector in question. We demonstrated the former type of methods above.

If a method name starts with letter 'f', then it represents the latter form of method i.e. it applies the vector operation to the object in question rather than resulting in a new vector. If you are familiar with Realsoft 3D Software Development Kit you will find that the method names correspond to those used in the SDK.

For example, the following method:

```
a.fadd(b);
```

adds the vector 'b' to 'a'. This corresponds the code:

```
a.x = a.x + b.x;
a.y = a.y + b.y;
a.z = a.z + b.z;
```

Or, if you want to double the length of a vector, call:

```
a.fmul(2.0);
```

which corresponds to the following code:

```
a.x = a.x * 2.0;
```

```
a.y = a.y * 2.0;
a.z = a.z * 2.0;
```

One very useful vector method is called 'noise', which allows you to compute various fractal effects. The noise takes two parameters: dimensions and octaves and results in a new vector object representing the strength of the noise field.

```
b = a.noise(3, 4);
```

For a list of all methods supported by the vector class, see the reference manual/vector class.

Matrices

Realsoft 3D uses a very standard 4x4 matrix to define so called 'local object space' property for all geometric objects. The matrix consists of 16 floating point values (4x4).

The mathematical theory behind matrices is beyond this tutorial. However, it is possible to use the matrix class even if you do not understand the deep theory.

The matrix class is a built-in class and there are no header files that you should include.

Constructing Matrices

As demonstrated in the Managing Geometric Objects tutorial, a new matrix object can be created by calling:

```
m = new r3Matrix();
```

This creates a so called 'identity' matrix. Applying such a matrix to does not transform objects in any way.

You may also pass four vector parameters to the constructor:

```
origin = new r3Vect(0, 0, 0);
x = new r3Vect(1, 0, 0);
y = new r3Vect(0, 1, 0);
z = new r3Vect(0, 0, 1);
m = new r3Matrix(origin, x, y, z);
```

The parameters define a coordinate system (i.e. a space), which is an alternative (more intuitive) representation for a matrix. For example, to construct a transformation matrix which moves the object from world origin to x=1.0, y=0.5:

```
origin = new r3Vect(1.0, 0.5, 0);
x = new r3Vect(1, 0, 0);
y = new r3Vect(0, 1, 0);
z = new r3Vect(0, 0, 1);
m = new r3Matrix(origin, x, y, z);
```

Or, to construct a matrix which scales the object to twice the original size:

```
origin = new r3Vect(0, 0, 0);
x = new r3Vect(2, 0, 0);
y = new r3Vect(0, 2, 0);
z = new r3Vect(0, 0, 2);
```

```
m = new r3Matrix(origin, x, y, z);
```

A third way to construct a matrix is by passing sixteen number parameters to the constructor, specifying the initial state of the sixteen attributes of the matrix. However, this is not very convenient and rarely used.

Constructing Transformation Matrices

Many applications allow you to construct matrices from a number of transformation components namely translate, scale, rotate and skew.

Concatenated transformations are executed in the reversed order (first in, last out).

Let's demonstrate this by going through a couple of examples.

If you run the following code:

```
m = new r3Matrix();
m.scale(0.5, 0.5, 0.5);
primLayer.LOCKEXCLUSIVE();
primLayer.TRANSFORM(m);
primLayer.RELEASE();
```

the result is that all the selected objects shrink towards the world origin so that they end up twice as small.

However, let us imagine you want to scale the selected objects about point [0.1, 0.2, 0.0] rather than about the origin.

To do this, you first have to move the object, so that this point matches the world origin. Then you apply the scale effect and finally you move the object back to its original position. The code to accomplish this is:

```
m = new r3Matrix();
m.translate(0.1, 0.2, 0.0);
m.scale(0.5, 0.5, 0.5);
m.translate(-0.1, -0.2, 0.0);

primLayer.LOCKEXCLUSIVE();
primLayer.TRANSFORM(m);
primLayer.RELEASE();
```

Because specified transformations are executed in reversed order, the translation [-0.1, -0.2, 0] is executed first. Then the object is scaled and finally moved back to its original position. The effect is that the object seems to be scaled down towards the point [0.1, 0.2, 0].

Specifying transformation center in one call

The transformation methods accept more than three parameters, allowing you to specify transformation center in one call.

The above scaling effect can easily be set up, simply by calling:

```
m = new r3Matrix();
m.scale(new r3Vect(0.1, 0.2, 0), 0.5, 0.5, 0.5);
```

SIn other words, if you pass four parameters to the scale method, the first parameter is assumed to be the origin of scale. The remaining three parameters specify scaling factors in x, y and z dimensions.

Let's take another example: to construct a transformation matrix rotating the selected objects about an axis aligned to world 'x' axis and passing through the point [0, 1, 0], call:

```
m = new r3Matrix();
m.rotate(PI, new r3Vect(0, 1, 0), new r3Vect(1.0, 0.0, 0.0));
```

The first parameter specifies rotation angle in radians. The second parameter specifies the origin of rotation and the third specifies the axis of rotation.

The script 'scripts/js/myclasses/mytransform.js' shows you couple of examples of tools transforming geometric objects. You can test these tools by selecting the pull down menu 'Scripts/Toolbars/myransform.js'.

Managing Materials

Creating a Material

In this tutorial we show how one can manage VSL (Visual Shading Language) material objects using JavaScript. Basic understanding of VSL is assumed.

VSL Material and other VSL specific objects are defined in the r3code library. Correspondingly, the header files for them can be found from 'scripts/js/real/code' folder.

To create a VSL material:

```
include("real/code/r3vsl.js");

mat = new r3Vsl(0);
mat.SetName("my material");
```

The above creates an empty VSL material object named 'my material'.

In order to let a VSL material define surface properties, you need to create and insert a so called surface shader object into the VSL material.

```
include("real/code/r3mpphas.js");   // shader

shader = new r3Mpphaselevel(0);
shader.SetPhase(R3CLID_PHASESURFACE);
vsl.INSERTPROP(0, 0, shader);
```

The above code creates a 'phase level' VSL object and sets its type to R3CLID_PHASESURFACE. VSL objects inside this level will be able to access and define various surface properties, such as surface color.

Let's create a texture mapping object to define the surface color property:

```
include("real/code/r3mptext.js");   // texture map

texture = new r3Mptexture(0);
texture.SetParent(shader);
texture.SetImageName("textures/lake.jpg");
```

By calling the SetParent() method you link the texture object to the shader object. The SetImageName() method asks the texture mapping object to use the "textures/lake.jpg" image.

To insert the material into the current project, you need to fetch the JavaScript class managing material objects in the current project.

```
matLayer = GetJS("CurrentProject.Materials");
```

Then you can insert the material into the current project by calling:

```
matLayer.LOCKEXCLUSIVE();
matLayer.INSERT(0, 0, vsl);
matLayer.RELEASE();
```

Reflection Mapping

The technique where an image is used for defining the surface color is called texture mapping. In Realsoft 3D, images can be used for defining any material property, such as Brilliancy, Transparency, Refraction Factor, Specularity, just to name a few.

The reason why the previous example resulted in texture mapping rather than some other type of mapping is that the default output channel for the texture mapping object is 'Color'.

There are many other channels available. Each channel is identified by two properties: class and name. For example, renderer uses the channel object whose class is R3CLID_COLORCHANNEL and name is 'Color' to represent surface color.

So, to get 'reflection mapping' rather than 'color mapping', we just need to assign the texture mapping object to the 'Reflection' channel:

```
texture.SetOutputChannel(R3CLID_REFLECTIONCHANNEL);
texture.SetOutputName("Reflection");
```

This will turn the texture mapping into reflection mapping.

Creating VSL Curve Objects

The VSL Curve object (r3Mpcurve) is defined in the 'scripts/js/real/code/r3mpcurv.js' file. However, what makes it interesting is that it does not define any curves itself. Instead, it uses the r3Curve object for this. So, in order to create a curve object we need to know how to fetch the actual curve object and how to access the curve data in it.

Let's create a material with a curve object:

```
include("real/code/r3mpcurv.js");

// create a VSL material
vsl = new r3Vsl(0);
vsl.SetName("curve");

// create a surface shader object for the material
shader = new r3Mpphaselevel(0);
shader.SetPhase(R3CLID_PHASESURFACE);
vsl.INSERTPROP(0, 0, shader);

// create a VSL curve object into the shader object
mpcurve = new r3Mpcurve(0);
```

```
mpcurve.SetParent(shader);

// attach to color channel
mpcurve.SetOutputChannel(R3CLID_COLORCHANNEL);
mpcurve.SetOutputName("Color");
```

So, we have now created a VSL curve object and attached it to the Surface Color channel.

In order to add some points into the curve object, we must first fetch the actual curve object from the VSL curve object. This can be done by calling the GetCurve() method.

```
curve = mpcurve.GetCurve();
```

The above call returns r3Curve object (scripts/js/real/code/r3curve.js).

By default all curve objects have two points, defining a linear curve. Let's get rid of these default curves. To delete existing points from the curve, call:

```
curve.DELETEALLPOINTS();
```

Then we can add a number of points to the curve by calling ADDPOINTIFNOTONLINE method. Let's add three points to the curve:

```
curve.ADDPOINTIFNOTONLINE(R3TID_VECTOR, new r3Vect(0, 0, 0),
    new r3Vect(0.3, 0.2, 0.1));
curve.ADDPOINTIFNOTONLINE(R3TID_VECTOR, new r3Vect(0.5, 0.5, 0.5),
    new r3Vect(0.4, 0.1, 0.2));
curve.ADDPOINTIFNOTONLINE(R3TID_VECTOR, new r3Vect(1, 1, 1),
    new r3Vect(0.9, 0.7, 0.6));
```

The first vector value specifies x coordinates for the point to be added. The second vector parameter specifies corresponding y values.

Because we attached the curve object to Color channel, the curve defines three sub curves: one for each color sub channel (red, green, blue). Therefore, we must pass vectors to the ADDPOINTIFNOTONLINE method. If we attached the curve object to say the Alpha channel, there would be only one curve. Correspondingly, we would call:

```
curve.ADDPOINTIFNOTONLINE(R3TID_FLOAT, 0.1, 0.5);
```

The final step is to insert the VSL material into the current project:

```
matLayer.LOCKEXCLUSIVE();
matLayer.INSERT(0, 0, vsl);
matLayer.RELEASE();
```

Layers

In the Select Window tabs, the user can manage various type of objects, such as geometric primitives, materials, rendering settings and so on. Each of the Select Window tabs correspond to a certain layer object. For example, the material tab shows you the contents of the 'material layer' of the current project. The Grid tab shows you the contents of the 'grid layer' of the current project.

We have already explored some of these layer objects, such as geometric primitives and material objects.

All layer objects have a common base class, called 'scripts/js/real/layer/r3objlay.js'. This class defines methods such as INSERT(), LOCKEXCLUSIVE() and RELEASE().

Let's go through a couple of other useful methods defined by the object layer class.

Exporting Selected Objects

The layer object defines a property called 'selected objects'. Many of the tools in Realsoft 3D tools are actually applied to selected objects. For example, the Move tool translates selected geometric objects. We have already used the method ENUMSELECTLIST to deform objects. Let's demonstrate another enumeration method called ENUMOBJECTS. This method enumerates the entire contents of a layer.

Let's assume we want to export geometric objects to a file named 'export.js'.

To create a Realsoft 3D file object, call:

```
myFile = new r3File(R3FIA_FileName, "export.js", R3FIA_Mode, "wa");
```

Then we need a hook function for handling geometric objects. In the hook function, we create a JavaScript wrapper object for the enumerated Realsoft 3D geometry object, fetch the attributes of it and write them out to the opened file. For sake of simplicity, we handle only the sphere object here. It would be a very straight forward task to add support for other geometric objects as well.

```
include("real/objects/r3sphere.js");

function myExportHook(r3obj)
{
  obj = R3ToJS(r3obj);
  if(!jso)
      return 1;

  // fetch the class identifier of the object
  iClid = obj.GetClassID();

  if(clid == R3CLID_SPHERE) {

    sName = obj.GetName();
    pCenter = obj.GetCenter();
    vA = obj.GetA();
    vB = obj.GetB();
    vC = obj.GetC();

    // write them out
    this.myFile.PUTS("ClassID: " + iClid + "(ellipsoid)\n");
    this.myFile.PUTS("Name: " + sName + "\n");
    this.myFile.PUTS("Center: {" + pCenter.x + ", " + pCenter.y +
                              ", " + pCenter.z + "}\n");
    this.myFile.PUTS("A: {" + vA.x +  ", " + vA.y + ", " + vA.z + "}\n");
    this.myFile.PUTS("B: {" + vB.x + ", " + vB.y + ", " + vB.z + "}\n");
    this.myFile.PUTS("C: {" + vC.x + ", " + vC.y + ", " + vC.z + "}\n");
  }
  return 1;
}
```

Then you can export objects by calling the ENUMOBJECTS method:

```
primLayer.LOCKSHARED();
primLayer.ENUMOBJECTS([R3RA_Hook, myExportHook]);
primLayer.RELEASE();
```

Setting Attributes to the Selected Objects

Another often needed task is to set desired attributes to the selected objects. There are many ways to do this. For example, you can use the ENUMSELECTLIST method to enumerate selected objects to a hook function, which then checks the type of the object and sets appropriate attributes.

Another approach is to call the SETONSELECTED method. This methods accepts a tag list, specifying attribute identifiers, and corresponding values. For example, let's assume we need to set spin and velocity for the selected objects. Velocity and Spin are attributes defined by 'scripts/js/real/objects/r3prim.js' class, which is the base class for all geometric objects.

```
vel = new r3Vect(0.1, 0.1, 0);
spin = new r3Vect(PI * 2, 0, 0);

primLayer.LOCKEXCLUSIVE();
primLayer.SETONSELECTED([R3PRIMA_Mass, 1.2,
                         R3PRIMA_Velocity, vel,
                         R3PRIMA_Spin, spin]);
primLayer.RELEASE();
```

Implementing New Classes

JavaScript supports a basic object oriented paradigm, allowing you to implement new kinds of objects by deriving them from existing objects.

In this tutorial we show you how to implement new JavaScript classes in Realsoft 3D. We also explore the internal structure of Realsoft 3D and demonstrate how to transform geometric objects, create new geometric objects, export objects to a file and create new material objects.



Note

This tutorial is not a JavaScript tutorial. Basic knowledge of JavaScript's object oriented programming model is assumed.

Class Header Files

In the previous tutorials we used the include() function to load various header files.

For example, we loaded button's header file:

```
include("oops/r3button.js");
```

Each of these header files define properties for one Realsoft 3D class. For example, 'scripts/js/oops/r3button.js' defines the JavaScript class corresponding to the Realsoft 3D Button class.

The substructure of 'scripts\js' folder corresponds to the library structure of Realsoft 3D. For example, the 'oops' sub folder stands for Realsoft Graphics Oy's Object Oriented Programming System and it defines classes for the r3oops

library. If you buy a plug-in that supports JavaScript you get a new folder that contains the header files corresponding to the classes in the plug-in library.

You can also add new sub folders into 'scripts/js' folder for your own JavaScript classes. In fact, the scripts folder should already have 'myclasses' sub folder, which contains a number of sample classes.

myWindow Class

Any of the sample scripts we have gone through so far can be turned to JavaScript classes easily.

Let's recall the first sample we went through in the previous Intro tutorial:

```
include("oops/r3window.js");

window = new r3Window(R3WGA_Parent, _r3gui,
                      R3WGA_Left, 300,
                      R3WGA_Top, 200,
                      R3WGA_Width, 300,
                      R3WGA_Height, 200,
                      R3WA_ReportCloseWindow, TRUE,
                      R3WA_ReportNewSize, TRUE,
                      R3WA_Title, "Dummy Window");

window.REALIZE();
```

Creating window objects is straightforward, but specifying all the above attributes creates a lot of typing overhead.

Let's assume we want to create a new JavaScript window, which creates the above kind of a window with minimal typing overhead. This can be done by deriving a new class which takes care of setting the appropriate attributes for us.

```
include("oops/r3window.js");

// constructor function
function myWindow(name, width, height)
{
    if(!arguments.length)
        return;
    this.base = r3Window;

    // our super class
    this.base(R3WGA_Parent, _r3gui,
              R3WGA_Left, 300,
              R3WGA_Top, 200,
              R3WGA_Width, width,
              R3WGA_Height, height,
              R3WA_ReportCloseWindow, TRUE,
              R3WA_ReportNewSize, TRUE,
              R3WA_Title, name);
}
// derive 'myWindow' from the 'r3Window' base class
myWindow.prototype=new r3Window;
```

This is how to derive new classes in JavaScript. If you save the above code as 'js/myclasses/mywindow.js', you will be able to create windows as follows:

```
include("myclasses/mywindow.js");

win = new myWindow("My cool window", 300, 200);
win.REALIZE();
```



Note

You should not put the R3WGM_REALIZE call into a constructor. This method should be called only once from the highest level (whoever creates your object is responsible also for realizing them). If you put the realize method into the constructor function of your class, it gets called multiple times causing flickering. Another method which you should not put into constructor functions is 'R3WGM_FIT'.

Implementing 'myToolBar' Class

Now that we know how to derive new JavaScript classes for existing classes, let's go through a more practical example. Let's create a fully functional tool bar class, which allows us to create custom tool bars as easily as possible.

Recall the packer example we went through in the beginning of the intro tutorial. We created a window with a packer geometric manager and then inserted a bunch of buttons into the packer. Typically toolbars contain one or more horizontally or vertically packed tool buttons. So we can take the packer example as a starting point for developing our tool bar class.

Again, we need to wrap the creation code into a constructor function to get a proper JavaScript class defined. Because the tool bar should be user configurable, we just create a window with an empty packer in the constructor. Then we implement another method which allows us to insert tool buttons into the toolbar.

So, let's see how all this can be implemented.

Here is the code:

```
// our tool bar consists of a window, a packer and one or more buttons
include("oops/r3button.js");
include("oops/r3window.js");
include("oops/r3packer.js");

// let the user to control the orientation of the tool bar
var MYTB_HORIZONTAL = 0
var MYTB_VERTICAL = 1

// add button with user specified label and callback function
function mytbAddTool(buttonLabel, fnCallBack)
{
    button = new r3Button(R3RA_Hook, fnCallBack,
                          R3WGA_Parent, this,
                          R3GA_Text, text);
    if(button) {
        // insert button into the packer
        this.packer.ADD(R3PAPF_FILLX, 0, button);

        // tell the button where to find the project
        button.layer = this.layer;
    }
    return button;
}
```

```
// constructor: create a window with empty packer
function myToolBar(titlebar, orientation, layer)
{
    if(!arguments.length)
        return;
    this.base = r3Window;
    this.layer = layer;

    this.base(R3WGA_Left, 200,
              R3WGA_Top, 150,
              R3WA_ReportNewSize, TRUE,
              R3WA_ReportCloseWindow, TRUE,
              R3WA_Title, titlebar);

    // create a packer with given orientation
    this.packer = new r3Packer(0);
    this.packer.SetOrientation(orientation);

    // define method for inserting tool buttons
    this.AddTool = mytbAddTool;

    // and give the packer to the window, as usual
    this.SetGmanager(this.packer);
}

myToolBar.prototype=new r3Window;
```

Testing 'myToolBar' Class

The above JavaScript program can be found from 'scripts/js/myclasses/toolbars/mytoolbar.js'.

Let's add a tool button which creates an analytic sphere object. For this we need to create a call back function. This function is called when the user clicks the tool button. In the function we simply create a sphere and insert it into the geometry layer.

```
function mySphereCallback(button, event, value)
{
    // create a sphere
    sphere = new r3Sphere(0);

    // set some sphere specific attributes
    sphere.SetRadius(0.1);
    sphere.SetCenter(new r3Vect(0.1, 0.0, 0.0));
    sphere.SetName("my sphere");

    // insert the sphere into the project
    button.layer.LOCKMUTEX(0, 0, sphere);
    button.layer.INSERT(0, 0, sphere);
    button.layer.RELEASE();
}
```

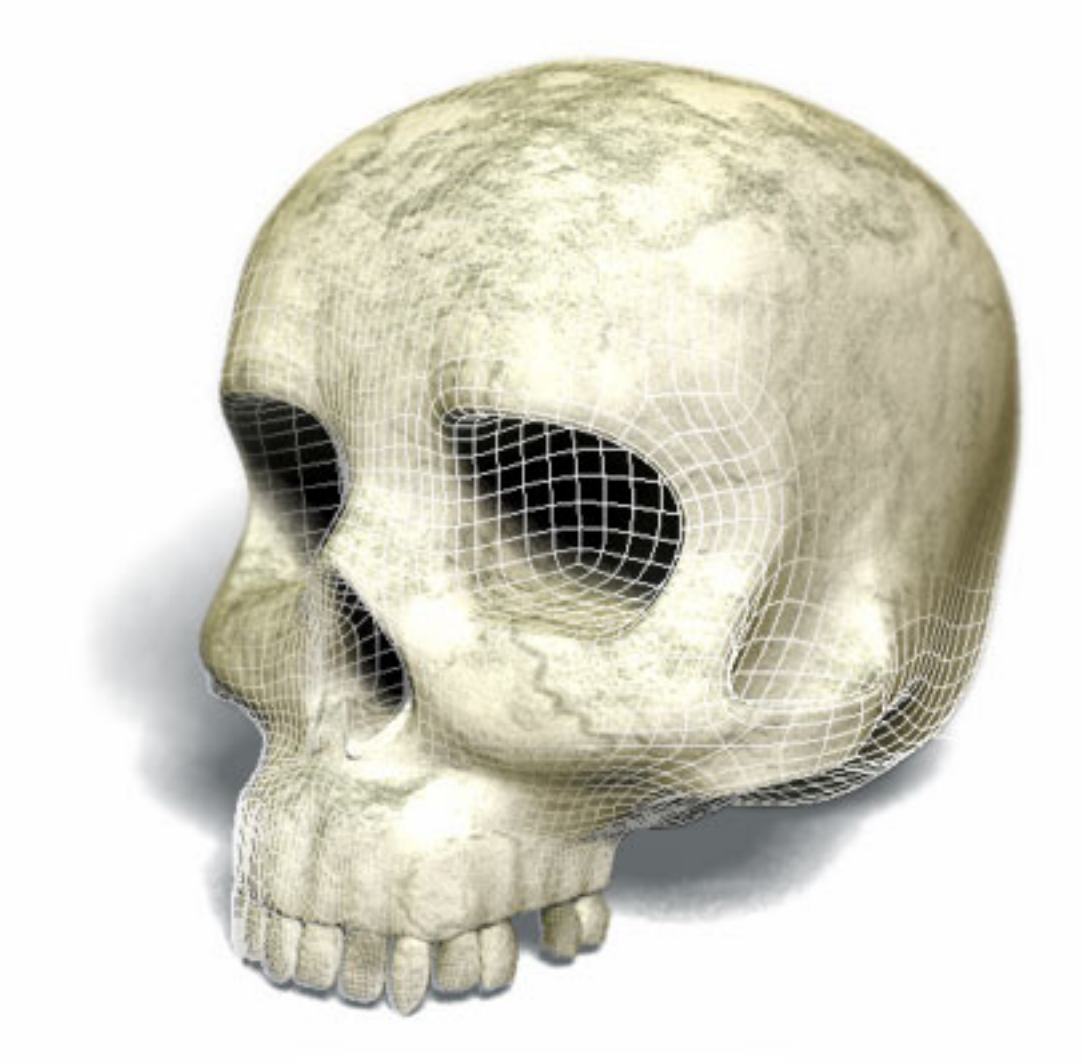
Now we have defined one callback which can create a sphere into the current layer. We have also defined a new tool bar class.

To actually create a tool bar, enter:

```
tbar = new myToolBar(MYTB_HORIZONTAL, primLayer);
tbar.AddTool("Sphere", mySphereCallback);
```

When you click the 'Sphere' button, the 'mySphereCallback' is called and new sphere will appear into the current project.

The 'scripts/js/myclasses/toolbars' folder contains several JavaScript examples derived from the 'myToolBar' class.



Chapter 13. Miscellaneous

Hot Keys

The shortcut keys activate or perform various frequently used program functions.

```
a   Again (repeat latest action)
b   Bend
c   Reserved
d   Drop selected objects to a level
e   Reserved (coordinate lock)
f   Fit view to selected objects
g   Grid on/off
h   OpenGL on/off
i   Invisible on/off
j   View to object
k   Object view
l   Extend (change Length)
m   Move
```

```
n   Enter numeric mode
o   Mirror (Opposite)
p   Open property window
q   Reserved (coordinate lock)
r   Rotate
s   Scale
t   Configure view for modeling in object space
u   Reset from modeling in object space
v   Reserved for view compass
w   Reserved (coordinate lock)
x   Reserved (coordinate lock)
y   Reserved (coordinate lock)
z   Reserved (coordinate lock)

A   Animate by path / Add SDS edge

B   Create Backdrop / SDS bevel
C   Create Camera from the current view
D   Create Subdivision cube / Destroy SDS face
E   Reserved for modifiers
F   Fit to selected handles or object
G   Create group
H   Extrude tool (H for Height) / Extrude SDS face
I   Create vertical axis line (looks like I)
J   Create NURBS curve by control points
K   Create NURBS curve by knotpoins
L   Create Level
M   Create MetaBall / Merge SDS vertices

N   Create NURBS mesh
O   Create analytic cylinder (O-like cross section)
P   Particle tool
Q   Reserved for modifiers
R   Create analytic Rectangle
S   Create analytic Sphere / Subdivide SDS face / Split SDS edge
T   Parallel (Texture) mapping tool / Tunnel SDS faces
U   Create analytic cube
V   Create analytic cone (sharp V like shape)
W   Reserved for modifiers
X   Reserved for modifiers
Y   Reserved for modifiers
Z   Reserved for modifiers

Ctrl a  Select all
Ctrl b  Define render box for view window
Ctrl c  Copy
Ctrl d  Duplicate
Ctrl f  Font tool
Ctrl g  Execute current macro
Ctrl i  Invert selection
Ctrl l  Load file
Ctrl m  Macro recording on/off
Ctrl n  New project
Ctrl p  Pause/continue macro recording
Ctrl q  Save active view window contents to an image file
Ctrl s  Save file
```

```
Ctrl t  Deselect
Ctrl r  File render window
Ctrl v  Paste
Ctrl x  Cut
Ctrl y  Redo
Ctrl z  Undo


1   Reserved as modifier (iplane horizontal lock)
2   Reserved as modifier (iplane vertical lock)
3   Reserved as modifier (iplane normal lock)
4   Reserved as modifier (lasso selector)
5   Reserved as modifier (snap to nearest point)
6   Reserved as modifier (snap to curve)
7   Front View
8   Side View
9   Top View
0   Ray trace current view


Space     Edit mode on/off
Alt Space Selection Painter
Home      Reset view
Insert    Move pivot point
Del       Delete selected object/point/face etc.

!   Animation record on/off
§   Create distant light
@   Create ambient light
£   Create point light
$   Create spot light
&   Boolean And
%   Boolean And Not
/   Hide wire geometry
*   Perspective on/off

-   Backwards 1 frame
+   Forward one frame
>   Play animation/stop animation
<   Rewind animation
)   Tracking on & play animation
(   Camera tracking off
#   Ground plane drawing on/off
;   Reset grid origin
=   Reserved

?   Unused
[   Unused
]   Unused
{   Unused
}   Unused

F1   Reserved
F2   Select window show/hide
F3   Toolbar show/hide
F4   Control bar show/hide
```

```
F5   View bar show/hide
F6   Choreographs
F7   Animation bar show/hide
F8   Numeric show/hide
F9   Status show/hide
F10  Reserved
F11  UV editor
F12  Color window

Ctrl F2  Info window
Ctrl F3  Scripting window
Ctrl F4  View property window
Ctrl F5  Floating view window
```

Modifier keys

Modifier keys affect the performed program functions while they are pressed down. They do not start any operation or tool.

Input plane constraints

```
1              Lock to horizontal direction of input plane
2              Lock to vertical direction
3              Lock to normal direction
Shift          Lock to nearest axis
```

Object space constraints

```
q              Lock to horizontal direction of object space
w              Lock to vertical direction
e              Lock to normal direction
```

World space constraints

```
x              Lock to horizontal direction of absolute space
y              Lock to vertical direction
z              Lock to normal direction
```

Grid control by mouse click/dragging

```
. (dot)        Set hotpoint
Alt . (Alt+dot) Set hotpoint for all views
```

Snapping

```
5              Snap to nearest point
6              Snap to curve
Ctrl           Snap to selected objects only
Alt            Snap to current level objects only
Alt            Snap to existing points in SDS object face creation
```

Shift	Grid snapping off for SDS face, edge and vertex handle moving and object translation handles

Mouse projection to 3D

, (comma)	Project LMB click to underlying surface
, (comma)	3D snap (don't project a dragged point to input plane)

Object and handle selection

Shift	Add to selection (OR)
Ctrl	Intersection with previous selection (AND)
Ctrl	Range select: select all from previous selected to clicked one
Shift Ctrl	Invert selection (XOR)
Alt	Subtract from selection
4	Lasso selector in dragging

View navigation by the right mouse button

Shift	Zoom (change view angle/zoom scale)
Ctrl	Pan (move camera)
Alt	Rotate camera about aimpoint
Shift Ctrl	Move camera forward/backward
Shift Alt	Rotate in 45 degree steps
Ctrl Alt	Adjust camera banking
Shift Ctrl Alt	Adjust camera banking in 45 degree steps

View navigation by the middle mouse button

Shift	Move camera forward/backward
Ctrl	Pan (move camera)
Alt	Rotate camera about viewpoint
Shift Alt	Rotate camera about viewpoint in 45 degree steps

View cruising with cursor keys

Tab	View camera cruising on
Tab+Shift	Turbo boosted cruising
Tab+Up	Accelerate
Tab+Down	Decelerate
Tab+Left	Turn left
Tab+Right	Turn right
Tab+Alt+Up	Turn up
Tab+Alt+Down	Turn down
Tab+Alt+Left	Bank counterclockwise
Tab+Alt+Right	Bank clockwise
Tab+Ctrl+Up	Move up
Tab+Ctrl+Down	Move down
Tab+Ctrl+Left	Move left
Tab+Ctrl+Right	Move right

```
Tab+Left+Right  Turn towards horizontal flight
```

Create/modify tools

```
Shift  Uniform 3D scale/45 degree rotate grid/Uniform rectangle and cube
Ctrl   Duplicate at modify ( transformation handles only )
Alt    Instance at modify(via handles)/Snap to existing SDS points
```

Compass menus

```
v compass       View tools
y compass       Light sources & texture tools
q compass       Create tools
w compass       Wire compass
e compass       Modify tools

x compass       Reserved
z compass       Reserved

1 compass       Unused (reserved for user compass)
2 compass       Unused
3 compass       Unused
```

Geometry modify tools

```
b   Bend
i   Invisible on/off
l   Extend (change Length)
m   Move
o   Mirror (Opposite)
r   Rotate
s   Scale
/   Hide/show wire geometry
```

Object hierarchy editing tools

```
d   Drop selected objects to a level
L   Create Level
&   Boolean And
%   Boolean And Not
Ctrl a  Select all
Ctrl c  Copy
Ctrl d  Duplicate
Ctrl i  Invert selection
Ctrl t  Deselect
Ctrl v  Paste
Ctrl x  Cut
```

Animation tools

```
!   Animation record on/off
-   Backwards one frame
+   Forward one frame
>   Play animation/stop animation
<   Rewind animation
)   Tracking on & play animation
(   Camera tracking off
A   Create path animation
```

Light sources

```
§   Create distant light
@   Create ambient light
£   Create point light
$   Create spot light
```

Gui elements

```
p    Property window
F2   Select window show/hide
F3   Toolbar show/hide
F4   Control bar show/hide
F5   View bar show/hide
F6   Choreographs
F7   Animation bar show/hide
F8   Numeric show/hide
F9   Status show/hide
F11  UV editor
F12  Color window

Ctrl F2  Info window
Ctrl F3  Scripting window
Ctrl F4  View property window
Ctrl F5  Floating view window
Ctrl+r   File render window
Ctrl+l   File load window
```

View window

```
f   Fit view to selected objects
g   Grid on/off
h   OpenGL on/off
j   View to object
k   Object view
t   Configure view for modeling in object space
u   Reset from modeling in object space
*   Perspective on/off
#   Ground plane drawing on/off
;   Reset grid origin
0   Zero key ray traces the current view
7   Front View
8   Side View
9   Top View
```

```
Ctrl b  Define render box for view window
Ctrl q  Save view to image file
```

SDS editing

```
A   Add edge
B   Bevel
D   Destroy face
H   Extrude face
M   Merge SDS vertices
S   Subdivide face / Split edge
T   Tunnel faces
```

Creation and construction tools

```
B   Create Backdrop
C   Create Camera from the current view
D   Create Subdivision cube
F   Font tool
G   Create group
H   Extrude tool
I   Create vertical axis line (looks like I)
J   Create NURBS curve by control points
K   Create NURBS curve by knotpoins
L   Create Level
M   Create MetaBall
N   Create NURBS mesh
O   Create analytic cylinder (O-like cross section)
P   Particle tool
R   Create analytic Rectangle
S   Create analytic Sphere
T   Parallel (Texture) mapping tool
U   Create analytic cube
V   Create analytic cone (sharp V like shape)
§   Create distant light
@   Create ambient light
£   Create point light
$   Create spot light
```

Unused keys for user shortcuts

```
?   Unused
[   Unused
]   Unused
{   Unused
}   Unused
Alt combinations are reserved for user's own shortcuts
```

Productivity Tips

Network rendering rules of thumb are:

- If some servers are slow, the box count should be 4-10 times the number of threads.
- Box division should be symmetric. 7*4 instead of 14*2 - it renders quicker as the boxes are "squarer".
- Put the slowest systems last in the server list defined by the render settings 'Boxes' tab. The fast systems have time to render the first boxes while the client is activating the slow systems.
- Save texture maps as JPG images for faster network render times.

Use analytical objects as much as possible in complex scenes - SDS and NURBS objects use larger amounts of memory.

Use the SHIFT key to constrain objects and tools to horizontal or vertical directions. The SHIFT key can also be used to size object proportionally when using object handles, and set rotation angles to 45 degree increments.

Snap to points and corners of objects with the '5' key.

On color swatches, right click to bring up the "expand" option. This allows you to quickly select the appropriate color using the color wheel.

Make use of clip maps to reduce scene complexity.

Use level structures to organize your work - drop multiple objects to their own level by multi-selecting with the SHIFT key, then right click on the select window and select "Drop to a level" from the popup menu. Projects organized with clever level use are far more manageable.

To change parameters of any sub objects - (e.g. change all objects in a level to "not selectable") simply select the level object, change the "not selectable" tag and all sub object are then not selectable. Likewise, to change all the objects in a scene to have an alpha value of 1 (to render to an alpha channel): Select the ROOT level, and change the 'Col/Surface properties/Alpha' value to 1 - all objects will be rendered solid against a clear background.

When panning and zooming objects, occasionally they may "disappear" in the view window. This is most likely due to the clipping planes utilized to speed up view refreshes. To see all the objects in a scene, simply drag the "Root" level icon into the view window. This sets the clipping planes to the extents of the scene and allows the user to see all objects. Conversely, dragging the object in question to the view zooms to the extents of that object/level.

Snap to edges and lines with the 6 key.

You can use CAD packages to model a complex geometry and use the IGES plugin to import it. This allows you to make use of CAD programs' inherent speed and accurate modeling tools, then import them into Realsoft 3D as splines, which can then be extruded or lofted into 3D shapes (Requires IGES plugin).

Use multiple Input Planes to align new objects with existing objects. Simply drag the IPlane onto the view to set it as the current IPlane - and make sure that under the view properties window "Lock to Input Plane" is set. To align the input plane to the current view angle, use the "Set Input Plane Orthogonal to View" button. To match the rotation of an object, simply copy the rotation angle from the object in question and paste it into the rotation angle field of the IP plane.

To set camera keyframes very quickly:

1. Create a camera from the view - select the camera button - then accept it straight away.

2. Under view properties dialog box, turn on 'Camera/tracking/track always'.

3. Turn on RECORD in the animation slider bar. (This is important - use "track always" with caution when NOT in animation recording as it saves the view position to the camera at all times.)

4. Move the time slider along halfway. Move the camera with the Alt/Right mouse button.

5. Move the time slider to the end. Move the camera with the Alt/Right mouse button.

6. Turn off RECORD. This has set 3 keyframe positions for the camera. Make sure you TURN OFF "track always" now before moving the view again.

7. Move the time slider and the view pans through the 3 keyframes you just set.



Note

Keep the movements relatively small between keyframes (<45 degrees per key) to keep the results predictable. The first keyframe is created automatically at frame 0.

One can save "selection sets" of meshes as separate objects by using the "group" tool. Edit the mesh - select the points - select "group" - accept.



Note

One can have as many sub-groups as required. Separate groups can have the same points inside.

To get soft shadows, create a point light and turn up the "size" field. Use the "Quality" setting to find a good balance of speed vs. output quality.

Boolean objects can be easily edited and animated. Simply turn off "wire invisible" of sub objects then move/size/rotate them.

To "depth" (3D) snap points - use the comma (,) key in conjunction with dragging.

To draw on an object - use "surface input" under view properties/input. This allows the user to draw curves on surfaces.

You can use the unified creation tools to create an SDS object that can then be edited. For example, SDS cubes and SDS rectangles are good starting points.

Use can convert NURBS meshes to SDS objects using the To SDS tool in the 'Construction' tab of the toolbar. Polygonal NURBS meshes are best for converting to SDS objects. Keep SDS objects as simple as possible at all times.

Troubleshooting

The ground plane is not visible

- Make sure that perspective projection is on ('Camera/Perspective' in view popup menu).
- Check that 'View properties/Input/Orientation guides/Ground plane' is set.
- Turn 'Grid draw' option off (disables ground plane drawing to avoid visual confusion between the two).
- Select 'View Orientation/Reset' from the view popup menu.

Cannot RMB-Ctrl-scroll a toolbar

- Move the mouse pointer over a toolbar area that does not activate a button, and check that 'Scrolled' is set using the toolbar's popup menu.

A window is missing

- Check that the window in question is set in the 'View' pulldown menu.
- Make sure that the window is not in a minimized state.

- Check that the window is included in the 'Windows' folder of the program installation (in case you have accidentally moved/deleted the file - if so, you may have to re-install suitable parts of the program).
- The window position may be outside the visible screen after loading a project from a higher resolution or a dual monitor system. To recover, add the startup parameter '-forcetoscreen' to the program's desktop icon or set the respective option using the Startopts program.

The screen is not updated correctly in OpenGL when moving/closing windows

- Update the driver library of your OpenGL accelerator.
- Switch to GDI mode and then back to OpenGL.

The program occasionally crashes in OpenGL mode

- Update the driver library of your OpenGL accelerator.
- Use the native wireframe drawing instead of OpenGL.

Geometric objects

Cannot see an object or parts of the object are not visible

- Open the property window and make sure that 'Gen/Invisible in real time rendering' and 'Wire/Hide Geometry' are not set.
- Select the object and use 'Camera/Auto Focus' from the view's popup menu. The object may be too far or behind the camera (outside the clip range).

Part of a fractal plant is missing and an error message 'maximal plant complexity reached' appears

- Open the property window, select the plant and increase the Maximal complexity value.

Fractal plant's UV coordinates are not continuous (form a checkered pattern)

- Open the property window, select the plant and enable the Set UV option.

Boolean operations

Components of a Boolean operation are not visible

- Make sure that none of the Boolean sub objects has 'Property Window/Gen/Hollow' set.
- Verify that use of 'Volume Inverted' object property is consistent. This option may become unintentionally set when experimenting with Booleans and moving objects around in hierarchy.

A surface renders with a noisy random pattern

- You have performed a Boolean operation where two cutting surfaces overlap exactly. Move either of the components slightly (1 micrometer is enough), make sure that the 'drill' is long enough, etc.

Rendering

An object is missing from ray traced images

- Select the object and check that 'Property Window/Gen/Invisible in real time rendering' is not set (check also the parent levels).

A parallel mapped texture renders with a noisy random pattern on the target surface

- If you are using 'Finite' options of the parallel mapping, the target surface should be properly inside the mapping cube.

File rendering shows a different image than the view window

- Make sure that the project includes a camera object, and that the camera and view orientations match. Set 'Current' option for the desired camera object, in case there are multiple cameras included.

The file rendered image appears stretched

- Verify that the aspect ratio of the current camera object and the output resolution are consistent (taking into account the pixel aspect ratio of the planned target media).

Cannot see through transparent objects, or parts of transparent objects appear black

- Make sure that 'Recursion Depth' of the rendering settings you are using is high enough. For example, 'Speed over quality' does not show transparency.

Cannot change the color of a backdrop object

- A backdrop surface is usually 'self-illuminating' - light sources do not affect it. To change the color, open the property window and go to the 'Col' tab. Select 'Illumination' from the 'Attribute' property list and define a new illumination color using the gadget below the property list.

NURBS curves are not visible in rendering

- Check that 'Property Window/Gen/Invisible in photo realistic rendering' is not set.

Rendering halts immediately and gives an error message 'channel xxx not found'

- Go to the Channels tab of the select window, create a new channel and rename it as specified in the error message.

Network rendering

Cannot use network rendering

- Windows95/98 have limitations in TCP/IP services required by network rendering.

Network rendering is slow

- Save texture maps as JPG images for fastest network render times.
- Check network configurations - the client may spend time in unsuccessfully trying to connect to a server.

A server system in network rendering cannot render the current project

- Check the texture file paths of the project. For example, a hard disk may be mapped to another letter in the server system as in the client.

Materials

A transparent material renders white

- Add 'Color=Constant(0,0,0) to surface properties. The default diffuse color is always white, and you have to eliminate diffuse shading.

Cannot see effect of a fog-like material

- Fog effects are heavily scale dependent. A fog material that looks good when mapped to a one meter wide object is practically invisible when mapped to a candle flame. Adjust the 'Turbidity' or 'Density' parameter of the fog.

- Check the rendering settings. Volumetric options should be set and recursion depth should be high enough (try a value greater than 2).

Animation

The camera does not keep its focus in the desired point after key framed rotations

- Set the pivot point of the camera to the desired center of rotation before key framing the camera.

A key frame-animated object rotates to the opposite direction to what was intended

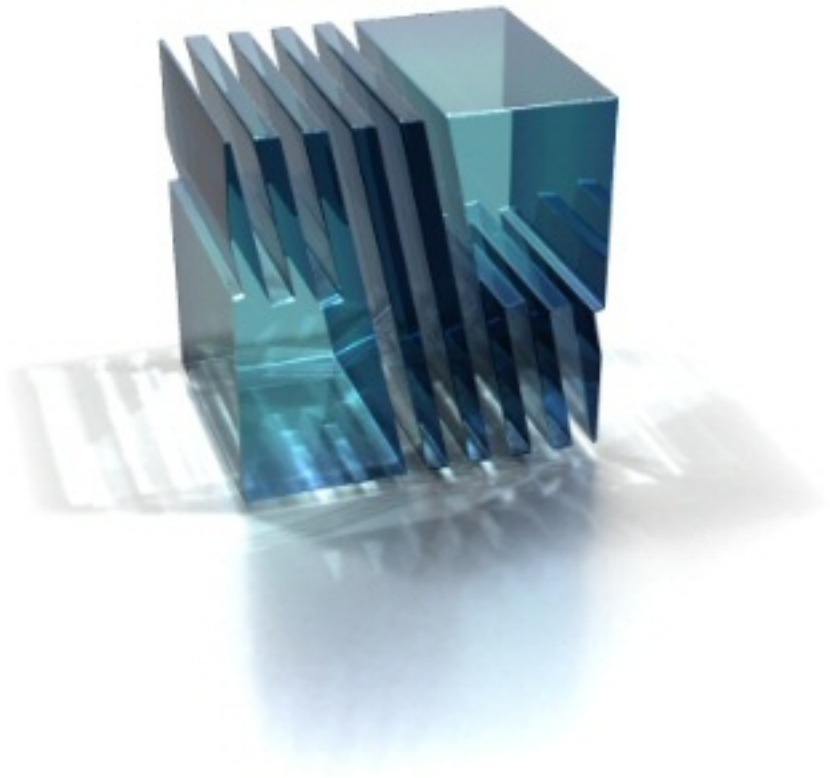
- Add a key frame in the middle. Rotation chooses the shortest way to new positions and the new key shows the right way.

Simulation does not affect the objects

- Check that the parent level has 'Properties/Spec/Simulation' enabled.

An object does not follow exactly a given path

- Path animation keeps the initial offset between the object and the path. Select the object and the path, then apply the Unmap tool of the 'Lattice Mapping' tool tab. Then move the object to the correct position (use point snapping if necessary). Re-map the object to the path.



Chapter 14. Imprint

Realsoft Graphics Oy
Polvikatu 3 C 22, 33100 Tampere, Finland
http://www.realsoft.com

Realsoft 3D User Manual Seventh Edition, April 2014. Copyright 2014 Realsoft Graphics Oy. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without permission in writing from Realsoft Graphics Oy.

All images are copyrighted by their respective owners. All rights reserved.

Front Knife image Copyright (BT)-grafik
Learning the Basics Copyright (BT)-grafik
Modeling Copyright (BT)-grafik
Subdivision Surfaces Copyright Gunnar Radeloff
NURBS Copyright Gunter Hagedorn
Modeling Buildings Copyright Pixel Perfect
Materials Copyright (BT)-grafik
Rendering Copyright Gunnar Radeloff
Post Effects Copyright Gunnar Radeloff
Animations Copyright Gunnar Radeloff
Simulations Copyright (BT)-grafik
User Interface Copyright (BT)-grafik
Scripting Languages Copyright Martin Weber
Miscellaneous Topics Copyright (BT)-grafik
Design Copyright (BT)-grafik



Index

Symbols

3D Painting, 225
3D Snapping, 76

A

Acceleration, 394
Accuracy, 78
Airplane, 417
Alpha Channel, 305
Aluminium, 250
Ambient Light, 292
Anchoring, 389
Angle Constraints, 386
Angle Grid:, 69
Animate UV, 382
Animated Material, 375
Animations, 353
Atmosphere, 296
Available GUI Objects, 421
Available GUI Objects Window, 421
Available Objects Window, 421
Average of Materials, 255

B

Background Color, 107
Banana, 164, 172
Batch Rendering, 396
Bezier, 162
Bi-Rail, 174
Bird, 133
Blending Materials, 218
Blur Effect, 337
Boolean Operations, 79
Boundary Surface Solids, 82
Breaking Curves, 158
Breaking Meshes, 166
Bubbles, 403
Bump Mapping, 239

C

Cabinet Group, 210
Camera, 105
Camera Backdrop, 106
Camera Lens Flare, 107
Camera Tracking, 108
Cauge, 114
Caustics, 317
Centripetal, 161
Chord Length, 161
Choreography Parametrization, 391
Clip Mapping, 262
Clouds, 295
Coffee Cup, 122
Collect External Files, 73
Collision Deformations, 409
Collision Detection, 409
Color Animation, 354
Color control, 276
Compass Macros, 430
Compass Simulation, 413
Compositing, 340
Concatenate, 161
Constraint Script, 113
Creases, 147
Creator, 401
Cross Sections, 175
Current Project, 71
Current Working Project, 71
Curve Order, 157
Curve Weight, 160
Curves, 155
Curves from Surfaces, 189

D

Dart, 415
Decorating, 23

Degree Elevation, 168
Depth of Field, 108
Dirt, 272
Displace Mesh, 191
Distributed Rendering, 285
Doors, 21
Drag Box, 73
Drag Box Selector, 73
Dragging, 73

E

Editing Key Frames, 355
Elasticity, 410
Euler Angles, 380
Event Driven Animation, 391
Explosions, 404
Extrude Curves, 177

F

Face Materials, 150
Fade Mapping, 263
Falloff, 291
Field Evaluator, 114
Field Rendering, 300, 383
File Rendering, 282
Fillets, 185
Fire, 297
Flashlight, 289
Floors, 207
Fluid, 118
Fluid Dynamics, 414
Fog Examples, 242
Font Curves, 109
Font Tool, 187
Foot Steps, 390
Frame Count, 356
Frame Range, 358
Freehand Curves, 156
Freehand selector, 76
Friction, 411
Fur, 245
Furniture, 25

G

Galaxy, 408
GI in Animations, 320
Glass, 248
Global Fog, 378
Global Illumination, 311
Gravity, 406
Grids, 78
Ground Floor, 207
GUI Customization, 421
GUI of VSL Material, 238

H

Hair, 44
Handles, 66
Harry, 29
Hierarchical Animations, 366
Hipped roof, 204
Hot Keys, 460

I

Illumination Map, 306
Image Vectorization, 188
Indirect Illumination, 309
Inertia, 411
Instances, 112

J

JavaScript, 434

K

Key Bindings, 423
Key Driven Animation, 391
Key Frame Animation, 353
Keyboard Shortcuts, 460
Kitchen, 210

L

Lake, 266
Landscape, 27
Lasso, 76
Lasso Selector, 76
Lattice Mapping, 85
Layers, 453
Leaves, 99
Lens Flare Animation, 345
Level Material, 224
Light Falloff Curve, 246
Lighting, 288
Link Object, 120
Localization, 431

M

Macro Button, 428
Macro Keys, 423
Macro Menus, 428
Macros, 90
Magnetism, 413
Magnifying glass, 318
Map2Obj, 153
Mapped Shadows, 307
Material drag and drop, 213
Material Mapping, 213
Materials, 213
Matrices, 449
Matte Shadows, 245

Measuring Units, 78
Metaball, 115
Motion Blur, 299, 383
Move Handle, 67
Multiple Cameras, 106
Multiple Floors, 207
Muscles, 393

N

New Classes, 455
Numeric Modeling, 76
NURBS Mesh, 164
NURBS Parametrization, 161

O

Orbiting Planets, 406
Outline Rendering, 314

P

Paper, 244
Parametric Objects, 192
Particle Effects, 335
Particles, 93
Path Animation, 368
Pendulum, 407
Periodic Animation, 364
photon map in Animations, 328
Pivot, 68
Pivot Point, 68
Plasma, 242
Point Snapping, 73
Post Shading, 347
Pot, 256
Previewing Materials, 264
Project, 71
Propeller, 145

Q

Quaternions, 379

R

Rational SDS, 144
Rebound, 412
Render Box, 284
Rendering Settings, 282
Rigging, 387
Rigidity, 410
Roof, 24
Roof Line, 204
Roof Modeling, 204
Rotate Handle, 67
Rotated Surfaces, 170
Rotation Handle, 67
Rotation Handles, 67
Rotations, 379
Rotator, 109

S

Saving a project, 72
Scale Handle, 67
Scale Handles, 67
Scaling Handle, 67
Scaling Time Lines, 355
Scanline, 301
Scope Mapping, 263
Scripting, 433
Scripting Languages, 433
SDS Compass, 126
SDS Handles, 128
SDS Knife, 153
SDS Rounding, 148
Shadows in Fog, 243
Sharp Corner, 158
Shinbone, 366
Simulations, 397
Sink, 150
Skeletons, 384
Slide Projector, 246
Smoke, 338
Snapping to Curve, 76
Soft Shadows, 290
Special Light, 293
Spin, 400
Steel, 248
Subdivision Surfaces, 122
Swap, 169
Sweep, 173
Swimming Fish, 370
Symmetric Modeling, 139

T

Technical Drawing, 350
Texture Mapping, 221
Thighbone, 366
Time Code, 356
Timing, 371
Toolbar Editing, 420
Torus, 171
ToSDS Tool, 135
Transformation Handle, 66
Transformation Handles, 66
Translation Handle, 67
Translation Handles, 67
Trim Booleans, 180
Trimming, 178
Troubleshooting, 469
Tube Object, 111
Type Casting, 251

U

Unified Tools, 60

Uniform, 161
Uniform Scaling, 69
User Interface, 420
UV Editing, 151

V

V3 Material, 224
Vectors, 447
Velocity, 400
Vertex Variables, 138
Video System, 356
Visible Lights, 296
Volumetric Rendering, 293
VSL Examples, 233
VSL Intro, 233
VSL Material, 223
VSL Procedures, 249
VSL Wizards, 238

W

Wall, 275
Waterfall, 382
Weight of Choreography, 364
Welding, 182
Window, 192
Windows, 21



Reference Manual

Seventh Edition

Reference Manual

Publication date 1985 ... 2019
Copyright © 1985 ... 2019 Realsoft Graphics Oy

All rights reserved. This documentation contains proprietary confidential information and trade secrets of Realsoft Graphics Oy.

THIS SOFTWARE AND RELATED MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN NO EVENT SHALL REALSOFT GRAPHICS OY BE LIABLE FOR ANY DAMAGES CAUSED USE OF THIS MATERIAL.

Table of Contents

1. Realsoft 3D 1
    - Models 1
        - Channels 1
        - Choreography Classes 2
        - File Rendering Configurations 5
        - Geometric Objects 9
        - Grids 87
        - Images 87
        - Materials 88
        - Output Objects 120
        - Post Particle Effects 125
        - Post Image Effects 128
        - Rendering Settings 138
        - View Models 146
        - Animation Object 146
    - Startup options 147
    - Scripting 149
        - Built-in JavaScript Classes 149
    - Tools 176
        - 3D painting Tools 176
        - Analytic Tools 181
        - Boolean Tools 188
        - Building Construction Tools 189
        - Construction Tools 192
        - Creation Tools 204
        - Particle Tools 207
        - Deformation Tools 210
        - Foot Print Tools 216
        - Geometry Manager Tools 217
        - View Control Tools 218
        - Lattice Mapping Tools 222
        - Light Source Creation Tools 226
        - Material Mapping Tools 228
        - Miscellaneous Tools 232
        - NURBS Curves and Surfaces 237
        - Plant Tools 255
        - Subdivision Surface Tools 256
        - Simulation Tools 273
        - Transformation Tools 277
        - Triset Tools 281
        - Unified Creation Tools 282
    - User Interface 291
        - Geometry Managers 291
        - Macro GUI gadgets 292
        - Windows 292
    - Windows 296
        - Advanced Selection Window 296
        - Animation Settings 297
        - Animation Window 298
        - Attributes Window 300
        - Choreography Window 300
        - Color Window 304
        - Curve Editor 305
        - File Rendering Window 308

Info Window 308
Material Browse Window 308
Numeric Window 309
Object Browse Window 310
Options Window 310
Property Window 312
Scripting Window 313
Select Window 313
Status Window 327
Tool Bar 327
Tool Control Bar 327
UV Editor 328
UV Editor Properties 330
View 330
View Control Bar 338
View Properties 341
2. BatchRen 350
Description 350
Startup Options 350
Examples 351
3. BatchRenW 352
Description 352
Using BatchRenW 352
4. RendD 353
Description 353
Startup Switches 353
Examples 355
5. RendW 357
Description 357
Pull-down Menus 357
Startup Options 357
The Control Window 358
6. ShowImg 360
Pull-down Menus 360
Startup Parameters 360
7. Software Updater 362
Description 362
Startup options 363
8. StartOpts 364
Description 364
Startup options 364
9. RsView 366
Description 366
Pull-down Menus 366
Startup Parameters 367
10. Plugins 368
Index 372



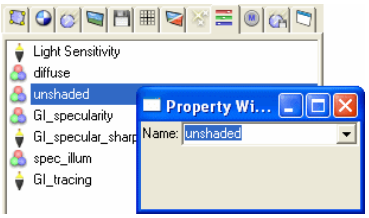
Chapter 1. Realsoft 3D

Models

Channels

Channel objects are very simple. They only specify a name and a the amount of floating point valued sub channels. However, channels play an important role when building customized rendering styles and solutions. By using channels, you can attach an unlimited amount of attributes, which are available in material shaders and rendering engine modules, to complete objects as well as their geometry points.

Currently three types of channels are available:



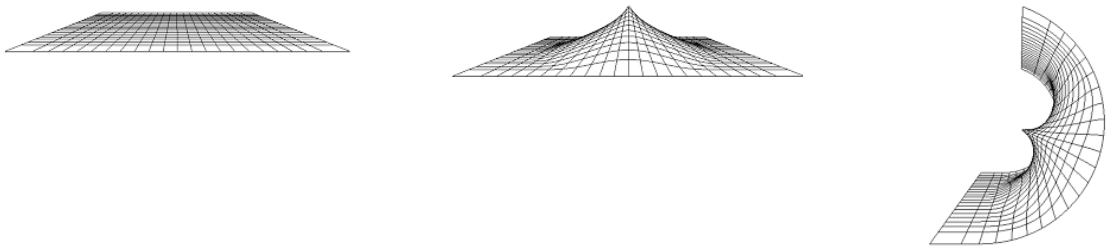
- **Float**: a single floating point value channel
- **Color**: a RGB triplet of floating point values
- **Vector**: three values for the x, y and z dimensions

Choreography Classes

Realsoft 3D choreography system is based on *construction stack* idea. A number of choreographs can be attached to an object. Choreographs are executed one by one and each choreography object modifies the state resulted by the previous choreography.

For example, a radial deformer can be used for generating a bump to a mesh. Then a 'bend' deformer can bend the mesh. Because choreographs modify the state resulted by the previous choreography, also the bump get bent.

When any of the choreography parameters is modified, the object is rebuild accordingly.



The first choreography defines the initial shape of the mesh. The second choreography (radial deformer) generates a vertical bump to the mesh. The third choreography bends the mesh and the bump with it.

There are two ways in which the user can modify choreography attributes:

- by modifying the actual target object, in which case the *current choreography* attemps to update its attributes accordingly. Not all choreographs support this, however. For example, you cannot modify noise parameters by editing the ripples of the target object. Keyframe choreography does support this method, by editing its animation curves according to the users actions (this process is called key frame creation).
- by editing the choreography attributes through the choreography window.

Assign

This choreography object is typically associated with the root level choreography and named 'init'. Its task is to maintain the initial state of the animated attributes.

To modify the initial values, make the init (the first) choreography the current choreography, and then modify the target object attribute.

Pose

Pose choreograph is similar to Assign choreography in that its job is to keep track of the current values of attributes. However, whereas Assign choreography saves the absolute values of attributes, Pose choreography stores attributes in relative way. Pose choreography allows you to 'translate' attribute values.

You can update the current Pose values simply by making the pose the current choreography and then modifying the attribute in question.

Pose choreography is special in that it is automatically created by the system. If animation recording is off and you modify an attribute that is already controlled by a choreography that doesn't allow target editing, a pose choreography is created to represent the attribute changes.

Keyframe

Keyframe choreography is also special in that it is automatically created by the system. If animation recording is on and you modify an attribute, key framer is created to record the attribute values in time sensitive way.

Keyframer does this maintaining animation curves. Every time you modify the target object, the keyframer inserts new point to the corresponding curve (inserts new key frames). Correspondingly, when time is changed, the choreography evaluates (interpolates) its animation curves and assigns the values back to the target object.

Noise

Noise choreography applies 3D fractal noise to the target attributes. Noise differs from assign, keyframe and pose choreographs in that its cannot update any of its parameters when the target object is modified. Noise choreography is handy for adding some randomness to animations.

Script

Script choreography can be used for controlling animated attributes via scripting languages, such as JavaScript.

Scrip choreography defines the following variables for this:

```
P - target attribute (in target object's local space)
Time - current time (or the current input attribute)
Input - input object, or null if input is time
Target - target object (whose attribute 'P' is).
```

Because target attributes are all referred through the variable 'P', scripts created this way are reusable.

The type of the 'P' variable corresponds to the type of the target attribute (integer, number, vector, vector4, axis, coordsys).

The following examples assume basic knowledge about JavaScript.

Example 1 - a constraint:

Let us imagine you have animated the 'Translate' attribute and want to write a script that constraints the value of the attribute's y component above the ground level. You can achieve this by attaching the following script choreography to the Translate attribute.

```
if(P.y < 0)
    P.y = 0;
```

Example 2 - a circular path animation:

To make an object to move along a circular path in its x-y plane:

```
// uses JavaScript Math object to evaluate a circle
P.x += Math.sin(Time * 2 * PI);
P.y += Math.cos(Time * 2 * PI);
```

Example 3:

This example shows how one can implement a path animation using scripting.

Create a script choreography for the Translate attribute, with the following JavaScript program:

```
    uvw = new r3Vect(Time, 0, 0);
    p0 = new r3Vect();
    p1 = new r3Vect();

    // evaluate two points from the input object: one with time=0 and
    // another one corresponding the current time
    Input.EVALUATE(new r3Vect(0, 0, 0), R3SPACE_ABSOLUTE, p0);
    Input.EVALUATE(uvw, R3SPACE_ABSOLUTE, p1);

    // compute translation: delta = p1 - p0
    delta = p1.sub(p0);

    // we need to map 'delta' from abs space to target object's parent space
    // because target objects translation is defined in its parent's space
    parent = Target.GetParent();
    parent.POINTTOOBJSPACEL(delta);

    // add delta to the target attribute
    P.fadd(delta);
```

Now, bind the script choreography to desired path object, say a nurbs curve and play the animation to see how the target object follows the curve.

Example 4 - a point deformator:

A typical deformator works as follows: it first maps a point to a simple geometry, such as a rectangle. Then it use the result to evaluate a point from another, typically more complex, geometry.

For example, let's imagine a deformator that turns any planar mesh to a sphere.

Such a deformator can be represented as a level object consisting of two sub objects: a rectangle and a sphere.

```
   deformator
    /     \
 rect    sphere
```

Then we can use this deformator object as an input for a script choreography, with the following script:

```
// this chor must be bound to an input object
// that has two sub objects defining desired deformation

if(Input) {

    // map target point to abs (world) space
    p = new r3Vect(P.x, P.y, P.z);
    Target.POINTTOABSSPACE(p);

    // fetch sub objects from the input object
    from = Input.GETSUBBYORDNUM(0);
```

```
    to = Input.GETSUBBYORDNUM(1);

    // map point to from's param space
    uvw = new r3Vect();
    from.MAPTOPARAMSPACE(p, uvw);

    // evaluate out through to's param space
    to.EVALUATE(R3SPACE_ABSOLUTE, uvw, p);

    // map point to target space
    Target.POINTTOOBJSPACE(p);

    // add deformation effect to the target point
    P.fadd(p);
}
```

Now save the choreography and you can later load and attach it to any target and input object. You can replace the 'sphere' with any geometric objects. For example, if you prefer a deformation that maps a rectangle to a cone, just use analytic cone instead.



Note

Never modify attributes through 'Target' object. For example, never call:

```
Target.SetMatrixTranslate(delta);
```

If you need to access an attribute of the target object, make the attribute animateable and access it through the 'P' variable.

Generic

Generic constructor doesn't define any animation effects by itself. Instead, it asks the associated input object (typically a geometric object, such as a skeleton) to apply the effect.

Generic constructor cannot be created directly. It is automatically created when the actual deformator object is created and bound to its target objects.

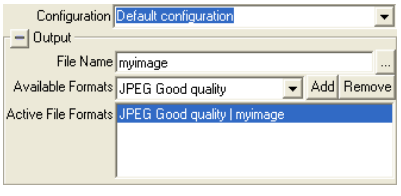


The points of 'arm' (SDS object) deformed by 'left arm' (skeleton)

Skeleton, bend, radial deformer, 3d lattice and many other geometric deformators are linked to the choreography system this way.

File Rendering Configurations

File rendering configurations combine most settings and other data components required when rendering a 'final' image into a disk file. For example, image resolution and rendering quality settings are part of the configuration data. Often, in addition to selecting a suitable file rendering configuration, the user only has to make sure that a suitable camera object is included in the scene.



The 'Configuration' field contains a list of file rendering configurations. Selecting 'New' from the end of the list adds a new item to the list. It can be renamed and edited as appropriate, and saved to disk for later use.

The file rendering system supports simultaneous output to several files or other output targets (a preview window etc. may be also included). The following output controls are available:

- *File Name:* The name and path for the output item selected in the 'Active File Formats' list below. Changing the string and hitting the 'Enter' key updates the respective file output. The name string is also used when adding new formats to the active list.
- *Available Formats:* A list of currently loaded/created output objects. You can pick a format and click the 'Add' key to insert a new output file to the active list below this control.



Note

If a suitable format is missing, you can create new output objects at the select window's 'Output Objects' tab.

- *Active File Formats* This list shows the total output file configuration. Each item in the list is a pair of a format name and an suitably abbreviated file name. Items can be removed by activating them and using the 'Remove' popup menu. The file name part of an item can be changed by activating the item, then changing the 'File Name' string and hitting the Enter key. Note that changing the 'Available Formats' field selection does not change any item in the active list.

File formats can be added also by drag and dropping them from the select window's 'Output Objects' tab into the active format list.

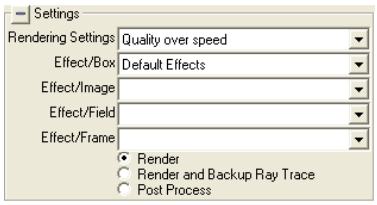


Note

You cannot add the same format twice to the active list, because the items on the active list are identified by the output object name. If you, for some unusual reason, truly want to output two identical files, create a duplicate of the format object at the select window and rename it.

Settings

Rendering quality settings and post processing settings can be select as follows:



- *Rendering Settings:* The render settings object which defines ray tracing quality and other photorealistic rendering options.
- *Effect/Box:* The post processing configuration (post image) which is executed for all rendered images and image sub regions.
- *Effect/Image:* The post processing configuration which is executed for complete images. If an image is rendered by dividing it to sub boxes (e.g. when using network rendering), the boxes are processed with this configuration after assembling them into a complete image.
- *Effect/Field:* The post processing configuration for rendered fields.



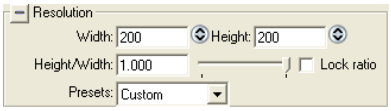
Note

If field rendering is turned off, these effects are executed for all rendered images after combining possible motion samples.

- *Effect/Frame:* The post processing configuration for complete frames. Executed after possible sub fields have been assembled into a frame.
- *Render/Backup/Post Process:* With this control, you can activate rendering with an additional ray trace backup file. After full rendering, post processing effects can be readjusted and rendered multiple times using the backup file.

- *Walkthrough mode:* If this option is enabled, the scene is treated as static (animations will not work). The full scene is sent to the render engine only in the first rendered frame, and after that, only camera changes are considered. This can speed up rendering remarkably, when the scene is complex and contains procedural objects such as fractal trees.

Resolution



These controls define the pixel resolution of the outputted images.

- *Width:* Image accuracy in horizontal direction (pixels)
- *Height:* Image accuracy in vertical direction
- *Height/Width:* Image resolution ratio.



Note

This is not necessarily the same as the image aspect ratio, which also depends on the pixel aspect ratio. Pixels are often not squares. For example, all common video standards use non-square pixels.

- *Lock ratio:* Locks the ration so that if you change one dimension, the other becomes scaled respectively.
- *Presets:* A list containing some common industry standard resolutions.

Box Rendering



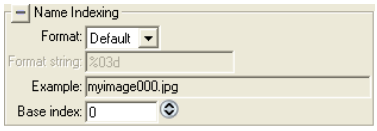
The 'Render Box' controls limit the rendering to a sub area of the total image. For example, it may happen that after spending one whole day for high resolution rendering of a complex scene, the user notices that the color of a small object at the top left corner was wrong.

Instead of re-rendering the whole scene, the user can render one quarter of the total image width and one quarter of the total height around the object to a new file. The position of the new sub image is known accurately and therefore it can be pasted over the original image using a paint or image processing program.

The controls are:

- *X:* The horizontal position of the top left corner of the render box (measured in pixels)
- *Y:* The vertical position of the top left corner of the render box
- *Width:* The width of the box (pixels)
- *Height:* The height of the box
- *Active:* Activates the sub box rendering feature

Name Indexing



The 'Name Indexing' controls define how the file name is generated from the actual file name and the current frame number. Rendered images are named as follows:

[filename][index].[extension]

For example:

myimage000.bmp

The first portion is the file name defined by the user (such as d:\mypics\test). Index is the current frame plus the base index definedthrough the *'Base index'* field. The number of digits in the index portion is automatically extracted from the number of frames in the animation. For example, if the length of the animation is 120 frames, the index portion contains three digits. The extension is determined by the selected file format.

The indexing controls are:

- *Format:* Selects between the default (zero padded) indexing and a customized, user defined indexing. The default indexing format is flexible enough for most cases. If a more exotic indexing convention is desired, it can be achieved by selecting 'Custom' to the 'Format' field and entering a suitable string to the 'Format String' field. The format string follows C language 'printf' conventions.

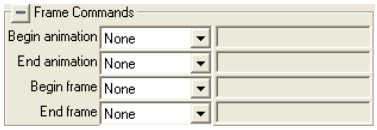
- *Format String:* This control shows the format string that is used for generating the index portion of the filename. If 'Format' is set to 'Custom', you can define an arbitrary format string to this gadget. For example, the string '%d' outputs indices without zero padding: frame0.tga, frame1.tga, … , frame99.tga…. '%05d' would output: frame00001.tga. '%03d' outputs: frame001.tga, etc.

- *Example:* shows a complete sample filename generated by the current settings.

- *Base Index:* Animation frame indices start from zero. If you want to start output file indexing from one (frame001.tga, frame002.tga, …), set 'Base index' to one.

Frame Commands

The 'Frame Commands' area contains four command gadgets. The commands can be defined using any of the installed scripting languages. In addition to this, DOS (or shell) scripts can also be used.



The following commands can be defined:

- *Begin animation:* This command is executed once in the beginning of animation rendering. For example, a special hardware for animation recording can be turned on.

- *End animation:* Executed once when the animation is ready. This can be used for clean up actions.

- *Begin Frame:* Executed before rendering of each frame.

- *End Frame:* Executed after each rendered frame. For example, the rendered image can be recorded to a single frame recorder.

The frame command can access the file name via the following predefined macros:

> *[image]* - Full name of the rendered image, for example 'd:\images\test000.bmp'.*[path]* - Path portion of the file name without the last path separator, such as 'd:\images'.*[base]* - Base name (i.e. path and extension removed). For example 'test'.*[ext]* - Extension without the dot. For example 'bmp'.

For example, if the file name is d:\realsoft3d\images\test.bmp:

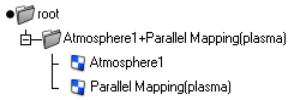
> [path] - results in 'd:\realsoft3d\images' [path]\[base].[ext] is the same as [image] - 'd:\realsoft3d\images\test.bmp'. [path]\[base].tga results 'd:\realsoft3d\images\test.tga'

Geometric Objects

Geometric Primitives

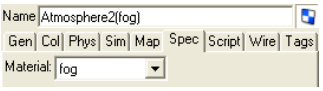
Atmosphere

The atmosphere object has no geometry. It generates a world filling volume, which can be used for global fog effects. The atmosphere object allows direct material selection from the **Material** field of its property tab. The material selector shows all materials of the current library that define volumetric properties. Note, however, that since the atmosphere has no geometry, only gas materials with constant properties should be assigned from the 'Material' gadget.



You can map a gaseous material to an atmosphere object ...

If the material includes turbulence or other density variations, it is better to map the material to the atmosphere using for example parallel mapping.

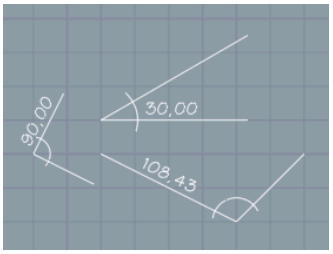


... or assign it directly as an object property

Angle Measure

The angle measure object is displays the angle between its two control lines. You can add angle measure objects to you scenes to show information about important angles.

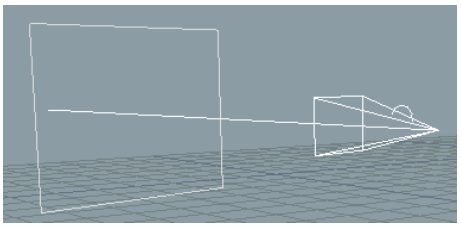
The properties of the angle measure object are:



- Photorealistic rendering method defines how the geometry is rendered.
  - **Ray Trace** - The angle measure and its dimension text is rendered using ray tracing
  - **Scanline** - The measure is rendered using scanline method
  - **Post Process** - Rendering is done at post processing stage
  - **Pseudo 3D** - A special correction is applied to the geometry, to obtain a constant appearance the same way as in real-time drawing. Thickness and size of the object will appear constant, regardless of the distance from camera or the viewing angle.
- **Text Size** - The size of numerical digits (per mills of the full image width), which display the angle
- **Thickness** - The radius of the line used in photorealistic rendering (per mills)
- **Threshold** - Threshold for measure drawing. Angle measures, which are seen in a very low angle (viewing direction almost parallel to the angle plane) should not be drawn, because the projected angle looks like a straight line. This threshold controls when clipping occurs. Value 1 draws all angles.
- **Angle** - The current angle

Backdrop

A backdrop object is a plane surface that automatically rotates itself perpendicular to the camera in ray trace rendering and scales itself to fill the image. It defines UV coordinates which run from (0,0,0) at the top left corner to (1,1,0) at the bottom right corner. A backdrop is a real plane surface whose distance can be adjusted using the **Distance** control on the property window's object specific tab.



A backdrop plane, such as all other geometric objects, can be textured (and shaded in general) using VSL materials. The distance can be modified using a ray hit shader, the backdrop can be translucent etc. Backdrops have some special object attributes set by default:

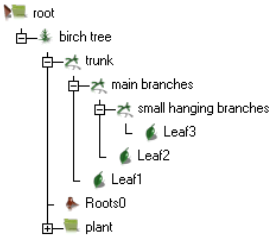
- They are self-illuminating (Property window's Color tab/Illumination channel)
- The diffuse shading is turned off (Color tab/Color channel)
- They do not cast or receive shadows (General tab)

Branch Object and Root Object

Branch object defines how a plant grows using a number of growth parameters. It generates a set of sub branches once per 'year'. Age is a relative concept here; you can consider the units as days, if the plant grows rapidly. Sub branches follow the same growth rules as their parent, unless child branch objects are present. In other words, plant's sub hierarchy defines the structure of the generated geometry in a natural way.

A branch object can have multiple child branches. When a branch decides to grow a sub branch, it seeks for active sub branches, and only one (the first active) sub branch is invoked to grow. For example, a certain sub branch type 'leaf branch' may be active during the first 10 age units, after which another sub branch 'flower branch' becomes active.

The plant root object is identical with the branch object, except that it grows to the opposite direction. As the name suggests, it adds roots to the plant.



Complex nested sub branching structure

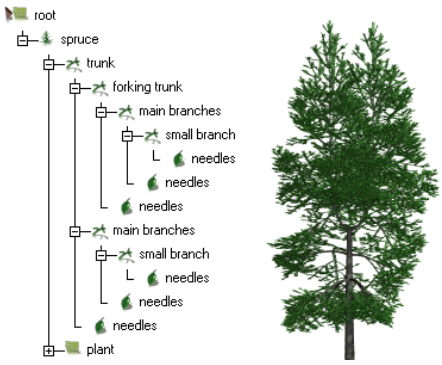
Model Type: Defines the type of the generated geometry. The options are:

- **Analytics** - Branches are built using cones, cylinders and spheres. This kind of geometry is very fast to render, and also quite memory efficient if branches are relatively straight (i.e. when the Density parameter is 0.5 .. 1.0). The UV Coordinates of this model type are limited: U component is constant zero, and V grows stepwise per geometry item.
- **Particles** - Branches are built from 2D particle segments. This kind of geometry is also fast to render and more memory efficient as analytic primitives. The real time geometry is drawn using thin unshaded lines, which makes the tree very fast to manipulate in modeling. Particle geometry does not support UV Coordinates at all in the current version.
- **NURBS** - Branches are ray traced NURBS curves. The NURBS option has no UV Coordinate support (except the native UVs of all NURBS curves), but the lowest memory consumption in ray tracing, reasonably fast rendering and very fast (wireframe only) real time drawing. NURBS curves do not currently support pointwise attributes, which means that branch diameter control is limited to the start and end points.

- **SDS** - The most detailed way to model the branches. Features: highest memory consumption, accurate UV support, possibility to apply displacement mapping etc.

Own Hierarchy: If set, the branch geometry generated by the selected object is placed into a folder dedicated for this branch type. You can then place texture maps and other material attributes, which are specific to this branch type, into the same folder.

Active by Cut: If set, the branch starts growing only when the parent branch is abnormally truncated by its Tail Cut Age or Rnd Cuts attributes. Many trees start growing a new trunk from one or two biggest branches if the actual trunk becomes cut in a storm. The new trunk usually has some special properties, such as a strong desire to grow upwards.



A spruce with a forking trunk

Activation: The age when this branch type becomes active. Inactive branches do not grow. The first active sibling branch defines the growth parameters for a new branch.

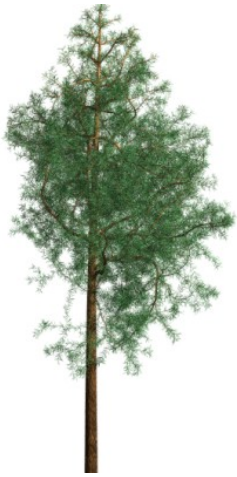
Deactivation: The end of active growth period.

Rnd Activation: Probability that the activation of the branch fails.

Age: The maximum age of this branch type. A branch's age cannot exceed the remaining growth age of its parent after the branching event. A negative value -1 automatically sets an age, which is exactly the remaining growth age of the parent after branching. Real world trees grow this way. However, it is often necessary to limit the age of sub branches of old trees, because the complexity grows exponentially by age.

Rnd Age: Randomizes branch age. The value 1.0 will set a random age between 0 ... Age.

Branching Age: The length of the branchless section in the beginning of the branch. If the value is zero, the branch starts growing sub branches immediately at its root. Some trees, such as pine, grow a long trunk, which has sub branches only at the topmost part, because oldest branches die and fall down. Use this parameter to simulate such trees.



An old pine tree has only the top branches left. Branchless value of the trunk is 8 years.

Tail Cut Age: Cuts the branch tail after the given number of years.

Rnd Cuts: The probability that the branch ends at a branching point. Only the sub branches continue their growth. This kind of growth happens when a branch gets damaged for some reason (gardener cuts the branch, a disease or insect kills it etc.)



The tree on the right has a high Random Cuts value. Its branches often split into two or three side branches.

Sub Count: The number of sub branches, which start growing from a branching point.

Rnd Count: Randomizes the count. Value 1.0 selects a random number from the interval 0 .. Sub Count.

Sub Thickness: The relative thickness of a branch vs. its parent.

Rnd Thickness: Randomizes the relative thickness value.

Thickness/Age: Changes the relative thickness by the age when branching happens. If the value is high, the relative thickness of oldest branches is smallest. For example, a spruce has a thick trunk. Even the strongest main branches are thin compared to the trunk. However, the trunk gets thinner towards the top, and the upmost branches are quite as thick as the trunk from which they grow.

Bumpiness: Moves the points of SDS modeled plants randomly, making the surface appear bumpy. Other geometry types are more limited with respect to this attribute: Analytic and Particle modeled trees allow only rapid variation of the diameter. Nurbs trees do not currently support Bumpiness.



Bumpy SDS branch

Density: Defines how densely the growth parameters are evaluated and new geometry added to the tree. The smaller the value, the stronger bends branches can make because of gravity and other deformation factors. The drawback is that densely evaluated geometry consumes more memory.

Length: Defines the growth speed of the branch. The value is relative to the length of the parent branch: if Length equals 1, the sub branch grows the same length per year as its parent branch.

Rnd Length: Random variation in the length. A value 0.5 changes the length -50% .... +50 % randomly.

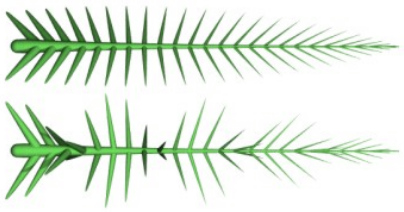
Length By Age: Defines how much shorter/longer the branch segments get every year. The value is multiplies the length, meaning that the factor 1.0 keeps the length the same. Many plants first grow rapidly higher to reach the light from the shadow of bigger plants. When they reach a sufficient amount of light, they target their growth energy to spread branches sideways.

Angle: The angle between the branch and its parent.

Rnd Angle: Random variation of the branch/parent angle.

Angle/Age: Angle reduction by age. Usually oldest branches have a greater angle with the trunk as the younger branches at the top, because the weight of the big branches bends them down.

Twist: Rotates sub branch directions around the branch. Twist option creates spiral wise branching patterns.



The lower branch twists sub branches around itself

Rnd Twist: Randomizes the twist parameter.

Gravity: Guides the branch orientation by gravity. The direction of gravity is defined by the plant's root object's z axis (blue axis line). The higher the gravity, the more branches will hang down. A negative gravity value makes branches to grow upwards (to simulate light attraction etc.).

Gravity/Age: Defines how the effect of gravity changes by the age. The actual gravity is the sum of Gravity and Gravity/Age parameters. For example, a branch end may seek light and grow upwards, whereas a branch start can bend down because of the total mass of the branch.

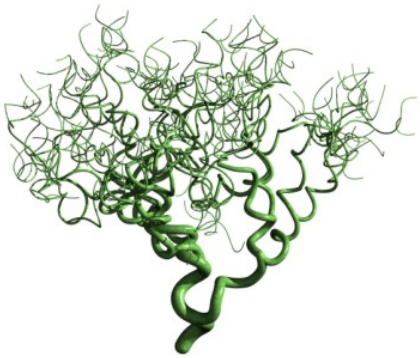


Positive and negative gravity

Side Shift: Turns the branch end to the side. By combining Twist and Side Shift Parameters, you can create spiral-like growth behaviors.

Deformation: Bends the branch randomly. All natural plants have some deformation. High value such as 3 bends the branch strongly.

Deform Scale: Defines how rapidly deformation bends occur. The higher the value, the quicker deformation turns growth direction. A low deform scale value will bend neighbor branches in a similar way.



Side shift = 0.5, Twist=1.0, some deformation

Horizontal Spread: Guides the branch to spread out horizontally around its parent branch. Horizon direction taken from the blue z axis of the object space of the plant root level. Many plants grow this way: the purpose is to catch as much light as possible. Note that will full horizontal spreading level 1.0 and Sub Count greater than 2, branches may overlap.



The branches of the tree on the left grow to random directions. Horizontal spreading makes the branches of the tree on the right to grow sideways.

Spread by Age: Changes the horizontal spread by the age when the branch started to grow. The actual spread parameter is the sum of the constant spread and the age dependent spread. If Horiz. Spread=0, Spread by Age=1.0, full horizontal spreading is achieved at the end of each branch. If Horizontal Spread is 1 and Spread by Age -1.0, spreading reduces towards the ends of branches.

Branch Profile Attributes

These attributes define how the diameter of the branch changes towards it end point. You can define the profile using a constant tapering factor with some randomization, or using a profile curve, or as a combination of both controls, in which case the tapering factor is the product of the two controls.

Note: NURBS curves do not support pointwise attributes in the current program version. Branch diameter control is limited to the start and end points with this geometry type.

Tapering: Defines how much each branch section gets thinner during one age unit (from a branching point to the next branching point). Tapering factor 0.2 means that the end radius is 20 % less than radius at the start.

Rnd Taper: Randomized tapering. A value 0.1 changes the radius randomly -10 % ... +10 % at every age step.

Profile Curve: The diameter profile of the branch (excluding the effect of the tapering factor). With the curve, you can define accurately the shape of a tree trunk, or the branches of a cactus tree, which branches actually thicken after the branching point.



A cactus has an unusual branch thickness curve. Branches are thickest in the middle.

Leaf Density Attributes

U: The leaf density around the branch.

V: The leaf density per age unit along the branch.

Density Curve: The density of leaves by age. This curve controls all leaf types in a deterministic manner; the density is the product of U and V factors and the curve value. Leaf objects have another curve, which controls the density of each individual leaf type by defining a stochastic distribution for it.

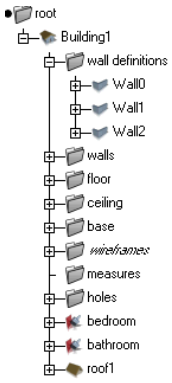


Flowers are more rare than ordinary leaves

Building

The building object constructs a complete building from the parameter lines. It manages roof, wall and room sub objects to perform this job.

The hierarchy of the building object has the following sub levels:



The structure of a building object

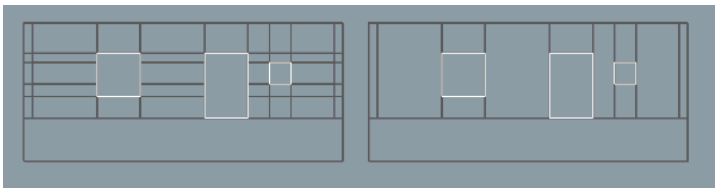
- wall definitions: Contains the wall parameters. The first sub wall object is the outer wall. Other walls split the building into rooms.
- walls: the wall geometry which the building object generates. You can place material attributes into this folder. Editing the wall geometry items while the buidling construction is enabled makes no sense, because next construction refresh will remove all changes.
- floor: if the floor feature is enabled, the floor surface is placed here.
- ceiling: if the ceiling feature is enabled, the ceiling is placed here.
- base: if the base feature is enabled, the base plate is stored here.
- wireframes: contains temporary geometry which is added to achieve a clean wireframe representation.
- measures: the measure lines showing the dimensions of the building are stored here, when the respective option is turned on.
- holes: the hole tool places the hole parameter data (tagged rectangles) to this folder. Note that a hole can be anywhere in the hierarchy of the building (embedded into a window or a door object) and it still works.
- room1, room2: room objects, which decorate room surfaces locally.
- roof: the roof creator object. If present, the building tool automatically extracts the roof edge curve and updates the parameters for the roof respectively.

The building object has the following options available in its property gadget (Property window/Spec tab):

Wall Geometry options:

- Build: enables automatic construction. Generated wall geometry can be edited manually only after Build is turned off. Turning Build on removes manual changes. When the option is enabled, only the parameter data is saved to disk files, which usually makes the file size small.
- Floor: enables automatic floor surface construction.

- Ceiling: enables automatic ceiling surface construction.
- Auto Texture: if enabled, the buidling object automatically adds and updates material mapping objects for walls, ceiling, floor and base.
- Geometry: SDS geometry has a better OpenGL preview, but is somewhat slower to render. It is easier to make Boolean operations with Analytic geometry.
- Height: height of the walls. Note that sub walls have an option 'Extend to Roof', which can override the common base height.
- Uniform Division: each wall section is divided vertically using the same division. This generates a denser SDS mesh, which is usually easier to edit manually.



A wall with 3 holes. Uniform wall geometry division on the left, minimal division on the right.

Base options:

- Shape: either a simple rectangular plate or a polyhedron, which matches the outer wall shape.
- Height: the height of the base. The base extends down from the walls. Changing the height does not lift the roof higher.
- Offset: shrinks the base plate inwards from the outer walls.

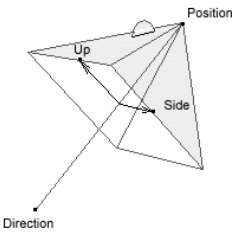
Measure Lines:

- Measure Lines: None, Left, Right or Both. Selects where the measure lines are shown.
- Offset: the distance from the wall polylines to the measure lines.
- Text Size: defines the font size (in pixels) for the displayed dimension values.
- Arrow Size: size of the end arrows (in pixels)

Camera Object

A scene can include any number of camera objects, in which case the 'Current' attribute indicates the current active camera. The Current attribute can be animated so that the current camera may change during the animation.

The geometry of camera object is defined by four points.



The Center specifies the position for the camera. The Direction point defines the aim point for the camera. The focal axis can be derived from the two by subtracting the Position from the Direction point. Also the focusing distance, which is needed for the depth-of-field effect, is defined as the distance between the two points. The two other points, 'Up' and 'Side', define the opening angle for the view pyramid in both directions. The image aspect ratio, for example, can be derived from these geometry points.

The geometry can be skewed, in which case the film surface is no longer perpendicular to the focal axis.

Realsoft 3D supports both perspective cameras and orthographic cameras. This property is controlled via the 'Projection' attribute. Orthographic cameras are often used for modeling purposes, or to simulate distant cameras.

In addition to geometric attributes, a camera object defines many other attributes well known from their real counterparts.

Film Sensitivity acts like a general brightness control.

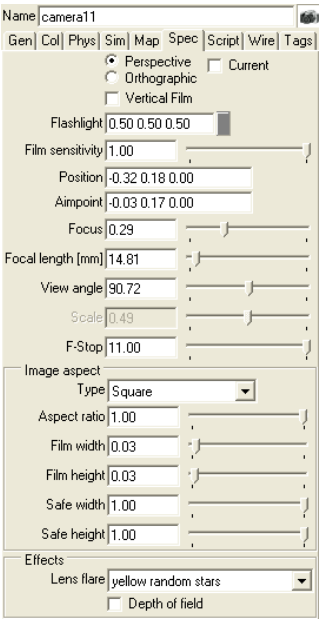
Depth-of-field effect depends on the properties of the camera. The following attributes can be used for controlling it:

- Depth Of Field - on/off
- Focal Length - the focal length of the lens
- Fstop - fstop value
- Focusing Distance - the distance between the aim point and the position point.

In real world, the lens flare effect depends on the properties of the camera. This is also the case with Realsoft 3D. The lens flare effect is automatically generated according to the properties of the light sources and the camera. Camera object defines an attribute 'Lens Flare', which specifies the name of the lens flare post effect for the camera.

The camera object also supports a 'Flash light'. The color of the flashlight can be defined freely. Black color turns the flash light off.

Camera Object Property Gadget



Perspective/Orthographic: The projection type of the camera.

Current: Selects the 'Current' camera, which is used in camera tracking and in file rendering.

Vertical Film: Activates a special film projection which keeps vertical lines parallel.

Flashlight: Defines the color and intensity of the camera flashlight.

Film sensitivity: Controls the overall brightness of images rendered using this camera. The higher the value, the more sensitive the film is, becoming more exposed so that the picture appears lighter.

Position: The location of the camera in 3D space.

Aimpoint: The point camera is directed to. Defines also the sharp distance for Depth of Field.

Focus: The distance from the camera position to the aim point. When using depth of field, the image appears sharp at this distance.

Focal length: Defines the opening angle of the camera.

View angle: The same as above, expressed as an angle value.

Scale: The same as above, expressed as inverse of width of the visible area at 1.0 meter distance. Useful in orthographic projection. The scale value describes how much the camera magnifies the image.

F-Stop: The lens opening.

Image Aspect Frame

Type: A list of predefined camera aspect ratios.

Aspect ratio: The current image aspect ratio (height/width of visible area). Note: this value is independent of pixel resolutions and pixel aspect ratio. Set them separately for example in the File Rendrering window.

Film width and Height: Film size also controls the aspect ratio and affects depth of field.

Safe Width and Height: You can define a safe sub region using these controls. The safe region is part of camera wireframe and is used as a visual guide only (does not affect rendering). When tracking a camera view, you can use an appropriate safe region guide to verify that critical objects fit into the image area that is always visible. For example, borders of TV screens always hide part of the received image signal. (As a general rule, 0.9 is "safe motion", 0.8 is "Title safe".)

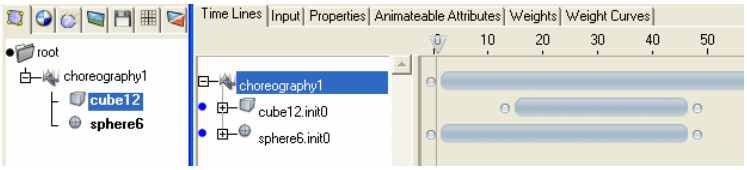
Effects Frame

Lens flare: Activates automatic camera/light source lens flare effect. You can edit the properties of the flare from the post particle effects tab of the select window: activate the same effect there and open the property window.

Depth of Field: Activates camera defined depth of field effect. The DOF properties can be edited from the post image effects tab of the select window. Select the DOF effect that is used in the current post effect configuration and change its properties. Note: this setting is relevant only when the post effect configuration includes a DOF effect with the 'Camera Defined' option enabled.

Choreography

The choreography object is a level object which provides hierarchical time line manipulation. You can place several animated objects inside a choreography level and control the timing of them all trough a single time line.



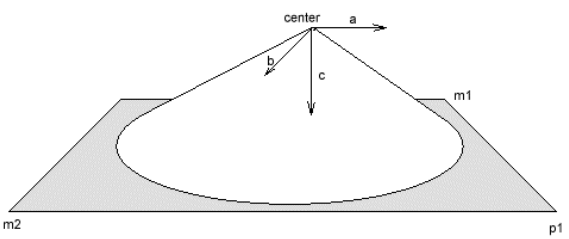
The attributes of the choreography object, available at the Spec tab of the property window - are the same as those which can be manipulated through the choreography editor:

- **Begin** - The start of choreography object's time line.
- **End** - The end of choreography object's time line.
- **Frequency** - Repetition frequency. For example, value 2 will repeat the sub animation events twice during the time line.
- **Phase** - Shifts the phase of the included animations.
- **Weight** - Choreography weight. Controls how strongly the choreography affects the object.
- **Periodic** - Periodic time evaluation for creating cyclic motions.

Cone

An analytical cone can be derived from an ellipse by a third axis 'C'. Three points 'p', 'm' and 'n' are used for defining so called 'clipping plane', which trims the infinite cone surface to a desired finite size.

For example, if the length of all the three axes is the same, a circular cone with 45 opening angle is achieved.



The cone supports efficient volume enclosure test, which makes it especially useful object in Boolean operations.

Constraint

The constraint object serves scripting purposes. Normal objects can access their own attributes at the Scripting tab. The constraint object can access attributes of all objects that were selected at the constraint creation time. For example:

- Create three spheres and create a simple animation where all the spheres move here and there.
- Select the spheres
- Create a constraint (pull down menu Tools/Construction/Create Constraint)
- In the Spec tab of the property window, set Command Language to JavaScript and enter the script below.

```
if(P.y < 0)
    P.y = 0;
```

The 'P' variable refers to the current animation channel, Translate attribute in this case.

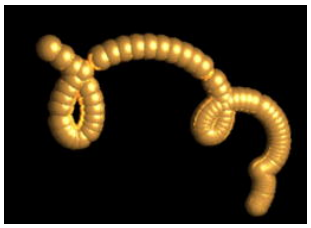
The constraint object defines the following variables:

- Target - target object. You can read any target attributes simply by calling appropriate Get method. However, you should modify target objects through the 'P' variable. This results more reusable code.
- Self - constructor object itself.
- Time - current time
- Type - the type of the target attribute (as defined in r3typids header file)
- Tag - attribute identifier, as defined in the class header file
- Index - attribute inded, or -1 for regular attributes

Creator

The Creator object is a level object that generates duplicate copies of its sub-objects during an animation. The level that contains the Creator must have the Simulation property turned on. There are many uses for creator objects: bullets being shot from a gun, bees leaving a hive, water dripping from a faucet, just to offer a few examples. Creator objects can be animated using keyframes or paths, just like many other object types.

The **Active** checkbox turns creation on and off. You can animate it to achieve object creation at suitable intervals.



A creator objects moves along a path and shoots spheres around

The **Parent Level** attribute specifies how many hierarchy levels above the creator the emitted objects are inserted. The default is 0, which causes objects to be inserted at the same hierarchy level that contains the Creator object.

The **Creation Interval** attribute defines the time interval between the creation of emitted objects; that is, it defines how rapidly the Creator emits objects. The smaller the value, the more objects are created. The time interval is specified in the current time units (defined through the Options window). So, for instance, if your time units are frames, setting the creation speed to 15 will cause a sphere to be emitted every 15 frames, or two times per second for an

NTSC video system. If the **Channel** gadget below the creation interval gadget includes a channel selection, materials that affect the creator are evaluated at each sample object position. The evaluated channel value is multiplied with the creation interval value. The result is used as the actual sample specific creation interval.

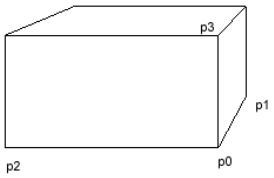
The **Life Time** attribute defines the time interval that an emitted object is active before being destroyed (i.e. its lifespan). The smaller the value, the shorter the object's life. The time interval is specified in the current time units (defined through the Options window). So, for instance, if your time units are frames, setting the Life Time to 15 means a sphere is active for 15 frames (0.5 seconds in NTSC video) before being destroyed. If the **Channel** gadget below the life time interval gadget includes a channel selection, materials that affect the creator are evaluated at each sample object position. The evaluated channel value is multiplied with the life time value. The result is used as the actual sample specific life time.

The **Velocity** attribute defines the rate at which the created objects move, in meters per second. Since this attribute is a vector value, you define velocities along the three axes. To get motion only along the X axis, you would enter a value such as "0.5 0 0". Velocity is defined in the Creator object's space. The **Channel** gadget defines a material channel whose contents is multiplied with the given velocity value to obtain the sample object specific velocity.

The **Spin** attribute defines the rate at which an emitted object spins. Since this attribute is a vector value, you define the spin along the three axes. If the associated **Channel** gadget includes a channel selection, materials that affect the creator are evaluated at each sample object position. The evaluated channel value is multiplied with the spin value. The result is used as the actual sample specific spin.

Cube

An analytical cube is defined by four corner points, defining three edges (or three faces) for the cube. Other faces are computed internally and they are always coplanar to their opposite counterparts.



The rounding attribute of cube defines an efficient render time edge rounding. For the sake of speed and memory use, the rounding is shown only in ray tracing.

Curve Manager

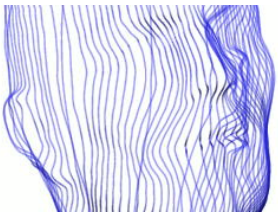
The curve manager object creates a set of NURBS curves by evaluating its parameter objects.

- **Displacement** - Moves the curves away from the parameter surfaces by displacing the points with the defined distance to surface normal direction.



Note

Also negative values are allowed.

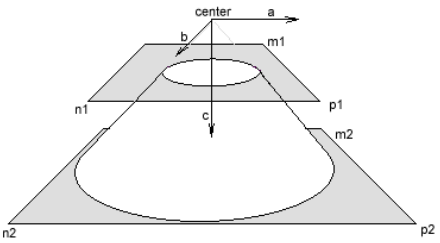


- **Channel** - If a channel is selected here, materials affecting the parameter objects are evaluated. The value of the selected channel is used to displace curve points.
- **U Curves** - If set, curves in the U parameter direction are evaluated.
- **V Curves** - If set, curves in the V parameter direction are evaluated.
- **Curves** - The density at which curves are evaluated.
- **Points** - The density of points in created curves.

Cutcone

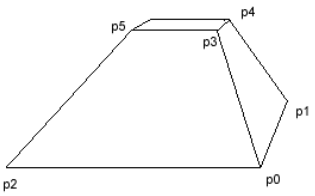
A cut cone object can be derived from a cone by introducing a second clipping plane.

Also the cut cone supports a quick inside / outside test, which makes it efficient in Boolean operations.



Cut Pyramid

A cut pyramid is defined by two rectangles: the top and the base for the pyramid. All faces of the pyramid must be planar 2d surfaces and the top and bottom faces must be coplanar.

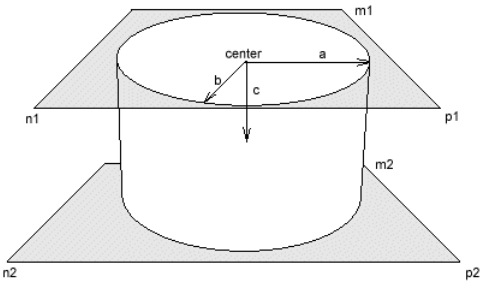


Cylinder

An analytical cylinder can be derived from an ellipse by introducing a third axis attribute 'C', the axis of the cylinder.

To get the infinite cylinder surface trimmed to a finite size, the cylinder object also defines two so called 'clipping planes'. These clipping planes define Boolean operations which clip out all points which don't reside between them. A clipping plane is defined by three attributes: one corner point (p) and two vectors specifying the associated edges for the plane (named as 'n' and 'm').

A cylinder performs an efficient volume test, which makes it useful object in Boolean operations.



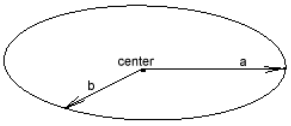
Distant Gravity

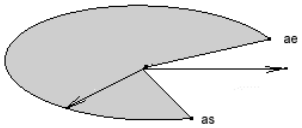
The distant gravity object generates a constant gravity field that does not depend on the distance. It is useful when simulating real world gravity effects i.e. when the scale of events is so small that the gravity can be considered constant.

A distant gravity object includes two points, which define the direction of the gravity field. The length of the gravity line is irrelevant: the **Strength** of the gravity is controlled using the value at the property window's Spec tab. The default value 9.82 m/s^2 is approximately the gravity on the earth surface.

Ellipse

An analytical ellipse is defined by a center point and two axes. The axes don't have to be perpendicular to each other. If the lengths of both axes are equal and the axes are perpendicular to each other, a special case called a 'circle' is achieved, in which case the lengths of the axes are the radius of the circle.

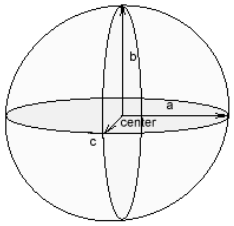




Ellipse, as well as nearly all other geometric shapes derived from it define an attribute called 'Sector'. When set, two additional attributes 'Start Angle' and 'End Angle' can be defined to achieve sector like shapes. Internally, all angle attributes are represented as radians, but the user interface allows use of other standard units as well.

Ellipsoid

An analytical ellipsoid is derived from an ellipse by introducing a third axis 'C'.



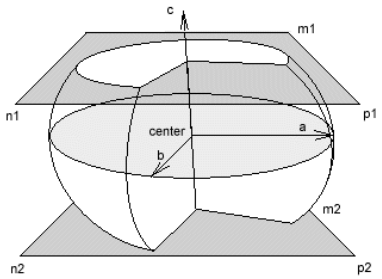
When all the three axes have equal lengths and they are perpendicular to each other, the shape is a sphere.

The sphere supports an efficient inside / outside test, which makes it especially useful object in Boolean operations. It is also very quick to render in ray tracing.

An ellipsoid doesn't support Sector attribute. If this option is needed, use an Ellipsoid Segment object instead.

Ellipsoid Segment

An ellipsoid segment differs from the ellipsoid object in that it supports two clipping planes as well as the sector attribute.



Extrude

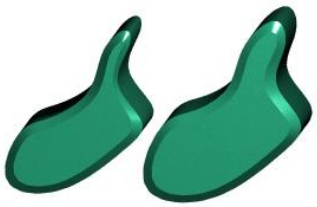
The extrude manager object builds an extruded NURBS mesh system from the parameter NURBS curves. The following options are available:

- **Radius** - The bevelling radius. Measured in millimeters.
- **Depth** - Depth of extrusion in current measuring unit.
- **Bevelling** - The bevelling type. The following alternatives are available:
  - *None* - Extrusion creates sharp 90 degree edges.
  - *Sharp Edge* - Edges are bevelled with a straight line cut at 45 degree angle.
  - *Rounding* - Edges are rounded smoothly along a circle arc.
  - *Groove* - Edges have groove like round bevelling.



None, Sharp, Round and Groove

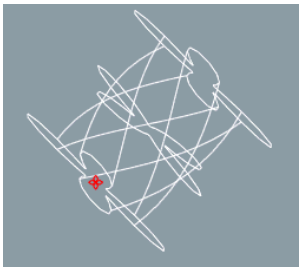
- **Bevel Out** - The beveling is created by expanding the parameter curves. This creates fatter objects.
- **Independent** - If set, extruded surfaces from each parameter curve are collected into a separate hierarchy level. If cleared, all surfaces are placed to a same level.



Inwards bevelling on the left, Bevel Out applied on the right

Fan

The fan object is used in fluid simulations. It generates a fluid environment and produces a flow with the following controls:



Turbulence: Amount of turbulence in the flow. If turbulence is zero, the flow has the same constant direction everywhere.

Radius: The radius of the fan object. Radial falloff below uses the radius.

Radial falloff: This setting defines how the flow reduces away from the fan axis.

- *Constant* - The flow does not depend on the distance from the axis.
- *Linear* - The flow decreases to zero at the outer fan edge at a constant speed.
- *Quadric* - The flow speed is defined by a 2nd order parabola: speed = k*d*d, where d is distance from the outer edge towards the axis. This kind of flow appears in real world water pipes.

Axis falloff: Defines how rapidly the flow decreases in the direction of the fan axis. The falloff is symmetric in front of and behind the fan.

- *Constant* - The flow does not depend on the distance measured along the fan axis.
- *Linear* - The flow is inversely proportional to the distance: flow = maxflow/(c+k*distance)



Note

The actual flow speed and other fluid properties can be defined from the Physics tab of the property window.

Field Evaluator

The field evaluator object is capable of evaluating useful material and physical properties at its location in 3D space. Evaluation follows general hierarchy rules of material mapping i.e. only materials mapped to the evaluator object or its parent levels are considered. Evaluated properties can be used as parameter information for intelligent animations. Usually the field evaluator object itself is not the target of animation or material mapping, but it is placed into the hierarchy of other objects to examine their properties.

The **Value** channel shows the result of evaluation of the material channel defined in the **Channel** selector below it. All surface property channels and user defined channels are available for evaluation.

Velocity, **Spin**, **Acceleration** and **Angular Acceleration** show the respective physical properties of the Field Evaluator object during animation playback. In addition, you can use them to enter key values when key framing objects which have been parametrized with the field evaluator attributes.

File Instance

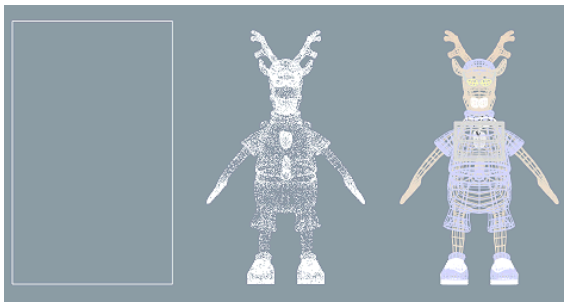
The File Instance object instances specified Realsoft 3D project file (.r3d). A file instance object can be transformed or animated similar to any regular geometric object. However, it cannot be single point edited or morphed.

File instancing often saves memory and speeds up real time rendering when building complex scenes. Another important application is related to project management. Several users can edit and improve components of the same scene. The results are automatically combined via file instancing.

The attributes of the file instance object are as follows:

- **File** - The name of the .r3d file to be instanced
- **Geometrics** - Load in geometric objects from the file
- **Load Geometry** - Load in geometry for accurate real time rendering
- **Materials** - Load in material objects
- **Effects** - Load in particle post effects
- **Render as instance** - Activates instancing in photorealistic rendering.

If the Load Geometry option is set, the full geometry is loaded and kept in memory for real time rendering purposes. In this case, a file instance will not offer any memory saving compared to using a regular object. If Load Geometry is disabled, only a simpler visualization is available. This often saves memory and speeds up real time rendering.



Three ways to visualizefile instances: bouding box, point cloud and full wireframe

The alternatives for simpler visualization are:

- If Wire Density is 1, the object is drawn as a simple wireframe bounding box.
- If Wire Density is greater than 1, the object is drawn as a point cloud. The Wire tab's Curve Quality and Wire Density controls together define the density of the point cloud.



Note

File Instancing resolves material and particle effect references primarily inside the instanced file. For example, if the user has created a wooden chair and adds a file instance of it into a new project, the wood material is first searched from the instanced file. If it is not found, then the wood is searched from the actual material library of the project.

If the Render as instance option is set, the render engine does not convert all file instances of the same target file to real objects but performs the instancing transformation during ray tracing. Only the first file instance is stored in memory; other items will transform the same geometry data. This typically slows down rendering by 20%-30%, but saves memory. For example, it is possible to model and render a forest by repeating thousands of instances of a dozen of basic tree shapes.

Applying render time instancing to simple basic shapes does not save memory, but actually can increase the memory use, because the transformation matrices require also some memory space.



Note

Render time instancing has some limitations. Currently render time instancing does not allow nested or scaled instances. Only translation and rotation are allowed. The program automatically disables render time instancing if these conditions are not set. Secondly, the applied geometric transformation may confuse complex VSL shaders which refer to 3D coordinate information.

Font

The font object creates a set of NURBS curves from a defined text. The font curves can be extruded to 3D object using other NURBS tools, such as the Extrude tool.

The font object has the following options:

- **Text** - A text string from which the curves are generated.
- **Name** - Selects the font typeface.

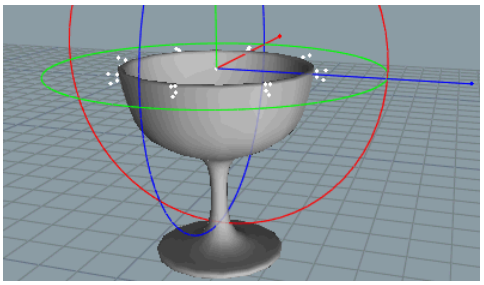


Group

A group object stores a set of links to geometry points. You can store a complex point selection to a group. Applying modifications later to the defined set of points becomes easier: you can modify the group instead of modifying the points directly.

You can also use the group for selecting and deselecting the points afterwards; the control bar shows the required tools as soon as you select a group object.

A group can refer to points of a single object or to multiple objects.

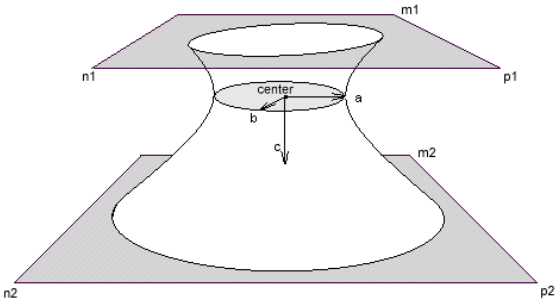


A group object for controlling the upper edge of a wine glass

Hyperboloid

A hyperboloid is defined by a center point, three axes and two clipping planes.

The object supports Sector attribute and an efficient inside / outside test.



Input Plane

The input plane object is a plane in 3D space. It stores the orientation of a modeling plane (=view's input plane) and the grid origin. You can place input plane objects to those areas of the scene where modeling actions are happening frequently. You can then set the modeling orientation and grid origin quickly by drag and dropping a suitable input

plane object from the select window into a view window. You may also apply the auto focus tool to zoom into the region defined by the input plane square.

For example, when modeling a room interior, it may be a good idea to put an input plane object on the ceiling, on the walls and on the floor.

The input plane property sheet contains controls for plane orientation (HPB angles) and the position of the grid origin included in the plane. The **Current** property is used whenever a view window is set to Lock to Input Plane state. When the current input plane is modified, the view updates automatically.

Instance

The Instance object duplicates geometry of another object. Instance stores only the changes made to the original object. Therefore, it is a memory efficient way to create multiple copies of a same geometry.

Instance property gadget has the following options:

Ignore matrix - When set, the original object can be transformed (move, rotate, scale) after instance creation without changing the instances. If the option is cleared, instaces take into account all object transformations of the original object and its parent levels. Which way is better, depends on the case. Even when the option is set, it is still possible to modify all instances simply by performing the desired transformation to sub objects of the original object or to its actual points.

Render as instance - activates instancing in photorealistic rendering. The render engine does not convert instances to real objects but performs the instancing transformation during ray tracing. This typically slows down rendering by 20%-30%, but saves memory. For example, it is possible to model and render a forest by repeating thousands of instances of a dozen of basic tree shapes.

Applying render time instancing to simple basic shapes does not save memory, but actually can increase the memory use, because the transformation matrices require also some memory space.



Note

Render time instancing has some limitations. Currently render time instancing does not allow nested or scaled instances. Only translation and rotation are allowed. The program automatically disables render time instancing if these conditions are not set. Secondly, the applied geometric transformation may confuse complex VSL shaders which refer to 3D coordinate information.

Interpolator



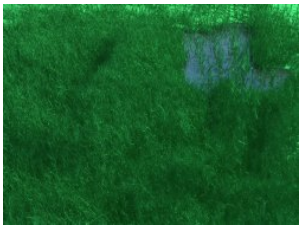
The Interpolator object is a level object which produces interpolated copies from its sub objects at photo-realistic rendering. For example, you can model 1000 sample hairs and render them as 100 000 hairs.

It is much faster and easier to control the small sample object set as the real amount of hairs. Also, project file size does not grow too much. For many simple geometries, modeling requires much more RAM memory as the rendering. Therefore, the interpolator makes it possible to render much more complex objects.

An interpolation hierarchy example:

```
interpolalator
    nurbs curve 1
    nurbs curve 2
    ...
```

Interpolation happens between direct sub objects of the interpolator level. Therefore, an interpolator must have at least two sub objects to function. Interpolated sub objects should have identical geometrical structure. In the example above, each nurbs curve must have the same number of points.



Vegetation modeled using the interpolator object



Note

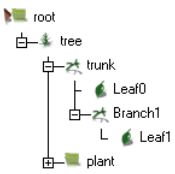
The object interpolates only the geometric attributes of its children.

The interpolator controls are:

- **Count** - Number of additional render duplicates between each two consecutive sub objects.
- **Bilinear** - Selects bilinear interpolation over consecutive object quadruples. Linear interpolation places duplicates along a straight line between each object pair. Bilinear interpolation places the copies into a rectangular area defined by 4 objects. Therefore, bilinear interpolation method usually fills surfaces better than linear interpolation.
- **Closed** - Selects closed interpolation. Interpolation continues from the last sub object back to the first sub object. The option for is useful for creating circular object groups.
- **Min and Max Distance** - Interpolation happens only when the objects are within the defined range. It does not make sense to interpolate hair between dog's tail and its ear, because the hairs would float in the air.
- **Random** - Additional random variation for interpolated copies. The value defines a maximal random displacement in the current measuring unit. For example, a random value 0.1 displaces points so that each point moves to a random position that is not further than 10 cm away from the original location.

Leaf Object

Leaf object generates leaves and needles along the surface of its parent branch object. To create a leaf, select a branch object or some of them and hit the Leaf tool button in the tool control bar, or select the pull down menu Methods/Branch/Leaf.



One leaf object adds leaves for the trunk and another one for the branches

Note that leaves are generated only for the direct parent object - parents of parents are not taken into account.

A branch object can have multiple leaf children. All leaf objects are considered active every time the parent branch object determines to insert a leaf. For example, a branch of a rose can have three sub leaf objects: a 'spike', a 'flower' and a 'leaf'.

Leaf object has the following attributes:

Type: Defines the type of leaf geometry. The choices are:

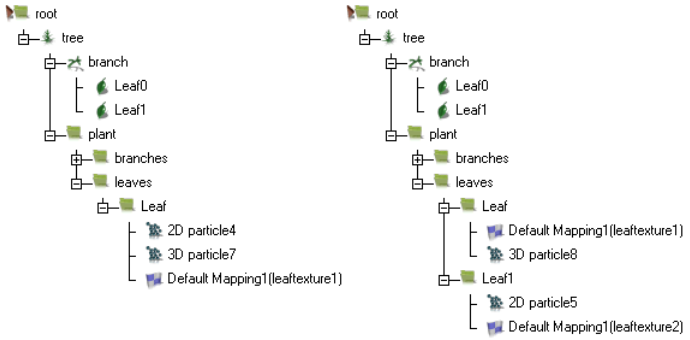
- **Copy** - leaves are generated by copying the sub objects of the leaf. If a leaf has several sub objects, the copy is selected randomly from the sub object list.
- **2D Particles** - Leaves are rendered as ray traced 2D particles. 2D particles are always symmetric by the axis, and therefore this option is mainly suitable for representing needles and spikes. The Size attribute of a 2D particle deaf defines particle's length, its radius at the start and the radius at the end.
- **3D Particles** - Ray traced 3D particle leaves. Each leaf is an elliptic sphere. Thin long particles look like needles, wider flat particles look like leaves.
- **Particle Quads** - This option renders each leaf as a planar UV mapped quadrangle By applying a suitable color texture and a clip map, natural looking leaves can be achieved.
- **None** - Turns leaf creation off (for temporary trunk/branch editing actions etc.).



Leaf types from left: Copies of Nurbs modeled curved surfaces, 2D particles, 3D particles and clip+texture mapped flat quads

Own Hierarchy: If set, the actual leaves generated by this leaf object are placed into a separate folder. This can be necessary when the plant object includes multiple leaf objects and you want to attach a unique material map to each type of generated leaves.

Size: Size of leaves. The three values represent length, width and thickness. Thickness is irrelevant for flat quadrangles. For 2D particles, the values define length, start and end radius.



The plant on the left has all its leaves in a same folder. On the right, the Own Hierarchy option places the leaves into separate folders so that they can be textured individually.

Random Size: Randomizes the size definition components. The random factor is multiplicative: a value triplet (1, 1, 1) will give each particle a random size between zero and the defined full size. A value triplet (0.5, 0.1, 0) varies length by 50 % randomly, width by 10 %, and thickness will get no random variation at all.

SeparateRnd: If set, an independent random value scales each leaf dimension. An example: an orange tree has 3D particle leaves as fruits. The maximum leaf size (radius of oranges) can be 0.05 0.05 0.05. Because some oranges are big, some small, the Rnd Size can be (0.5 0.5 0.5) for 50 % size variation. Separate Rnd must not be set, because it would make some oranges long and thin, some flat.

Leaf Angle: The angle between the leaf and the branch from which it is growing. Note: the angle around the branch depends on several other attributes of the parent branch and the leaf itself, such as twist and horizontal spread.

Random Angle: Randomizes the angle between the leaf and its parent branch. Randomization is additive: a random value 1.0 adds a random angle variation between -90 ... +90 degrees to the regular leaf angle.

Horizontal Spread: Guides the leaves to spread out horizontally around the branch. Horizon direction taken from the blue z axis of the object space of the plant root level. Many plants grow this way: the purpose is to catch as much light as possible. Note that will full horizontal spreading level 1.0, leaves may start overlapping unless there is a sufficient amount of position randomization.



Horizontal spreading increases towards the end of the branch

Spread by Age: Changes the horizontal spread by the age when leaf started to grow. The actual spread parameter is the sum of the constant spread and the age dependent spread. If Horiz. Spread=0, Spread by Age=1.0, full horizontal spreading is achieved at the end of each branch. If Horizontal Spread is 1 and Spread by Age -1.0, spreading reduces towards the ends of branches.

Random Position: Randomizes the position of leaves. If the value is zero, leaves will grow in regular rows and columns along the branches. Full random level 1 places the leaves to totally random positions.



Regularly and randomly positioned needles

Shift: Applies regular incremental shift to leaves. Shift is incremented by U direction (= around the parent branch), and moves each leaf along the branch. For example, if U density is 2 and Shift = 1.0, leaves switch the side of the branch regularly. In other words, Shift parameter produces spiral like distribution of leaves around the branches.



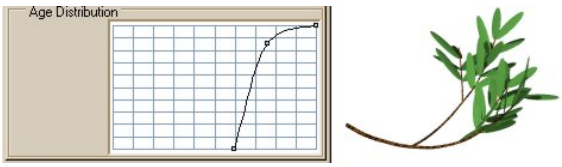
Both plants have Leaf Shift set to 1.0. Parent branch's leaf density in U direction is 2 on the left, 20 on the right. High U Density reveals the spiral form distribution.

Gravity: Guides the leaf orientation to take gravity into account. The direction of gravity is defined by the plant's root object's z axis (blue axis line). The higher the gravity, the more leaves will hang down. A negative gravity value makes leaves to grow upwards (to simulate light attraction etc.).



3D particle leaves are spread sideways using a high horizontal spread value, and then pulled down using some gravity

Age Distribution: The distribution of this particular leaf type in the plant by age. A rapidly ascending curve that starts from x=0.9, y=0 and ends at x=1, y=1 will guide the leaf to grow only at the end of branches. For example, plant flowers can be made to grow only at branch ends this way.



Distribution for leaves, which grow only at branch ends

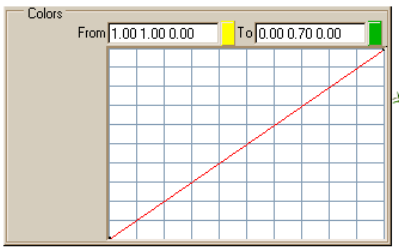
The curve defines a stochastic distribution for the event that the leaf grows at a certain age. For example, if the curve travels at the 50% probability level (y=0.5), about one half of the attempts to grow a leaf of this leaf type will succeed.



Note

Each branch object has top-level controls Leaf Density U, Leaf Density V and Leaf Density Curve. These properties control all sub leaves the same way. The Age Distribution curve gives a more detailed, leaf specific control.

Colors: The From and To fields define a color variation range for leaves. Each leave gets a randomly picked color from the defined range. The age distribution curve below these two fields define another color which depends on leaf's position in the plant (0 at the root, 1 at the end of branches). The final color is the product of the random color and the age dependent color.





Leaf colors variate from yellow to green. Old leaves are darker than young leaves.

Note: Leaf colors are evaluated and stored only when the Set Colors option of the plant's seed object is enabled. If colorization is done using materials and texture maps, which replace the colors computed using the options above, it is wise to turn the Set Colors option off to save some memory.

Level Object

A level object corresponds to 'directories' or 'folders' in Disk Operating systems.

Level objects allow all geometric objects in Realsoft 3D to be grouped into a hierarchical tree structure. For example, a table consists of two sub objects: a base and a cover. The base consists of any number of sub objects, such as legs and so on.

All geometric objects, including levels, have a 4x4 matrix associated with them. This matrix defines so called local object space for the object. All geometric attributes for the object are defined in this local space.

Linear transformations are applied to an object by combining a 4x4 transformation matrix to the object space matrix of the object. This transforms the geometry of the object, as well as the geometry of the sub objects. The 4x4 matrix is usually represented to the user as four vector fields: Translate, Scale, Rotate and Skew, but it can also be manipulated directly through scripts or the SDK.

The level object also plugs in a feature called 'Boolean Operations'. If the type of the level is 'Union' (OR) , then all the matter defined by the sub objects of the level is added into the universe. In other words, the logical operation between the sub object is OR.

If the type of a level is 'Intersection', then the sub objects are Boolean operated with each other using the AND logical operator.

Levels also allow geometric objects and material description objects to be grouped into a single entity. For example, one can create a wooden sphere by creating a level consisting of two sub objects: a sphere and a 'wood' mapping object. The level object property gadget includes the following controls:

- *Type of the level* - Defines how sub objects of the level are combined. The following alternatives are available:
  - *Boolean union (OR)* - The sub objects add matter to the scene i.e. the volume of the level is the sum of the sub volumes.
  - *Boolean Intersection (AND) - The volume of the level is the intersection of the sub volumes.*
  - *Boundary surface solid* - The surfaces of the sub objects are combined to define a volume in space. With this option, you can build a solid object from surfaces that do not define a natural volume themselves (a set of NURBS patches, SDS objects etc.). The surfaces should form a closed system without holes, otherwise rendering errors may appear.
  - *Material mapping average* - This option is useful for combining several materials into a weighted average. Most materials replace the result of the previous material: adding marble over a wooden sphere makes it look marble. With the mapping average level, you can make a weighted mixture of the materials, say 70 % marble, 30 % wood. The weight of each material can be defined using the 'Scope' value of the corresponding mapping object, or by assigning a scope value in each material.
- *Simulation* - Enables physical simulation computations for child objects in animations.

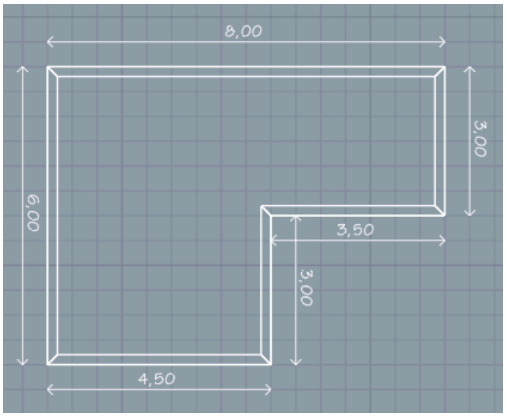
Link

A link object serves only object evaluation purposes. You can use link pointing to a geometry object in systems which do not set special requirements for parameter objects. For example, you can use a link to define an animation path.

Measure

The measure object is displays its length in view windows. You can add measure objects to you scenes to show information about object dimensions.

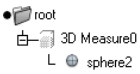
The properties of the measure object are:

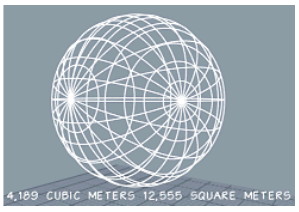


- Photorealistic rendering method defines how the geometry is rendered. Note that the object must be made rendering-visible using the Gen tab of the property window.
  - **Ray Trace** - The measure line and its dimension text is rendered using ray tracing
  - **Scanline** - The measure is rendered using scanline method
  - **Post Process** - Rendering is done at post processing stage
  - **Pseudo 3D** - A special correction is applied to the geometry, to obtain a constant appearance the same way as in real-time drawing. Thickness and size of the object will appear constant, regardless of the distance from camera or the viewing angle.
- **Text Size** - The size of digits, which display the length. The value is given as per mills of the full view width in Pseudo 3D mode. If Pseudo 3D mode is disabled, the unit is millimeter.
- **Thickness** - The radius of the line used in photorealistic rendering(per mills)
- **Arrow Size** - The size of the arrow end symbols (per mills)
- **Threshold** - Angle threshold for measure drawing. Measure lines, which are seen in a very low angle (viewing direction almost parallel to the measure line) should not be drawn, because the projected line becomes a single dot. This threshold controls when clipping occurs. Value 1 draws all lines regardless of the angle.
- **Length** - The current length of the measure

3D Measure

The 3D measure object computes and displays the volume and/or area defined by its sub objects. For the volume computation, sub object must be proper solids (analytic objects or properly closed boundary surface solids). Sub objects can include arbitrary boolean operations.





The properties of the measure object are:

- Photorealistic rendering method defines how the displayed geometry is rendered.
  - **Ray Trace** - The volume/area text is rendered using ray tracing
  - **Scanline** - The text is rendered using scanline method
  - **Post Process** - Rendering is done at post processing stage
  - **Pseudo 3D** - A special correction is applied to the geometry, to obtain a constant appearance the same way as in real-time drawing.

- **Text Size** - The size of numerical digits (per mills of the full image width), which display the volume and area
- **Thickness** - The radius of the line used in photorealistic rendering (per mills)
- **Accuracy** - Accuracy of the computation. Minimal value 0 computes the volume and area of the bounding box. The higher the number, the more accurately the result will be computed. For example, the default accuracy 100 computes first 4 digits of the volume of a sphere correctly (error is less than 0.001 cubic meters when radius is 1 m).
- **Volume** - The current volume (a read-only property)
- **Measure Volume** - Enables volume measuring
- **Area** - The current surface area (a read-only property)
- **Measure Area** - Enables area measuring
- **Fused** - If set, overlapping volumes do not cumulate - they are added to the total sum only once. Similarly, surfaces traveling inside solid volumes do not contribute into the 'Fused' area. If the option is cleared, the result values are computed from each solid sub object separately and summed up.
- **Show On View** - Displays the computed volume on view windows

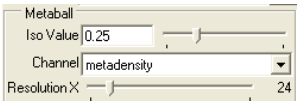
Metaball

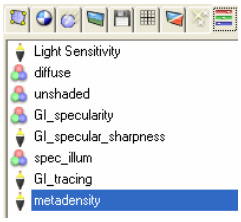
The metaball level object creates a boundary surface for a density distribution defined by sub objects. The sub objects can be symmetric or elliptic spheres or 1D particles and a VSL material can be applied to modify the density in space. By default, the density decreases from 1.0 at half radius distance from the sphere center to zero at the sphere edge. The boundary surface of the metaball system follows the area where density is exactly a given Iso Value. Because the density distributions of metaballs melt together, the result surface behaves like liquid.

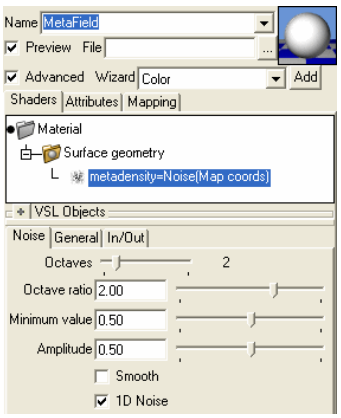
The following controls are available:

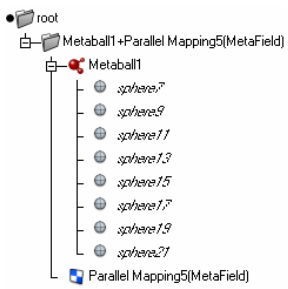
Iso Value: The density value which defines the surface. Decreasing the value makes the result object thinner.

Channel: An optional channel that modifies the density distribution. If set, the materials (more exactly, the Surface Geometry shaders) that affect the metaball system are evaluated. The total density field is the product of the spherical density fields and the material defined channel values.

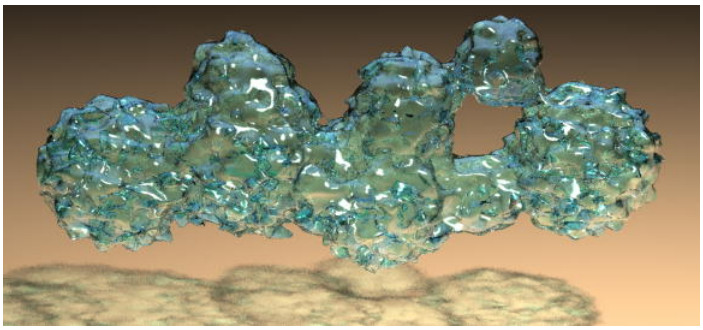








Create a material that assigns a value to the density channel and map it to the metaball system.



Bumpy metaballs created by the system above. Because of the material defined density, only half a dozen metaball objects generate a detail rich surface

Resolution X, Y, Z: The accuracy of the 3D grid which is used for searching the iso surface. The resolution defines the density (and quality) of the generated surface. For example, if the diameter of a metaball system is 20 cm and the resolution values are 40, the surface will show details of the magnitude 20 cm/40 = 0.5 cm.

Expansion: Expands (or shrinks) the size of metaballs from their true geometric size. For instance, an expanded size allows simulated meta objects to melt together, while collision detection prevents excessive penetration.

Test Interval: A speed optimization setting. Boundary surface is searched using the given test interval stepping along the Resolution X, Y & Z grid, and the exact surface is computed only if the density changes across the iso value limit. If the density distribution does not contain small details, the test interval can be several resolution units.

Accuracy: Defines how accurately the surface points are computed to match the exact iso value surface.

Field: Defines how the density field increases from zero at the metaball outer edges towards the full value 1. The higher the degree, the smoother metaballs melt together.

- *Linear* - Density increases at a constant rate along the sphere radius.
- *Quadric* - Density grows proportionally to the square of the radius.
- *Cubic* - Density increases proportionally to the third power of the radius (d = c * r*r*r).
- *Volume* - Density jumps to the full level at the metaball edge.
- *4th Degree:* - The density decreases proportionally to the 4th power of the radius. The smoothest way to blend metaballs.

Rendering:

- Triset - the surface is rendered polygonally using the Marching Cubes Algorithm. This is usually the fastest way, but the result shape often suffers from various rendering errors which are typical to polygon models.
- Procedural - the surface is resolved during ray trace rendering without polygonization. This method renders a perfectly smooth surface, when the Accuracy setting is high enough. It is also extremely memory efficient.

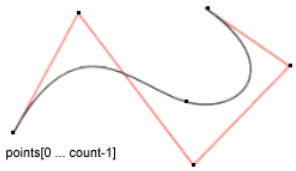
Hierarchical: If set, the full sub hierarchy of the metaball level is examined to find metaballs. Otherwise only direct children of the metaball level are considered. Full hierarchy scan is not always desirable, because the hierarchy can contain animation parameters (morphing keys, creator object's samples etc.).

NURBS Curve

NURBS curves and surfaces are defined by number of four dimensional control points where the first three components define position in 3D space and the fourth component acts as a weight parameter. Increasing the weight pulls the curve towards the control polygon. Thanks to this property, NURBS curves can be used for representing quadric surfaces accurately.

Non-uniform parametrization allows representation of standard curves and surfaces. For example, one can use NURBS to represent standard Bezier and B-spline surfaces.

Realsoft 3D uses a very standard NURBS representation. Only the 'Closed' attribute needs some special attention for example when exporting Realsoft 3D NURBS objects to other applications. Many applications support only open NURBS curves and surfaces.



A NURBS curve is defined by the following attributes:

- Control Points
- Order: Order is degree + 1. For example, the order of a quadric curve is three. Order defines the minimum number of control points.
- Closed If true, the curve is closed. Realsoft 3D makes the curve periodic internally by extending the knot sequence and using a modulo operator.
- Knots: Knots define a set of parameter values in increasing order.

NURBS Curve Property Gadget

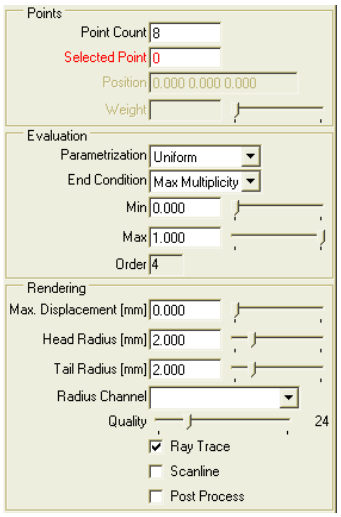
Point Count: The amount of control points defining the curve. Reducing the point count usually changes the shape.

Selected point: The index (0, 1, 2 ...) of the currently selected point.

Position: The location of the selected point in 3D space.

Weight: The weight of the selected control point(s). Increasing the weight pulls the curve towards the selected control points.

Parametrization: Curve parametrization type. Parametrization affects curve shape significantly, and is relevant also in path animations. Uniform parametrization means that all point intervals represent an equal parameter interval. The chord length parametrization makes the parameter division using point distances.



End condition: Max Multiplicity end condition pulls an open curve to the first and the last control points. Phantom end condition uses a similar parametrization for the end points as for the interior points.

Min: The start parameter value for curve evaluation. With this and the next 'max' value, you can limit a suitable sub section from the curve.

Max: The end parameter value for curve evaluation.

Order: The order of the curve. The higher the order, the smoother the curve. Order = 2 produces a polyline. The most usual smooth NURBS curve type has order 4.

The Rendering frame includes settings which control NURBS curve rendering in the photorealistic render engine. The curve can be made visible in the final rendering using the Gen tab's visibility options.

Max. Displacement: The maximal amount of curve displacement defined either by moving the curve points or by changing the radius in a 'Scanline' shader.

Head radius [mm]: Thickness of the curve start (in millimeters).

Tail radius [mm]: Thickness of the curve end (in millimeters).

Radius Channel: An optional channel defining the curve diameter. If set, replaces the start and end radius values. The thickness values can be defined by creating a VSL material, where a scanline type shader assigns the values to the selected channel, and mapping the material to the curves. For example:

```
Material_for_Curves
    Scanline
        my_diameter_channel = Noise(map coords)
```

Quality: Rendering quality for curves. A smooth NURBS curve is drawn using short segments in the scanline or ray trace rendering. This value defines how densely the curve is subdivided to intervals. The higher the value, the smoother the result appears. The value is used per control point interval, so the same quality value works as well with a 5 point curve as with a 100 point curve.



Note

The NURBS curve supports also curve displacement by materials the same way as the thickness control presented above:

```
Material_for_Curves
    Scanline
        coordinates += Noise(map coords)
```

Ray Trace: If set, the curve is rendered using a ray traced geometry which casts shadows, is visible in reflections etc. The geometry type used in this context is quite memory efficient, and tens of thousands of curves can be included in the scene. The curve is procedurally refined towards a smooth shape during the rendering. The Quality value defines the computation accuracy. The minimal quality is usually sufficient for simple curves. High quality values may be necessary if a material is applied to deform the curve shape densely.



Hair created using ray traced NURBS curves

A ray traced curve defines the following geometry information that is available for VSL shaders:

- *UV Coords:* The first U component runs along the curve - head=0, tail=1. In all shaders except the Scanline V runs cylindrically around the curve, and the third component is the distance from the curve center. Scanline shader evaluates only the curve axis points, hence V & W are irrelevant in that context.



A spiral pattern mapped around curves using the native UV coordinates

- *U Direction, V Direction, Normal:* U direction is the tangential head-tail direction of the curve in the evaluated point. V direction is a direction orthogonal to U direction. In a Scanline shader, it defines the direction where the V component of UV coordinates is zero, and Normal channel is the cross product of the U and V directions. These directions are useful when displacing the curve position orthogonal from the curve tangent. In other shaders, where a real surface point is already examined, Normal is the usual surface normal and V direction is the cross product of U direction and Normal.

Scanline: If set, the curve is rendered using scanline geometry. This rendering method is often quicker than the ray trace approach, but scanline curves are invisible in reflections and do not cast ray traced shadows. The curve shape controls are the same as in ray tracing.

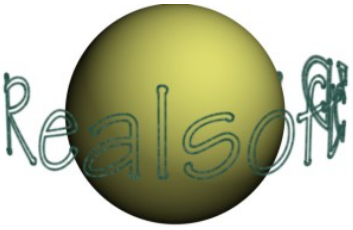
Post Process: Draws the curve in post processing phase, which is the last step in the rendering pipeline. Post processed curves do not react with lights or cast shadows, and do not show up in reflections. Due to the simplicity of the rendering method, they usually render quite fast. With this method, you can add for example text labels or rain effects into the image.

The curve shape can be controlled the same way using VSL materials as in ray tracing and scanline rendering. To define a color for a post processed curve, use a Post Particles shader. Color, Alpha and Fade channels defined in that shader control the rendering. Fade defines transparency.



Note

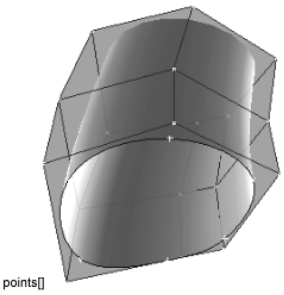
It is often impossible to add new post processed depth layers to an antialiased image from the ray tracer without getting some antialiasing errors. For example, a red sphere in front of a white background has some pink edge pixels, and rendering a post processed object between the red and white objects happens 'too late'. Such artifacts can be reduced by turning ray trace antialiasing off and by using image resolution down scaling after post processing.



Colored NURBS curves rendered in post processing

NURBS Surfaces

NURBS mesh is constructed from a two dimensional array of control points. Correspondingly, there are two sets of Count, Order, and Closed attributes: one set for each dimension.

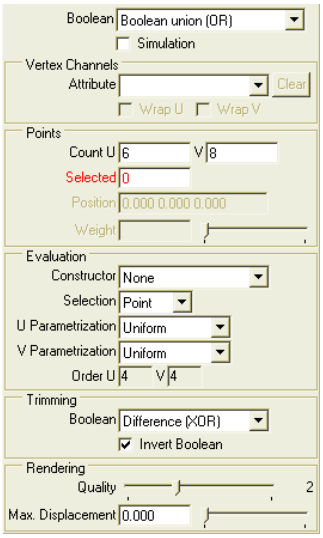


NURBS Mesh Property Gadget

Attribute: Selects a channel for pointwise value control. The value can be set for the selected points using the value gadget below the channel selector.

Clear: Removes the pointwise values of the channel in the Attribute field from all points.

Wrap U, Wrap V: Activates modulo wrapping into the unit interval for the selected channel component. This option is useful for UV coords channel and other texture parameter channels. Thanks to it, texture parameter can be interpolated correctly over the mapping seam on closed surfaces.



Count U: The amount of mesh control points in U direction. Reducing the point count usually changes the shape.

Count V: The amount of control points in V direction.

Selected: The index of the selected control point.

Position: The position of the selected control point.

Weight: The weight of the selected control point(s). Increasing the weight pulls the mesh towards the selected control points.

Constructor: Selects how child NURBS curves rebuild the mesh.

Selection: Selects geometry editing by single points or by complete row/column curves.

U Parametrization, V parametrization: Parametrization affects mesh shape significantly, and is relevant also in path animations. Uniform parametrization means that all mesh patches represent an equal parameter square in parameter space. The chord length parametrization computes the parameter division from the patch size.

Order U, Order V: The order of the mesh in U and V directions. The higher the order, the smoother the mesh. Order = 2 produces a polygonal mesh. The most usual smooth NURBS mesh type (Cubic mesh) has order 4.

Boolean: The trim curve operation. This becomes relevant when the mesh has several trim curves. By default, the exclusive OR operation is used.

Invert Boolean: Inverts trimming. If set, trim curves cut holes. If not set, trim curves leave their interior left.

Quality: The quality in photorealistic rendering. NURBS mesh rendering is based on a numerical algorithm. The quality value controls the amount of algorithm iterations. The higher the value, the more accurately the 'ideal' surface shape is computed.

Max. Displacement: If the value of this field is greater than zero, and if surface geometry shaders are mapped to the surface, the surface is displaced according to the bump height channel in ray trace rendering. 'Max. Displacement' defines the maximal amount of displacement.

The displacement value is a real distance value (the unit can be selected from 'Options/Window' pull down menu, 'Metrics' tab), so usually quite small values are sufficient (such as 0.01 meters=1 centimeter). The amount of displacement has a great impact on rendering speed, so it is wise to use a minimal 'Max. Displacement' limit value here. Note also that displacement is symmetric to both sides of the original surface. Therefore, a displacement map that is centered around zero and whose amplitude varies between -1 and 1 centimeters renders faster than a bump map whose amplitude range is from 0 to 2 cm. In former case, the 'Max Displacement' value can be 1 cm, whereas the latter case requires a limit value of 2 cm.

Note: Use of displacement mapping does not automatically increase the rendering accuracy to meet the new surface detail requirements. If the displacement map includes lots of details, the original surface must be reasonably dense and the rendering Quality value described above must be high enough. If rendering artifacts appear, you can increase the mesh density using the CountU and CountV values described above.

Particle Objects

A particle object consists of a number of simple geometry items, such as points or lines. Despite of their simplicity, particles can create interesting rendering effects, thanks to the following reasons:

- A particle cloud or flock can include a very large number of items.
- Particle appearance can be customized using VSL materials.
- Particle effects can be mapped on particles.

Particles can be rendered as scanline objects before ray tracing or in the post processing phase after ray tracing. Scanline particles can be controlled using a VSL material with a Scanline shader. Post particle rendering can be controlled using a Post particles shader.



Note

All usual surface attributes such as color, transparency etc. affect appearance of scanline rendered particles. The scanline system passes the result of its rendering to the ray tracer, which in turn evaluates the surface attributes when combining the scanline data with ray traced objects. Therefore, you can even bump map scanline particles by modifying the surface normal in a surface properties shader!



Note

With a post particle shader, you can set and randomize may useful properties such as particle color, transparency, geometry points and diameters.

All particle types have the following controls on the property window:

Rendering tab

- **Age Definition** - This selector defines, how the render engine sees the age of particles. The following three options are available:
  - *Relative Age* (default) grows from 0 at birth to 1 when particle dies, regardless of the actual total life time
  - *Age* after birth increases by time, but is not scaled to the 0..1 interval

- *Remaining Life* time decreases down to zero

Note that the gadget does not define the actual target channel for age definition. It must be selected from the Geometry tab. For example, age can be rendered directly to the Color channel.

- **Post Processing** - If set, the particles are rendered in post processing phase of the rendering pipeline. This happens after ray tracing and is purely raster rendering, but distance and other channels can be taken into account. Post particles have only a limited interaction with ray traced items, but they are very quick to render. A Particle effect must be mapped to the particles, otherwise they does not render anything visible.
- **Scan Line** - If set, particles are rendered by the scanline system before ray tracing, meaning that transparent particles show the items behind them, and reflective particles can mirror ray traced objects.
- **Ray Tracing** - If set, particles are rendered as ray traced objects. This is usually the most memory intensive and slowest solution, but gives the most realistic results. 1D particles are rendered as spheres, 2D particles as cylinders or cones with rounded ends and 3D particles as elliptic spheres.

Geometry tab

- **Count** - The current number of particles
- **Max Count** - The maximal number of particles. This limits how many items constructors such as emitter can add to the particle object.
- **Creation Rate** - Defines how many particles constructors will add during the animation. Zero rate means that the particle count remains the same during the animation. A negative value means that when a particle dies, it is no longer regenerated, and particle count starts decreasing.
- **Properties** - A list of attributes that can be assigned as a shared value common to all points or as a pointwise value.
  - *Speed* - Defines how fast particles move. The value is used when the Emitter construction is active.
  - *Life Time* - Defines how long particles live. When a particle dies, a new one is created to replace it. Therefore, the Life Time attribute has meaning only together with the emitter construction. You can select the type of age definition from the Rendering tab (relative 0..1 age, true age or the remaining life time).
  - *Color* - The usual diffuse color of particles.
  - *Velocity* - The motion vector of particles. Defines how much particle(s) move in one second.
  - *Transparency* - Transparency property for rendering.
  - *Reflection* - Reflectivity for rendering.
  - *Alpha* - The alpha channel value.
  - *Fade* - The Fade channel makes scanline particles translucent. Also some particle effects use the fade value for transparency control.
  - *Illumination* - Self illumination color. Used in scanline rendering.
  - *Optical Thickness* - A well known but in particle rendering only rarely useful optical property.
  - *Scope* - A material mapping attribute.
  - *UVW Coordinates* - Defines the position of each particle in the 'Emitter' and 'Surface' construction methods. This property is initialized to a random value when a construction is activated first time.
  - *Other user channels* - All custom channels of the Channels library are available here for rendering control purposes.

- **Common Value** - The base (initial) value of the currently selected attribute. If 'pointwise' is not set, the gadget shows the common value of all particles. This control also defines the value range of life time and speed for recreated particles, when a particle constructor is active. When an attribute is changed to pointwise, the per point value is initialized using the common value.
- **Pointwise** - Makes an attribute pointwise i.e. a value per particle point is allocated. Every particle can have its own private value for the attribute.
- **Point Value** - The 'per point' value of the currently selected attribute. When set, the pointwise value overrules the Common Value.
- **Render Channel** - The rendering engine channel to which a pointwise attribute is copied at rendering preprocessing time. This way all properties can be used to control particle shading.

Note: A pointwise value for all rendering channels (Color, Fade etc.) can be initialized using the map2Obj tool.

Constructor tab

- **None** - No particle construction applied. Particles can be freely moved around in 3D space.
- **Distribute over surface** - The particles are placed randomly around the surface of the sub object. To use many objects, collect them into a single level.
- **Emitter** - Particles are generated on the surface of the sub object with defined life time and velocity properties. New particles appear as old particles die. The following attributes control the behavior of emitted particles:
  - *Life time randomness* - Defines how much the life time of created particles varies. Value 1 will produce a totally random life time between zero and the object's common Life Time attribute.
  - *Speed randomness* - Randomizes the speed of emitted particles. If the value is zero, all particles have the same speed (defined by the common Speed attribute of the object).
  - *Direction randomness* - Randomizes the direction to which emitted particles travel. Value 1 will produce a totally random direction. Value zero will emit the particle to the exact surface normal direction of the creation point.
  - *Position randomness* - Randomizes the position of the particles to be created. If zero newly created particles are created on the surface of the emitter object. The bigger the value the larger the displacement.
  - *Colorize* - If set, the color of the sub object in the emission point defines the color of the emitted particle.

  In addition to these three controls, also the common/pointwise properties of particles defined in the Geometry tab control the particle behavior. Common values define the range for randomization and pointwise values are used until the particle dies and a new one is created to replace it.
- **Filler** - The particles are placed randomly inside the volume of the sub object. The sub object must define a solid volume; use analytic objects or freeform surfaces combined into a boundary surface solid.

1D Particles

A 1D particle object consists of a set of points. Points are suitable for modeling dust clouds, symmetric flares, stars and other objects which do not need a 'direction' property.



A moving sphere emitting 1D scanline particles

1D particle specific properties:

- A 1D particle renders in scanline as a round disk that is facing towards the camera.
- Scanline particle location can be displaced using a 'Scanline shader' in a VSL material. The 'Coordinates' channel defines the position.
- 1D scanline particle object defines the following UV space (UV Coords channel): U runs from 0 at the middle point of a particle to 1 at the edge of the particle. V runs from zero at the first particle to 1 at the last particle.
- The 'Size' property at the geometry tab defines the radius of particle discs.
- In post processing, 1D particle drawing can be controlled by using the following channels in a 'Post particles' shader:
  - *Color* - particle color
  - *Fade* - particle transparency
  - *Coordinates* - position of a particle
  - *V direction.X* - the first sub channel defines particle radius
- Ray traced 1D particles are rendered as spheres, unless the **Ray trace as quadrangle** option is set, in which case each particle is rendered as a flat square facing towards the camera. Quadrangles define also UV coordinates, which makes texturing easy.

2D Particles

A 2D particle object consists of a set of straight lines. Lines are suitable for modeling rays, fur etc.
2D particle specific properties:

- Particle ends can be displaced using a 'Scanline shader' in a VSL material.
- 2D particle object defines the following UVW space (UV Coords channel) in scanline drawing: U runs from 0 at the start point of a particle line to 1 at the end of the particle. V runs from zero at the center line of a particle to 1 at the edge of the particle. W runs from 0 at the first particle to 1 at the last particle.
- 2d particles have the following additional properties:
  - *Diameter1* - The diameter of the start point of the particle in millimeters.
  - *Diameter2* - The diameter of the end point of the particle in millimeters.
  - *Length* - The length of particles (in the current measuring unit).
- In post processing, 2D particle drawing can be controlled by using the following channels in a 'Post particles' shader:
  - *Color* - particle color
  - *Fade* - particle transparency
  - *Coordinates* - start point of a particle
  - *U direction* - end point of a particle
  - *V direction X and Y* - two first sub channels define the radius values at the end points



2D particles sprayed onto a surface

3D Particles

3D particles define a full 3D coordinate system. They are therefore suitable for effects that require full orientation control in 3D space. The current post effect set does not require such control.

- 3D particles can be ray trace rendered as as spheres/ellipsoids or as rectangular planes. The latter alternative, available via the **Ray Trace Rectangles** option, is a very suitable feature, when a large number of UV textured rectangles is needed. 3D particle rectangles are rendered in a memory efficiet manner and they define UV coordinates. You can easily texture them to represent tree leaves etc.

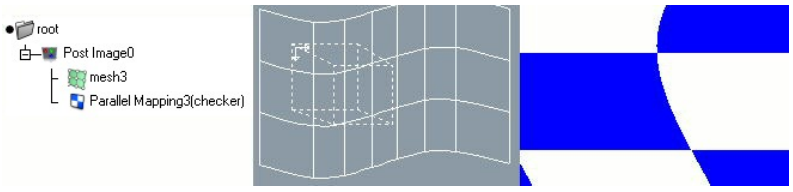
The post processing control channels of 3D particles, available in a 'Post particles' shader, are:

- *Color* - particle color
- *Fade* - particle transparency
- *Coordinates* - the middle point of a particle
- *U direction* - a control point defining the first elliptic radius
- *V direction* - a control point defining the second elliptic radius
- *Normal* - a control point defining the third elliptic radius

Post Image

The Post Image is a special kind of particle object. It is classified as a particle object because the 'Render particles' post effect module renders it at the post processing phase. It is a level object, which can contain another geometry object. The sub object can be a rectangle, a NURBS mesh or another 2D geometry.

The geometry defines an image plane warping. Each point of the rendered image corresponds a point in the surface. The image plane can be colored by using a material which includes a 'Post particles' shader. By animating the material mapping or the sub geometry, you can create animated waves, warps and other interesting effects.



A post image including a NURBS mesh mapped with a checker pattern. A part of the result image on the right.

Path Level

The Path Level object is used - as the name suggests - for path animations. It uses two sub objects: the actual motion path and an axis curve, which defines the horizon plane for banking control.

Path Level implements some useful banking behaviors and an option for constant speed motion. These are not available when using a plain NURBS curve as a motion path.

Path Level has the following controls in its property gadget:

- Path rotation type - one of the following banking options:
  - No banking - All banking with respect to the horizon is eliminated.
  - Natural banking - The target is rotated to match the normal of the curve in a non-cumulative manner.

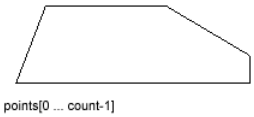
- Cumulative banking - The target is rotated to follow the curve using the shortest rotation route between each two consecutive positions. This usually banks the target i.e. rotates it around the curve. Banking cumulates towards the end of the path.
- Aeroplane banking - The banking behavior simulates aeroplane. Banking increases when the path turns and returns back to zero during straight sections of the path. The length of the horizon axis under the path level controls the amount of banking. Making the axis longer decreases banking.
- No rotation - The targets are not rotated at all.

- Constant Speed - selects point density independent motion.

Polygon

An analytical polygon is basically a trimmed plane. Its geometry is defined by three or more geometry points on a plane.

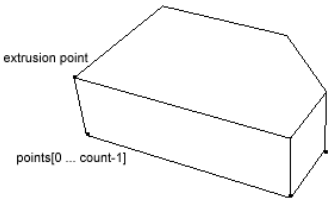
Because polygons are planar, its volume is a half space. In other words, the one side of the polygon is said to be inside the polygon and the other side the exterior. For example, you can use the polygon object to split an object into two parts using the boolean operations.



Polyhedron

A polyhedron is derived from a polygon by introducing so called 'extrusion point'. The extrusion vector can be computed by subtracting the first geometry point from the extrusion point. The top and base planes are perpendicular.

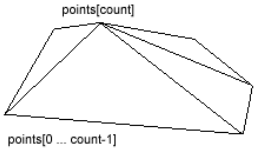
The polyhedron supports fast Boolean operations.



Polymid

A polymid object is derived from a polygon by introducing a new 'Apex' attribute.

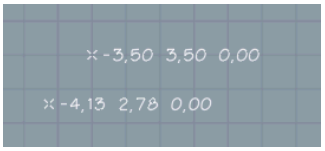
The polymid supports fast Boolean operations.



Position Mark

The position mark object is displays the coordinates of its location in view windows. You can add position marks to you scenes to show information about important positions.

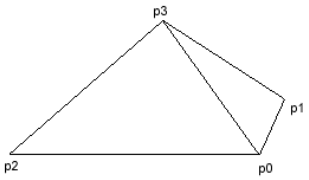
The properties of the position make object are:



- Photorealistic rendering method defines how the geometry is rendered.
  - **Ray Trace** - The mark and its text is rendered using ray tracing
  - **Scanline** - The object is rendered using scanline method
  - **Post Process** - Post processed (= raster) rendering
  - **Pseudo 3D** - A special correction is applied to the geometry, to obtain a constant appearance the same way as in real-time drawing. Thickness and size of the object will appear constant, regardless of the distance from camera or the viewing angle.
- **Text Size** - The size of numerical digits (per mills of full image width), which display the coordinates in view windows
- **Thickness** - The radius of the line used in photorealistic rendering (per mills)
- **Position** - The XYZ coordinates of the object

Pyramid

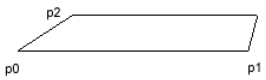
An analytical pyramid is defined by four points: three of them define the base rectangle and the fourth point defines the apex.



Rectangle

An analytical rectangle is defined by three points. Opposite edges are always parallel to each other. The fourth corner point is computed from the given three points internally.

The volume defined by a rectangle is an infinite half space, trimming out the other side of the plane. Which one of the sides is 'inside' depends on the winding order of the points. The interior side is defined by the surface normal computed using the right hand winding rule.



For example, one can use a rectangle object to split an object into two parts using the Boolean operations.

Roof

The roof object creates a standard roof shape from its parameter curves: a roof edge polyline and the optional roof top polylines. If no roof top lines are defined, the object computes a straight line across the roof, which splits the roof symmetrically.

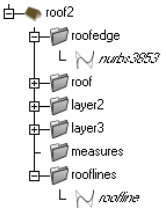
Roof object can operate as an individual object, or in conjunction with a building object. When placed inside the hierarchy of the building object, the building object automatically stores a roof edge line computed from the walls into the roof object.



Note

Roof object's Build option can be disabled and the geometry can be replaced with any user modeled roof shape. Wall object's Extend to Roof will then fit the walls with the custom roof shape.

The hierarchy of the roof object contains the following sub levels:



- roofedge: Contains the polyline, which defines the roof edge.
- roof: contains the auto generated top surface (coating) of the roof (a polygonal SDS mesh). You can place material attributes into this folder. Editing the geometry while the roof's Build option is enabled makes no sense, because next construction refresh will remove editing.
- layer2: the second roof geometry layer (support below the coating).
- layer3: third roof geometry layer (the actual body of the roof with insulation etc).
- measures: the measure lines showing the dimensions of the building are stored here, when the respective option is turned on.
- rooflines: the Roof Line tool places the polylines, which define the roof top, to this folder. Roof lines can be created also manually using the NURBS curve tool.

The roof object has the following controls in the Property window's Spec tab:

Roof Geometry options:

- Build: enables automatic construction. Generated roof geometry can be edited manually only after Build is turned off. Turning Build on removes manual changes. When the option is enabled, only the parameter data is saved to disk files, which usually makes the file size small.
- Auto Texture: if enabled, the roof object automatically adds and updates material mapping objects for walls, ceiling, floor and base.
- Roof End: Select from three alternatives, Straight (gable roof), Match Roofline (user defined angle) and Beveled (hipped roof).
- Height: Basic height of the roof. Note that when roof lines are included, the lines define the height. Move the roof lines to change the height.
- Eave Width: horizontal distance from the wall to the roof edge.
- Eave Elevation: Roof height is measured starting from the base height of walls. Eave Elevation moves the actual roof edge up or down from the zero level.



From the left: Straight, Beveled and Match Roof Line options

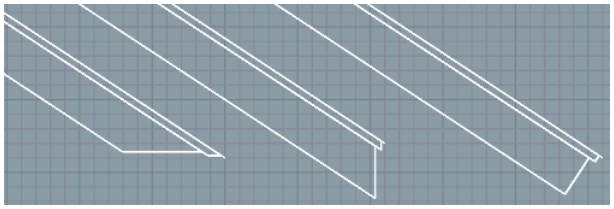
Sub Layer 1 and 2 have identical options:

- Add Layer: the respective roof layer geometry is added to the scene.
- Filled Top: build a surface between the roof layer and the layer above it. Usually the layer is invisible inside the roof structure, so it can be left out from the model just to make rendering a bit faster.
- Bottom Fill: Defines how the roof layer is filled from below. Perpendicular fill adds a surface which is parallel to the roof top. Horizontal option fills the bottom with a flat horizontal surface.



Parallel and Horizontal roof layer fill

- Roof Edge: Can be Horizontal, Vertical or Perpendicular. See the picture on the right.
- Thickness: the thickness of this roof layer.
- Offset: Distance from the edge of the upper layer to this layer.



From the left: Horizontal, Vertical and Perpendicular roof edges

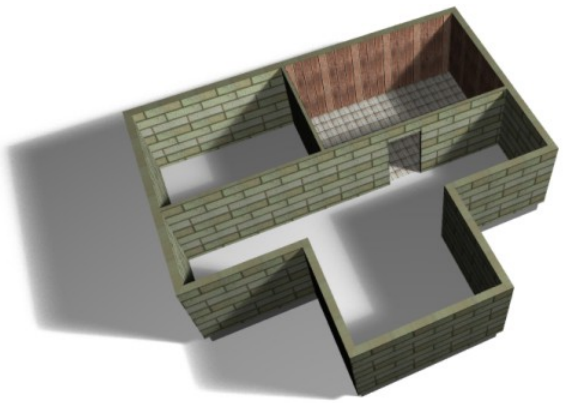
Measure Lines:

- Measure Lines: Enables measure line drawing.
- Offset: the distance from the wall polylines to the measure lines.
- Text Size: defines the font size (in pixels) for the displayed dimension values.
- Arrow Size: size of the end arrows (in pixels)

Room

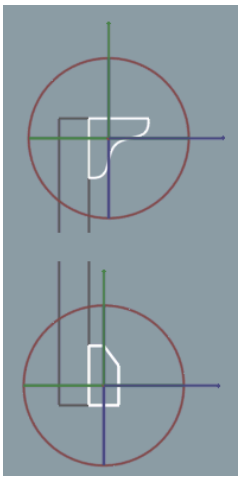
The room object creates new surface layers around the surrounding walls, roof and ceiling. The room surfaces can be decorated independently from the rest of the house. The object can also generate automatically mouldings for the floor-wall edges and the wall-ceiling edges.

The surfaces are extracted automatically from the parent building object. The position indicator of the room object defines, which room space will be decorated. Moving the indicator to a new room moves the decorations as well. A room object moved outside the hierarchy of the building is inactive.



One room object added to the building

The room object has two special sub folders called *floormouldingsample* and *ceilingmouldingsample*. When an example section of a moulding is placed in either of these folders, the object automatically duplicates, stretches and transforms the moulding to travel around the respective floor/ceiling edge of the entire room. The example moulding can be placed anywhere in 3D space. Its object space must be properly oriented: the green y axis must point upwards and the blue z axis must point outwards from the wall side of the moulding.



Object spaces for floor and ceiling mouldings



Note

Repeating moulding structures, such as wooden panel walls, can be defined using a replicator object.

The Property window's Spec tab controls for the room object are listed below. Some of these options are available in the tool bar, too.

Room Options:

- Build: Enables room construction. Generated surfaces can be edited manually only after Build is turned off. Turning Build on removes manual changes. When the option is enabled, only the parameter data is saved to disk files, which usually makes the file size small.
- Texture Floor: Adds a floor surface to the room.
- Texture Ceiling: Adds a ceiling surface to the room.

Walls:

- Texture Walls: Enables wall decoration. When set wallpaper surfaces are created.
- Wall Range, From, To: When Wall Range is set, only the walls with indices between From and To are decorated. Indexing starts from the wall closest to the indicator.



Note

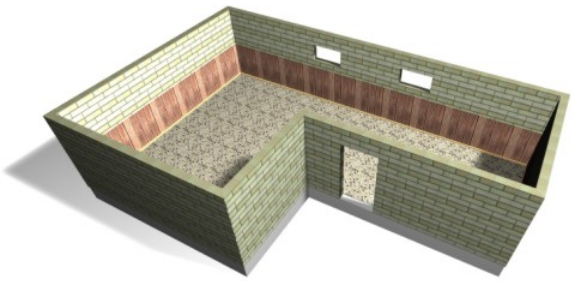
It is possible to add several room objects to a same room space, in order to apply several different textures to the walls. Only one room object should have the roof and floor surface creation enabled.



Two room objects applied to the same space

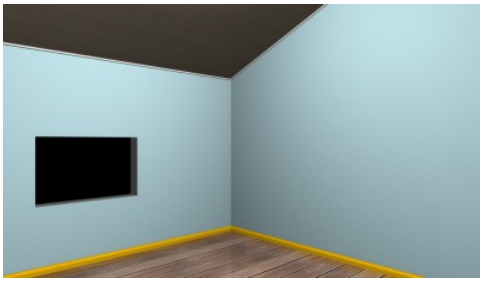
Wall Height:

- Start, End: The relative height of the textured area of the walls.
- Extend to Roof: Enables automatic wall surface extension to the limiting roof surface. This option can be useful, when the room space is not limited by a flat ceiling, such as attic rooms.



Start=0 and End=0.5 textures lower half

- Wall-Roof Penetration: A tolerance factor, which helps to close the gap between extended wall surfaces and the roof.
- Fitting Interval: The wall surface height is matched with the roof at every corner and where a roof line crosses the wall. If the roof has a complex or even a smoothly bending shape, a low fitting interval (such as 0.1 meters) may be necessary. This option defines a maximal distance between two fitting positions.



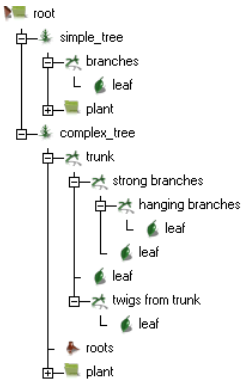
emphasis>A room in the attic

Measure Lines:

- Draw Measure Lines: Enables measure line drawing. The lines are always drawn inside the room.
- Offset: The distance from the wall polylines to the measure lines.
- Text Size: Defines the font size (in pixels) for the displayed dimension values.
- Arrow Size: Size of the end arrows (in pixels)

Seed Object

Plant seed object is the highest hierarchy level of a plant object. It builds a fractal plant using growth rules defined by its attributes and child objects (branches and leaves). The seed object includes a 2 point line geometry, which defines the initial growth direction and speed. All dimensions of the plant (branch diameters, leaf sizes etc.) are relative to the length of the line. Plants have some gravity related attributes, and the direction of the z axis (the blue object handle) defines the direction of the gravity.



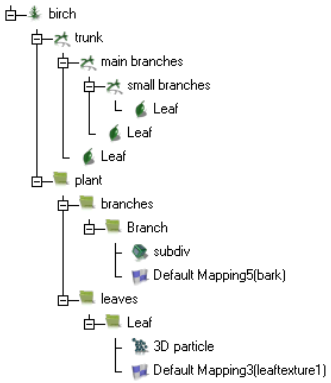
Plant hierarchies. The simple hierarchy on the top grows the roots and the branches using the same rules. More complex trees use sub objects to define how trunk and branches grow.

The seed object generates the geometry of the plant under its own hierarchy. You can assign materials and constant attributes (color etc.) to the created plant hierarchy levels. Do not change the actual geometry items, because they are regenerated every time when a growth parameter changes and when loading a plant object from a file. If you need to edit the geometry items, move them away from the plant hierarchy.

To create a plant seed object:

- Activate the Plant tool from the tool bar or from the pull down menu.
- Click on a view window to define the point where the plant grows.
- Move the mouse upwards to define the direction and speed of the growth. Click second time to finish the tool.

The plant tool automatically inserts a branch object into the seed object in order to set up a fully functional plant.



Materials assigned to the tree geometry

To change the direction of growth from the root, point edit the plant's axis line. Do not rotate the tree, because that would rotate the gravity direction as well. It is probably a good idea to make the generated plant geometry invisible while point editing the axis line.

Plant seed object includes the attributes described below.

Height: Controls the height of the plant. The value defines the approximate length of growth during the first year. The value is not exact because Random Length and other branch attributes contribute into the result.

Thickness: The thickness of the plant. The value defines the initial radius of the plant in the point where growing starts. Note that the branch profile and other attributes may change the radius from the value defined here.

Random Seed: Seed value for the computation of randomized plant properties. Change this to get unique trees.

Render Plus: Adds plant complexity for photorealistic rendering. The idea is to use easily manageable preview objects in the modeling stage and render the full detail only in the final images. The defined value is added to the plant's age, and the leaf density parameters are multiplied with (1+value). Because the number of branches usually grows exponentially by age, plant's complexity grows very rapidly at every render plus step. Usually a small value like 1 is enough; it typically increases the complexity by a factor of 8.

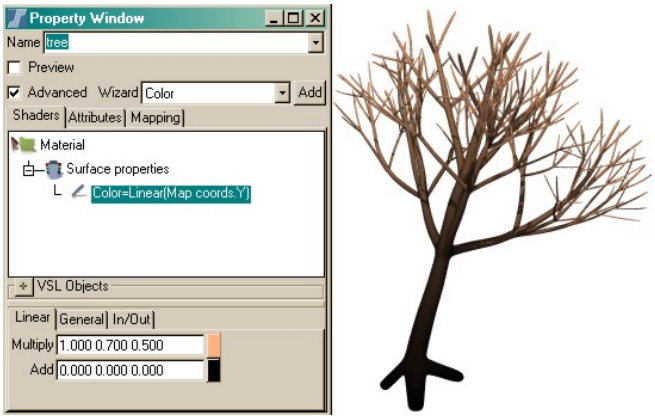
Max Complexity: Limits plant's complexity (number of branch segments + number of leaves) to a safe value range. Some attributes increase complexity exponentially. Mistyping a high value could rapidly consume all memory, cause heavy memory swapping and make it hard to recover from the error. The complexity limit protects you from this kind of troubles. The default value of complexity is a plant root's class attribute; use the preferences window to adjust it to match the performance of your computer system permanently.



Modeling vs. rendering complexity

Set UV: If set, the following information is stored into the tree geometry's UV Coords channel:

- UV Coords.X - 0..1 parameter around the branch
- UV Coords.Y - 0..1 parameter, which starts from zero at the root ends, and reaches 1 at the end of branches.
- UV Coords.Z - the ratio (current branch radius/radius at the root)



A simple color gradient material

Set Colors: If set, the plant computes and stores colors for leaves and other items which include colorization features. Alternatively, the plant can be textured using materials in the plant hierarchy.

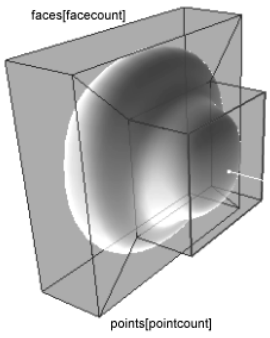
Subdivision Surfaces

Realsoft 3D's Subdivision Surface implementation is based on Catmull-Clark surfaces. However, Realsoft 3D supports rational SDS surfaces - there can be a 'weight' parameter associated with each control point. Increasing the weight pulls the corresponding surface point towards the control point in question.

A subdivision surface object is defined by a set of control points and a set of faces. Each face can refer to three or more control points.

There are many pointwise attributes that can be defined, such as color, illumination, or transparency. Also, there can be pointwise values for any user defined channel.

Vertex sharpness can be set for any set of vertices. It can be Rounded (default), Semi Sharp or Sharp. There's also a similar control for edge sharpness. Additionally, a 'Free' edge sharpness rounding can be defined numerically.



SDS objects can include multiple UV channels defining how textures are applied on the surface.

Realsoft 3D's SDS implementation also supports per face materials: a material name and a set of UV parameter values (one per vertex in the face) can be attached to any face.

SDS Properties

Handles

SDS supports a small but very powerful set of handles through which most commonly needed tools can be applied.

There are three sets of handles: point, edge and face handles, which can all be selected. When a handle is selected, it typically shows a set of sub handles. For example, when a face handle is selected, also the face normal handle is shown for the selected face.

The usual selection modifiers and the drag box selector are available when selecting SDS handles.

Point Handles

The point handles are shown when the SDS object is in the point editing mode.

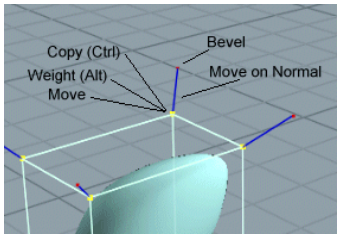
Each SDS point acts as a translation handle by default.

Points can be duplicated by dragging them with the Ctrl key pressed.

The weight parameter associated with the points can be changed by dragging the point with the Alt key pressed.

The 'Move on normal' tool can be activated by dragging the surface normal handle.

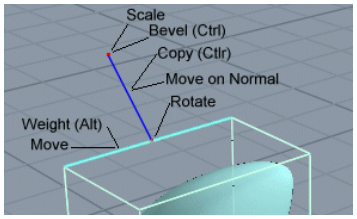
The Bevel tool can be activated by dragging the end point of the surface normal handle.



Handles shown in the point editing mode

Edge Handles

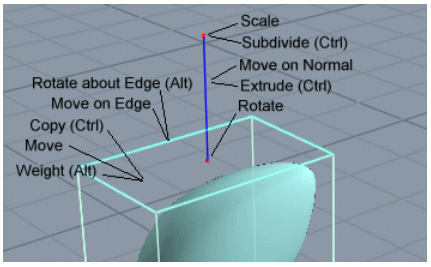
The edge handles are shown when the SDS object is in the edge editing mode.



Handles shown in the edge editing mode

Face Handles

The face handles are shown when the SDS object is in the face editing mode.



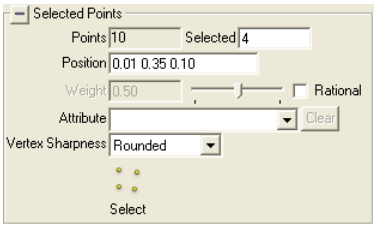
Handles shown in the face editing mode

Property Gadget

The SDS property gadget has the following frames: SelectedPoints, SelectedEdges,Rendering, MultiUV, Faces.

Selected Points frame

- Points - the number of vertices in the SDS object
- Selected - index of the selected vertex. If more than one vertex is selected the index of the first selected vertex is shown in the conflict color.
- Position - 3D position of the selected vertex.



- UV - current UV channel value for the selected vertex. The current UV channel can be selected from the list in the Multi UV frame.
- Attribute - list of pointwise (defined separately for each vertex) attributes. Use this to select the pointwise attribute.
- Value - selected pointwise attribute value for the selected vertex.
- Clear - removes the selected pointwise attribute from the SDS object.
- Rational - if checked pointwise weights can be controlled. Unchecking this checkbox removes pointwise weights.
- Weight - sets the weight for the selected vertex/vertices. Can be used only when the Rational checkbox is checked.
- Vertex sharpness - sharpness of the selected vertices. Possible values are
  - Rounded
  - Semi Sharp
  - Sharp
- Select - selection mode, select by Point, Edge or Face.

Selected Edges frame

- Edge Sharpness - sharpness of the selected edges. Possible values are

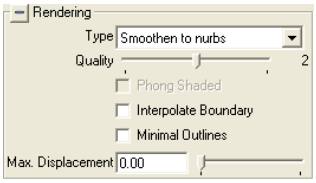


  - Rounded
  - SemiSharp
  - Sharp
  - Free

  If free is selected, then the edge rounding can be set numerically.
- Edge Rounding - can be set to edges that have Edge Sharpness set to Free.

Rendering Frame

- Type - How the SDS object will be rendered. The possible values are
  - Smoothen to nurbs - the surface is rendered using bicubic NURBS patches
  - Smoothen to rectangles - the surface is rendered using bilinear phong shaded quadrangles
  - Smoothen to triangles - the surface is tessellated to phong shaded triangles



  - Polygonal - The original control polygon is rendered without tessellation. Normal smoothing can be turned on using the Phong shaded option described below.
- Quality - Rendering quality, the higher the value, the smoother the surface. Controls how many NURBS patches are fitted into the surface.
- Phong Shaded - If set, normals of a polygonally rendered SDS surface are interpolated using Phong shading making the surface appear smoothly curved. This method works well only with relatively dense control polygons.
- Interpolate Boundary - If set sets maximum crease value for boundary edges (i.e. edges that are not shared be two faces).

- Minimal Outlines - If set, the density of outlines appearing on the rendered surface in outline rendering mode is minimal (i.e. matches the density of the control polygon). If unset, NURBS patch fitting density defined by the Rendering Quality setting above and triangularization of polygonal surfaces are visible in rendered outlines.

- Max. Displacement - If the value of this field is greater than zero, and if surface geometry shaders are mapped to the surface, the surface is displaced according to the bump height channel in ray trace rendering. 'Max. Displacement' defines the maximal amount of displacement.

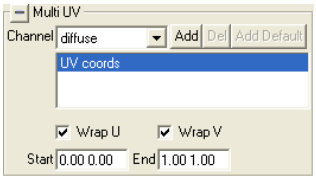
  The displacement value is a real distance value (the unit can be selected from 'Options/Window' pull down menu, 'Metrics' tab), so usually quite small values are sufficient (such as 0.01 meters=1 centimeter). The amount of displacement has a great impact on rendering speed, so it is wise to use a minimal 'Max. Displacement' limit value here. Note also that displacement is symmetric to both sides of the original surface. Therefore, a displacement map that is centered around zero and whose amplitude varies between -1 and 1 centimeters renders faster than a bump map whose amplitude range is from 0 to 2 cm. In former case, the 'Max Displacement' value can be 1 cm, whereas the latter case requires a limit value of 2 cm.



Note

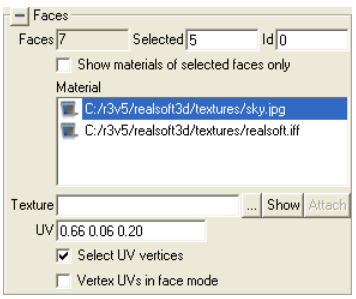
Use of displacement mapping does not automatically increase the rendering accuracy to meet the new surface detail requirements. If the displacement map includes lots of details, the original surface must be reasonably dense and the rendering Quality value described above must be high enough. If rendering artifacts appear, you can apply the SDS Smooth tool to increase the control polygon density.

Multi UV frame



- Channel - possible UV channels.
- Add - add the selected channel as a UV channel.
- Del - Delete the selected UV channel. The selection is shown in the list view gadget.
- Add Default - adds the default UV channel (UV coords).
- List of UV Channels added to the object.
- Wrap U - If checked interpolates U to nearest modulo at the boundaries defined by Wrap Start and Wrap End. Check this if the UV space is closed in the U direction.
- Wrap V - If checked interpolates V to nearest modulo at the boundaries defined by Wrap Start and Wrap End. Check this if the UV space is closed in the V direction.
- Wrap Start - Minimum boundaries (U, V) for modulo wrapping. This should usually be 0.0 0.0 for texture maps.
- Wrap End - Maximum boundaries (U, V) for modulo wrapping. This should usually be 1.0 1.0 for texture maps.

Faces frame



- Faces - number of faces in the SDS object.
- Selected Face - index of the selected face. If more than one face is selected the index of the first selected face is shown in the conflict color.
- Selected faces only - if checked the Material list view below shows only those face materials and face textures that are attached to the selected faces. Otherwise all face materials and face textures are listed.

- Material - shows the face material and face texture names of the selected faces or of all faces. This control has a context menu where the selected face material or face texture can be detached from the faces. If Selected faces only is checked then the face material or face textuer is detached from the selected faces. Otherwise it is detached from all the faces.
- Texture - Select a image file to be used as a face texture for the selected faces. Click the Attach button to attach the face texture.
- Attach - Attach the selected image file to the selected faces as a face texture.
- UV - the face material or face texture UV for the selected UV vertex. Check the Select UV vertices checkbox and select a UV vertex in the UV Editor window. Now the UV control can be used to set the face material or face texture UV for that vertex numerically.
- Select UV vertices - if checked UV vertices can be selected in the UV Editor. Otherwise faces can be selected.
- Vertex UVs in face mode - if checked per vertex UV values are shown in the UV Editor when the SDS object is in the Face selection mode. Otherwise face material and face texture UVs are shown.

Additional Controls in The Wire Tab

- Material Rendering - Defines how the SDS object is rendered in shaded mode.
  - Single - Only one face material or face texture per face is rendered. This is much faster than the Full mode.
  - Full - All face materials and face textures as well as mapping objects are considered. It is often preferable to use low Texture Quality settings when using this mode.
- Force Tessellation - If set, triangular tessellation of faces which have more than 4 vertices is shown in wireframe drawing.

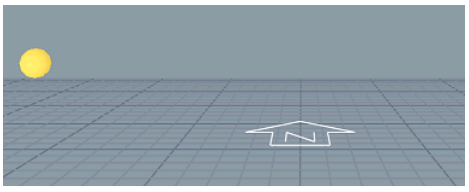
SunMoon Object



SunMoon object manages its first child object to simulate the position of the sun or the moon in a given date and a location.

The date is defined using local time. The applied time zone correction is geographical; the computation may include one hour error at certain time zones. The daylight saving time is not considered.

The location is defined by the usual longitude and latitude coordinates. Both coordinates should be given in decimal degree form (not as degrees.minutes.seconds).



Sun's position at Tampere, Finland 21 o'clock in the midsummer evening

To simulate the position of the sun or the moon, the sunmoon object rotates its first child object by its pivot point. This means that the azimuth and altitude angles of the selected target (sun / moon) can be found from the Rotate values of the object space of the first child. The first component, the heading angle, is the azimuth and the second component, the pitching angle, shows the altitude. Zero heading corresponds to the geographic north (= negative z axis of the sunmoon object space).

A sunmoon object can be created using the Sunmoon Tool. Some examples:

Point light sun

1. Create a point light, which represents the sun. You can add some visible objects like a glowing, shadow invisible sphere around the point light. Make sure all components of the sun are collected under a single hierarchy level. If not, select all components and apply Drop to level from the popup menu of the select window.



Note

The position of the sun does not matter. However, its distance from the origin should be such that the visual size of the modeled sun appears correct when viewed from the origin. The distance should also be large enough (e.g 1000 times the size of the rest of the scene). Otherwise the position of the sun might change when the camera moves around the scene. However, do not try to use the real astronomical scale; the accuracy of computers is not sufficient for shadow computations from a light source that is millions of kilometers away from the illuminated objects.

2. Select the sun and click the SunMoon tool. It is located in the Light Sources tool tab.

3. Adjust the date and the location using the toolbar controls.

Distant light sun

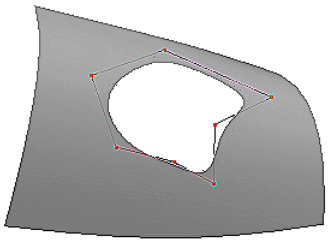
1. Create a distant light, which represents the sun. The position of the light does not matter, but it should be carefully aimed towards the world origin. As with the point light, you can add some visible objects around the light. Collect all components of the sun under a single hierarchy level.

2. Select the modeled sun object and click the SunMoon tool.

3. Adjust the date and the location.

Trim Curves

A trim curve is just a NURBS curve defined in the parameter space of a NURBS mesh. Trim curves are defined as sub objects for a NURBS mesh.



Any number of trim curves can be inserted into a mesh, in which case they are Boolean operated. The operator type can be selected from the mesh attributes.

Trim curves are derived from NURBS curves. In addition to attributes defined by NURBS curves, trim curves have the following attributes:

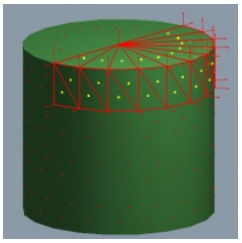
- Cut - if TRUE, the trim curve acts as true trim curve, trimming the parent. If FALSE, the curve doesn't trim the parent mesh. Such a curve is useful in 'welding'.
- Invert - If TRUE, the inside / outside property is inverted (controls whether the surface points inside or outside the curve are trimmed out).
- WrapU, WrapV - These attributes are meaningful only for trim curves of closed meshes. They tell the trim curve to close through the shortest route in the parameter space of the parent mesh. Thanks to this option, a trim curve can travel over the u=1/v=1 boundaries of a closed mesh.

Triset

Triset is a geometry, which consists of a number of triangular faces. Faces usually share the vertex data: connected triangles can refer to same 3D vectors. Each corner of a triset face has its own private normal vector. Normal vectors can be adjusted to achieve sharp or smooth looking shading over face boundaries.

Triset is much simpler object as the Subdivision object. It is also much more memory efficient. Its main function is to represent imported polygonal objects. If a triset geometry needs advanced editing, it can be converted to the SDS form, edited and then converted back to the efficient triset form.

Triset property gadget has the following controls:



Faces: Displays the number of triangles in the selected triset object.

Points: Displays the number of vertices.

Selected: The index of the selected vertex in point editing mode.

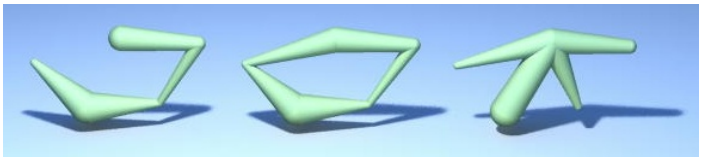
Position: The position of the selected vertex in point editing mode.

Phong Fix: Controls how rapidly illumination is faded towards the surface tangent. Normal interpolation of triset faces does not look smooth when ray traced illumination hits the surface at low angles, because the polygonal surface geometry starts casting shadows and blocking the illumination. By fading illumination at low angles, these shading errors can be made less visible.

Select: Edit mode selector. Trisets can be modified using point, edge and face handles. In face edit mode, selected faces display their surface normal handles, which can be edited using the mouse.

Tube

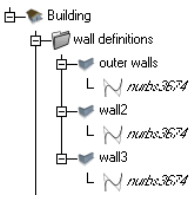
The tube object connects its sub objects (analytic spheres) with cylinders and cut cones, to form a continuous tube. It has only one object specific control: the **Type** of the tube, which can be open, closed or star like. The star alternative connects the first sphere to all other spheres.



The three tube types: Open, Closed and Star

Wall

The wall object defines a sequence of building walls. The object generates visible geometry only when placed into the 'wall definitions' sub folder of building object. The wall object needs one parameter object, a NURBS polyline. Currently smoothly bending NURBS curves define the wall by the control polygon.



Wall objects

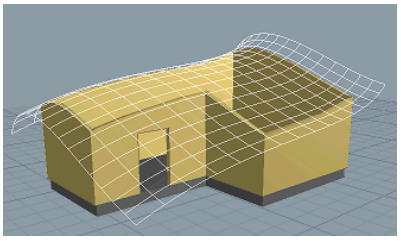
The sub wall object has the following options available in the Property window's Spec tab:

Wall Geometry options:

- Thickness: Thickness is wall specific. Outer walls are usually thicker than inner walls.
- Side: defines to which side of the parameter polyline the geometry is built.
- Snap Start, Snap End: If set, the respective end of an open wall segment is expaned to match the closest other wall. Room and buidling dimensions can be edited easily, because it is not necessary to adjust every wall dimension separately.
- Virtual: virtual walls divide space but do not generate visible wall geometry. They are considered in wall snapping and in room divisions. For example, it is possible to divide the floor of a room into two separate areas, which have different materials, by splitting the space with a virtual wall. Wirtual walls also provide a way to cut staircase holes through ceilings and floors in buildings, which have several floors.

Roof Fitting options:

- Extend to Roof: when enabled, the wall height is adjusted automatically so that the wall joins with the roof seamlessly. Outer walls often need this option, whereas inner walls stop at the ceiling.
- Wall-Roof Penetration: A tolerance factor, which helps to close the gap between extended walls and the roof.



Walls extended to fit a curved roof

- Fitting Interval: The wall height is matched with the roof at every corner and where a roof line crosses the wall. If the roof has a complex or even a smoothly bending shape, a low fitting interval may be necessary. This option defines a maximal distance between two fitting positions.

Material Mapping Objects

Introduction

Material Mapping objects allow the user to assign material properties to geometric objects. They also allow post effects to be associated with objects. For example, one can associate a lens flare post effect with a set of 1D particles to get a flock of lens flares (perhaps acting under the influence of a gravity field).

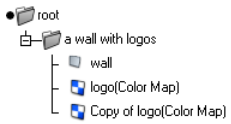
The system follows the fundamental Realsoft 3D philosophy: a 'wooden sphere' consists of two sub objects, a sphere and a wood material mapping object.



Material Mapping Tool Set

Another idea behind the system is that different volumetric mapping methods, such as parallel mapping, spherical mapping, etc. are represented using a geometric object. For example, the material space for a spherical mapping is visualized using a sphere object.

This system has two remarkable features:



First, it allows any number of materials to be assigned per object.

Second, the tool set for editing geometric objects and material mapping objects is the same. One can move and rotate textures and other material effects as easily as one can move geometric objects.

A third advantage is that geometric objects visualize the material space in a very intuitive way. For example, the perspective projection is visualized using a pyramid object.

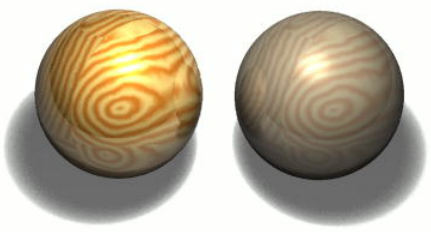
All material mapping objects define the following attributes:

Material - Name of the material to be mapped.

Effect - Name of the post particle effect to be mapped.

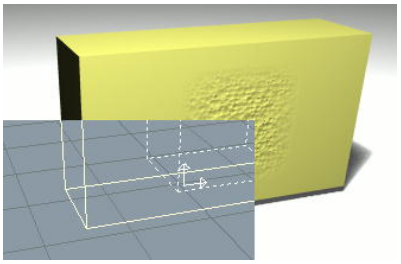
Translate, Scale, Rotate, Skew - Material coordinate space transformation that is applied after the mapping specific transformation is computed. For example, by changing the 'Scale' factors to 0.5, you can make a texture pattern 50 % smaller.

Scope - Controls how strongly the material affects the object.



Full wood and 'scope=0.5' wood

X, Y, Z Fading - Values greater than zero fade the material away at the both ends of the 0..1 unit interval of the respective mapping coordinate dimension. In other words, the material fades smoothly away at the mapping geometry edges. The value defines the width of fading area: value 1 will start fading from the middle point of the mapping. Value 0.1 will produce a 10 % fade area, 5 % at both ends.



A bump map faded smoothly at parallel mapping edges, X and Y Fading = 0.5

Priority: Evaluation order control. By default, material maps are evaluated in the order they appear in the hierarchy, deep sub level materials last. The hierarchy defined evaluation order can be overruled with the mapping priority setting. The higher the priority, the earlier the mapping will be evaluated.



Note

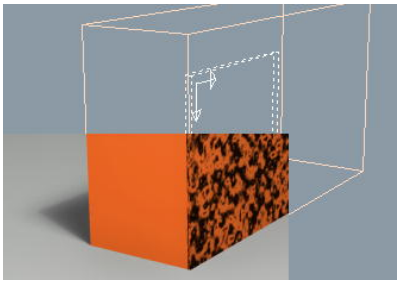
This means that the associated material will have a weaker effect! Later evaluated materials can overwite the result. Some special materials, such as a global illumination shader, may work best if it is evaluated last. Because most material mappings have the default priority 0 and only some built in objects use evaluation priority -2 .. +2, a value -5 is usually sufficient to guarantee that the mapping is evaluated last.*Note 1*: The priority control is only rarely needed. Usually the hierarchy provides a much better way to control the order. However, in some cases, it would be too laborious to place the mapping to the end of all involved sub hierarchy levels. In such cases this feature can be a real time saver.



Note

The priority controls only material maps in the object hierarchy. SDS face materials don't use any mapping object and therefore they are always evaluated last.

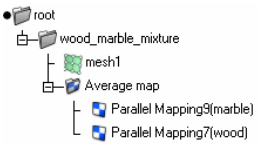
FiniteX, FiniteY, FiniteZ - If set, the material effect is limited inside the corresponding dimension of the mapping object (to 0..1 coordinate range)



A thin layer of stone mapped to a cube using a parallel map with the 'Finite Z' option set

Average Mapping

The average mapping object is a level object which uses any number of other material mapping sub objects. Material effects defined by the sub objects are evaluated and averaged and then assigned to the target objects. The Scope attribute can be used for weighted averaging, i.e. to control how strongly each sub mapping object contributes to the result.



Cube Mapping

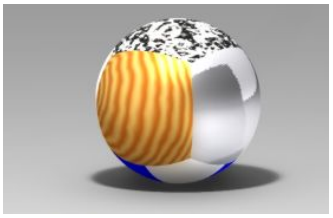
The cube mapping object allows six different materials to be associated with an object - one for each side of the cube. This is particularly useful when setting up environment maps.

A cube mapping has the following object specific controls:

Volumetric Map: This cube mapping type uses 6 pyramid maps, one from each face of the cube into the center point of the cube.

Environment Map: The material is chosen using the reflected ray direction. This mapping can be used to simulate reflective surfaces inside a room interior.

Surface Normal Map: The material is chosen according to the surface normal direction.

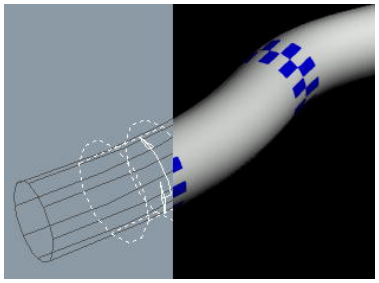


Front, Back, Top, Bottom, Left, Right Material: The six materials, from which one is chosen for the examined point using the selection rules (Volumetric/Environment/Normal) described above.

The geometric properties of a cube map are the same as those defined by the cube geometry object.

Cylinder Mapping

A cylinder mapping object can be used for wrapping a material around an object cylindrically. It uses a cylinder geometry object to define the material space.



Checkered stripes mapped to a tube using cylinder mapping

Default Mapping

A default map doesn't define any geometric properties. Instead, it uses the following five methods for defining the material space coordinates:

- Surface - The target surface defines the material space. This is a useful method when creating waving flags and skin textures, where the texture must stretch with the surface.



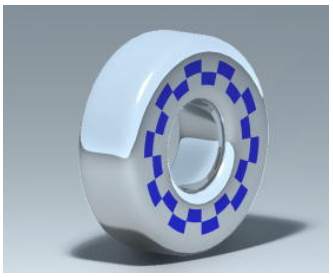
Note

Some surfaces do not define valid UV coordinates along the surface.

- Identity - This method uses the rendering space coordinates. Because the rendering space is tied to the camera, this alternative is suitable only for still images.
- Null - No material space computations at all. The material space remains as possibly defined by the previous material mapping objects. A quick mapping method for materials that do not contain any patterns, such as glass.
- Camera - Uses the current camera orientation to project material properties to target objects. A point under the top left corner of the image has coordinates (0,0,distance), the bottom right corner (1,1,distance).
- Custom - Moves a selected channel to the 'Map coords' channel. This is a suitable mapping type when you want to parametrize a material with a user channel that has been added as a 'vertex variable' to a SDS or NURBS object.

Disk Mapping

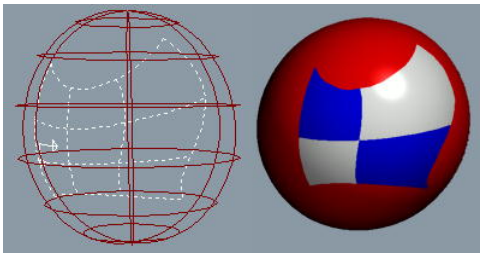
The disk mapping object uses an ellipse object to define the geometric mapping properties. It can be used, for example, to attach a 'Good Year' logo to the tires of a formula one car.



A checkered pattern disk mapped to a chrome wheel

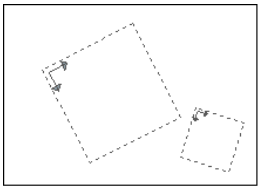
Mesh Mapping

A mesh mapping is a NURBS mesh that 'lives' on another NURBS mesh. In other words, the control points of the mesh map are UV values in the parameter space of the parent mesh. A mesh map can be created from the pull down menu 'Tools/Unified/Rectangular Object' and by setting the 'Class' option to 'MeshMap'.



Parallel Mapping

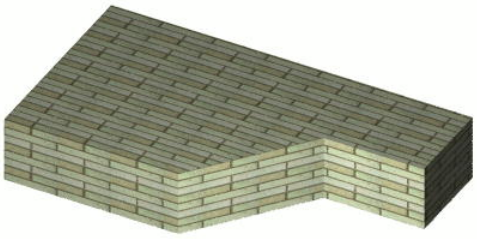
A parallel mapping object uses a cube geometry object to define the material space.





Two logo textures attached to a rectangle using a parallel mapping. Ray traced image on the right.

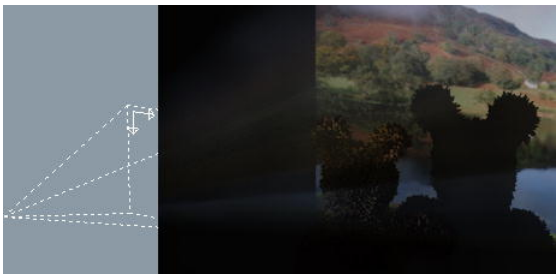
Parallel mapping has one special option: the **Roll** feature. The roll option rotates the Z axis of the mapping perpendicular to the surface normal, and other axes respectively. The option is most useful when texturing objects that consist of several planar surfaces. For example, one can map one brick pattern to one wall and then spread the pattern over all walls by using the roll option.



One single parallel mapping roll-maps all sides of a polyhedron

Pyramid Mapping

The Pyramid mapping object maps materials to geometric objects using a perspective projection. The attributes of a pyramid map object are the same as those defined by the pyramid geometry object. This mapping method is useful for creating slide projector effects.



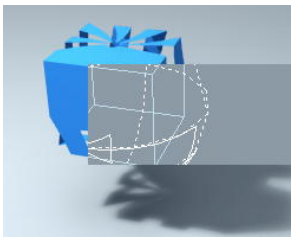
A slide projector created by pyramid mapping an illumination texture to a light source

Sphere Mapping

A sphere mapping object wraps a material spherically. It uses a ellipsoid geometry to define the material space. This mapping method is useful, for example, for defining environment maps.

A sphere map has one object specific option:

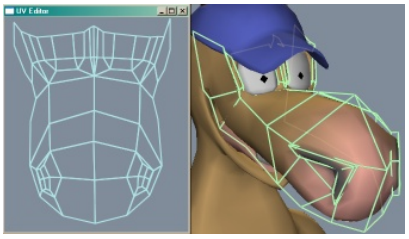
Environment Map: The mapping coordinates are computed from the reflected ray direction, not from the point position as usual. This mapping technique can be used for simulating reflective surfaces.



Spherical Clip Mapping

UV Set

UV Set is a powerful object for defining UV coordinates and material attributes for subsets of SDS objects. The power stems from the fact that the UV set object is derived from the SDS object itself. Many of the tools available for editing SDS objects are now also available for UV editing.



UV Set for texturing the upper half of a character's face

Creating UV Set objects

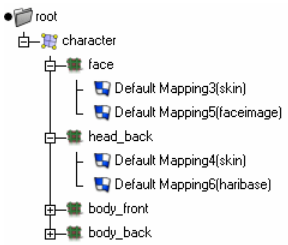
There are two ways of creating UV set objects:

1. Select an SDS object in the object edit mode and apply the *UV Set* tool with the *Cubic UV Set* option. This creates six UV-set objects representing front, back, up, down etc. directions.

2. Switch the SDS object into the edit mode, select faces and click the *UV Set* tool with the *Selected faces* option. This creates a single UV set of the selected faces. In this case, the UV coordinates for UV set vertices are fetched from the UV coordinates channel of the SDS object.

Of course, you can use edit/copy-paste and duplicate tools to create new UV sets from existing UV set objects.

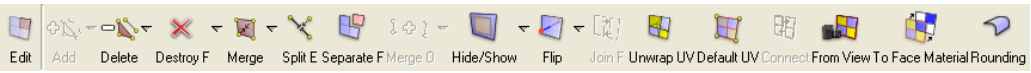
UV set objects are created as child objects to the SDS object.



Surface division into UV areas and their texturing is managed efficiently using the select window

Any number of material mapping objects (parallel, spherical etc.) can be applied to an UV set objects to texture the actual SDS model. UV set also supports the direct texture definition in the property window's Col tab, as well as vertex channels, which now can be assigned memory efficiently only to sub areas of the SDS mesh.

Because the UV set object is SDS based, many SDS specific modeling tools can be used for editing. You can delete faces from the UV set, add new faces to it, join faces, unwrap and so on. All these tools are instantly available in the control bar as soon as you select an UV set.



UV editing tool set

In addition to the standard SDS editing tool set, there are also some UV set specific tools:

UV Set to Face Material

Converts selected UV sets to face materials of the parent SDS object. UV Set object's child mapping objects define the face material names. The UV values for the face materials are taken from the UV Sets. With this tool, you can convert the hierarchical UV set based texturing into a simpler form for exporting purposes.

Connect

This tool can be used for connecting two separate topological face groups into one.

From View

This tool takes the current view projection and uses it to set the UV coordinates of the selected faces of the UV set. Select some faces, navigate on the view window and click the tool. Simple and quick.

Light Sources

Realsoft 3D supports a rich set of light sources. The light source attributes described below include quite extensive controls. Even more lighting power can be achieved by using materials that define custom lighting effects using a Light properties shader.

The controls that are available for most light source types are listed below.

Intensity: Controls the brightness of the light source. The intensity value of a light source is combined with its RGB color value. For example:

intensity=0.5, color=(0.8, 0.7, 0.6)->light color=(0.4, 0.35, 0.3).

Volumetric: If set, the light source is considered in volumetric lighting. Fog materials that have diffuse color become illuminated by this light source.

Diffuse Only: If set, the light source creates only the default diffuse shading i.e. the illumination is proportional to the angle at which light hits the surface. Specular lighting and other VSL defined shading effects will be ignored.

Falloff Frame

These controls define how the illumination fades as the distance from the light source increases.

- *None* - Illumination does not depend on the distance.

- *Local* - Illumination fades to zero within the falloff radius.
- *Distance* - Illumination is inversely proportional to the distance. Light intensity decreases to 50% when the distance is doubled.
- *Distance squared* - A physically correct falloff for point light sources. Intensity decreases rapidly by distance.

Radius: The falloff distance. Local falloff drops to zero at this distance, 'Distance' and 'Distance squared' falloffs decrease to 50 % at this distance.

Shadow Casting Frame

Shadow casting properties are available for all light source types. Point and spot lights support shadows mapping.

No shadows: The light source does not cast any shadows. This kind of light source renders quickly.

Ray traced shadows: Shadows of the light source are computed using ray tracing. Such shadows are realistic and obey material properties (glass object do not cast black shadows etc.).

Mapped shadows: Shadows are first rendered to a bitmap which is then filtered in a suitable way. This kind of shadows have smooth edges and often render quickly. However, mapped shadows are more inaccurate than ray traced shadows and do not support shadow translucency.

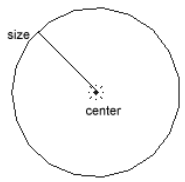
Resolution: The accuracy of the shadow map. Both x and y dimensions use the same pixel resolution. A low resolution, for example 200 pixels, will make shadow computations quick but inaccurate. A higher resolution, for example 1000 pixels, is accurate, but shadow map rendering takes much longer time.

Static: If set, shadow maps are not deleted after rendering. Consequent rendering steps can then use the existing shadow maps. This speeds up rendering. The result is correct if the scene does not change near the light source in question (e.g. when only the camera is animated).

Point Light

A point light source is defined by a center and a radius. A point light illuminates light to all directions. The radius attribute can be used for controlling the fall off.

It is also possible to set a 'size' attribute, in which case the point light becomes an area light generating soft shadows in a physically correct way.



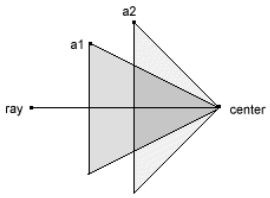
Point light also supports an 'Ambient' option. If set, the light source no longer emits light rays radially but randomly to all directions. The position and size attributes are then used for controlling the strength of illumination. This allows the user to create local ambient lighting effects.

Point light specific options:

- *Point/Ambient light* - The Ambient option turns the point light into an ambient light.
- *Position* - The position of the light
- *Size* - The radius of the light source. Size is used when computing mapped shadows. It is also used for ray traced smooth shadows i.e. when 'Quality' is greater than 0. The higher the size, the smoother the shadows become.
- *Quality* - The sampling quality for ray traced smooth shadows. A high quality value increases rendering time but produces smooth shadows, whereas low values render quickly but may introduce some dithering artifacts at shadows edges. The bigger the light source size, the higher quality value is needed for a proper quality.

Spot Light

A spot light is defined by two light cones. The intensity of the light drops gradually to zero between the inner and outer angles. A spot light source is often quicker to render than a point light or a distant light because the illuminated area is more compact.

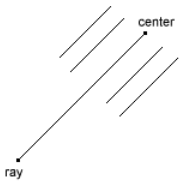


Spot light specific options:

- *Position* - The position of the spotlight
- *Aimpoint* - The point at which the spot is aimed
- *Angle1, Angle2:* - The angles that define the spot beam. The greater one of the two angles defines where spot illumination ends. Illumination fades from the full intensity to zero between the two angles. If the angles are equal, the spot light beam has a sharp edge.
- *Material* - This control includes a list of the fog like materials of the current material library. Selecting a material will render a fog cone that matches the spot beam. This provides an easy way to make the spot beam visible.
- *Size* - The radius of the light source. Size is used when computing mapped shadows. It is also used for ray traced smooth shadows i.e. when 'Quality' is greater than 0. The higher the size, the smoother the shadows become.
- *Quality* - The sampling quality for ray traced smooth shadows. A high quality value increases rendering time but creates smooth shadows, whereas lower values render quickly but may introduce some dithering artifacts at shadows edges. The bigger the light source size, the higher quality value is needed for a proper quality.

Distant Light

A distant light is defined by two points: center and ray. These two points define the direction in which the light source illuminates. All light rays are parallel to each other. A distant light can be used for simulating distant light sources, such as the sun. At rendering time, the program moves a distant light source internally infinitely far away from the target. Therefore, no light fall off settings are available.



Special Light

A special light source is a level object which doesn't define any geometric data. However, it allows the user to use any geometric object as a source of light; all sub objects radiate light. If the sub object is a single point, a point light is achieved. By using a NURBS curve as a sub object, one can create one dimensional area lights. Using a rectangle creates a two dimensional area light. The colors of sub objects of a special light sources are taken into account, including texture mapped colors in a 'Light properties' shader.

Special light specific options:

- *Quality* - The illumination sampling quality. Defines how densely the sub geometry surfaces are evaluated for estimating the total illumination. A high quality value increases rendering time but produces smooth shadows, whereas low values render quickly but with dithered shadows. The bigger the sub object are, the higher quality value is needed for a proper quality.

Ambient Light

An ambient light is actually a point light with the 'Ambient' option set. The point geometry can be used for falloff purposes; for example, you can add a local ambient light source in front of a TV screen to simulate the local illumination it creates, or a local negative ambient light source in a dark corner of a room scene.

Usually ambient light sources work best if the Diffuse option is set and shadows are turned off. Both options are allowed for special effects purposes.

photon map

photon map stores indirect illumination by tracing photons and storing their surface hit positions with the carried illumination into a point cloud. Illumination in a surface point is then computed by finding and averaging the closest photons. photon map has the controls described below. The frequently needed controls are available on the toolbar, and the rest can be found from the property window.

- *Photon Count:* - The number of photon-surface hit points stored in the map
- *Brightness:* - The brightness of the photon map illumination
- *Sampling:* - The number of closest photons used for evaluating illumination in the examined point. Increasing the value reduces noise, but makes the illumination less detailed and also slower to render. A typical Sampling value for a high quality photon map is 100-1000.
- *Radius:* - Limits the search radius when the closest photon set defined by the Sampling value is collected. The program automatically maintains a reasonable radius value (it is updated when the Sampling rate changes). The user can manually adjust the automatic value. Decreasing the value makes rendering faster, but may increase noise in areas where photon density is low. Increasing the Radius value may sometimes fix rendering errors visible in a photon map.
- *Sharpen Shadows:* - A filtering option, which prevents incorrect illumination spreading from bright areas to strongly shadowed areas. The option is not available for pre-filtered photon maps.



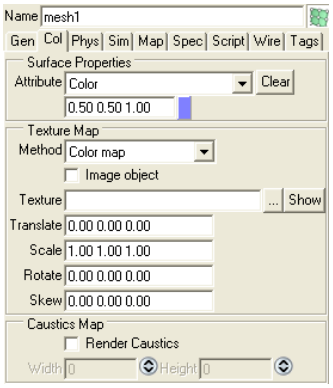
photon map illuminated interior scene. Courtesy of Pixel Perfect.

Filter Tool

Illumination of a photon map can be preprocessed using the Filter tool. This tool evaluates the illumination defined by the photon map at the actual photon positions. This is done by averaging data typically from hundreds of nearby photons, which is a time consuming process. After pre-filtering, illumination can be resolved from a small number of closest photons. Therefore, pre-filtering tool is a great time saver for camera animations in static scenes, and also for single images if the number of photons is clearly smaller than the number of image pixels.

Common Properties of Geometric Objects

Color Tab



The Col tab contains color and other surface property related settings. These controls provide a simple and easy way to modify surface appearance (shading) of objects. When more control is needed, VSL materials provide a more powerful way to define shading properties.

Surface attributes defined in the Col tab are evaluated before the materials assigned to the object. This has two consequences:

- Surface attributes are 'weaker' than material assigned attributes. A color defined by a VSL material overrules a color defined here.
- Materials can utilize Col tab attributes. For example, a glass material can derive the transparency from the usual object color attribute. Then you can easily create different colored glass objects by using the same material and changing the object color.

The Surface Properties Frame

Attribute: This gadget contains a list of ordinary surface attributes. Color is the most commonly needed attribute, but you can also define Illumination (=self illuminating color), Reflection, Transparency, Fade, Alpha etc. properties here. For more information about channels, check the Channels documentation.

Clear: Removes the value for the attribute defined by the previous field from the object. The object will then use a value possibly defined by a parent level, or if the parent value is not set, the default value of the attribute.

Attribute Value: This field controls the value of the selected attribute. A floating point, vector or color gadget is displayed depending on the attribute type.

The Texture Map Frame

Method: Selects the type of the texture mapping. A color map defines the usual diffuse color. 'Environment map' makes the object mirror like (black diffuse color, full reflectivity) and uses the defined texture file as a spherical reflection map for the surrounding world.

Texture: The name of the image file for texturing.

Translate, Scale, Rotate, Skew: These four gadgets define a texture transformation matrix. For example, the texture can be tiled and repeated several times by decreasing the two first components of the Scale vector.

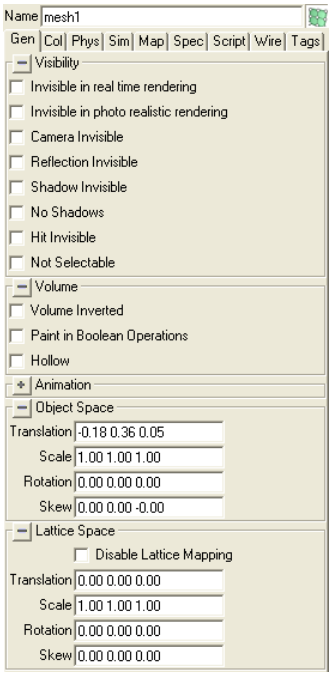
The Caustics Map Frame

Render Caustics: Enables caustics rendering for the object i.e. makes the object receive caustics illumination. Note: the top level caustics switch in the render settings must be also enabled.

Width, Height: The accuracy of the caustics effect for this object. Caustics illumination is stored to a object specific illumination map, and these values define the resolution of the map. The higher the resolution, the finer details caustics illumination contains. However, high resolution usually requires that the caustics Sampling value is high enough - otherwise the illumination details may become too weak or noisy. A good thumb rule is that Width and Height are about the sampling value multiplied by ten. For example, the default sampling rate 20 is enough for a 200*200 caustics map.

General Properties

The General Properties tab includes some frequently needed properties of geometric objects.



Name: The name of the selected object(s).

Class Icon: At the right side of the name gadget, a small icon shows the class of the selected object. The associated tooltip tells the name of the object class (rectangle, cube etc.).

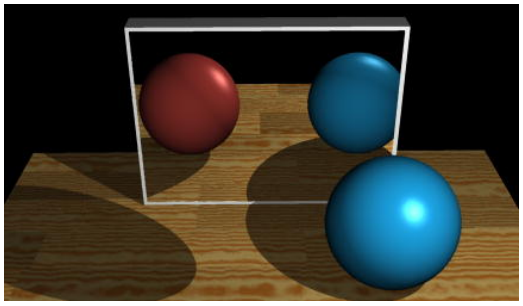
The Visibility Frame

Miscellaneous visibility options.

Invisible in real time rendering: Hides the object (and its sub objects) from real time view rendering (both GDI and OpenGL).

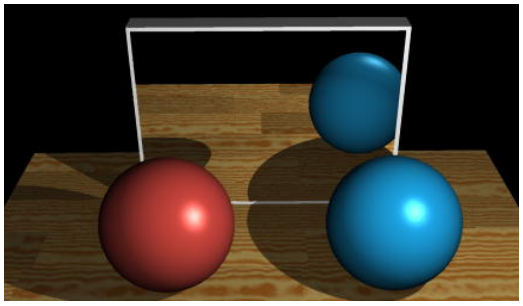
Invisible in photo realistic rendering: Removes the object from ray tracing and other 'final' rendering systems.

Camera Invisible: Makes object invisible for the ray tracing camera. Invisibility covers only the direct view from the camera: the object remains visible via reflections, casts shadows etc.



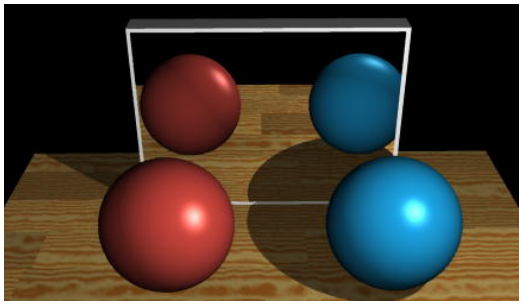
A camera invisible red sphere in front of a mirror

Reflection Invisible: Object will not show in reflections or trough transparent objects.



The red sphere is now reflection invisible

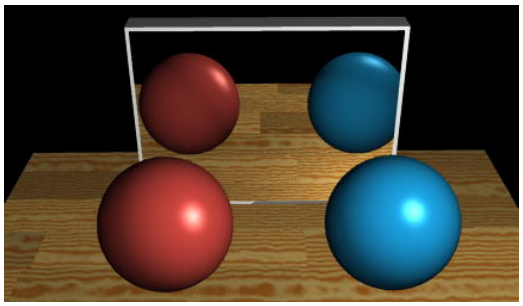
Shadow Invisible: If set, the object will not cast any shadows. This can be used as a rendering optimization trick. For example, room walls can be made shadow invisible if the camera and all light sources are inside the room.



The red sphere made shadow invisible

No Shadows: If set, the object will not receive any shadows cast by other objects or the object itself. Light sources do illuminate the object; only the shadows are eliminated.

Hit Invisible: If set, the object will be ignored in collision detection, 3D painting and in the surface projection of various tools such as the particle brush tool and the paste tool. For example, you can make a water surface hit invisible before brushing some rock particles into the bottom of a lake.



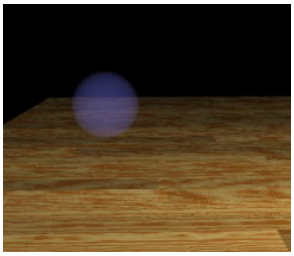
No Shadows set for the floor

Not Selectable: If set, the object (and possible sub objects) cannot be selected by clicking or dragging the object wireframe on a view window. The object can still be selected from the select window. The option is useful when editing complex scenes with lots of wireframe.

The Volume Frame

Volumetric object properties for Boolean operations and volumetric rendering effects.

Volume Inverted: The object fills the exterior of its natural volume with matter. For example, a tiny sphere becomes an infinite space with a small bubble in it.

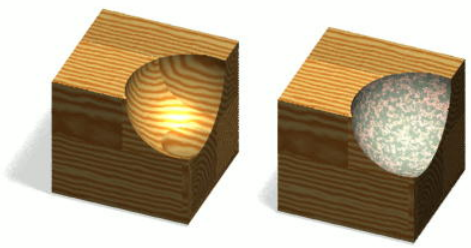


Fog bubble



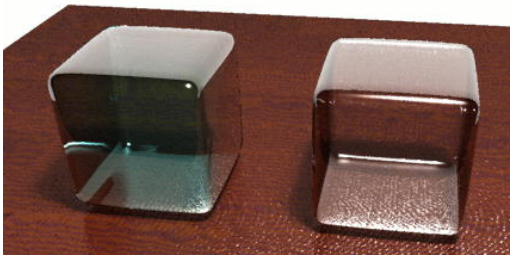
Inverted fog bubble

Paint in Boolean Operations: If set, the object keeps its original surface attributes when cutting another object. By default, the first sub object of a Boolean AND operation level defines the properties of the result.



Wooden cubes cut with a marble sphere. The marble cutter at the right has the 'Paint' option enabled.

Hollow: Makes an object hollow i.e. an infinitely thin surface without a volume. This shows up when Boolean cutting the object or if the object is made of transparent, refractive matter.



Two glass cubes made of the same material. The one on the right is hollow

The Animation Frame

Object behavior in animations.

Rotate in HPB: By default, object rotations are animated using so called Quaternions, which produce rather intuitive smooth rotations. This option selects a more machine like rotation interpolation in the Heading-Pitching-Banking angle space.

Animate only when selected: Objects can be animated by modifying them animation recording on. Sometimes objects change (because of a lattice map connection etc.) even when they are not selected. This option helps preventing unintended animation event recording for unselected objects.

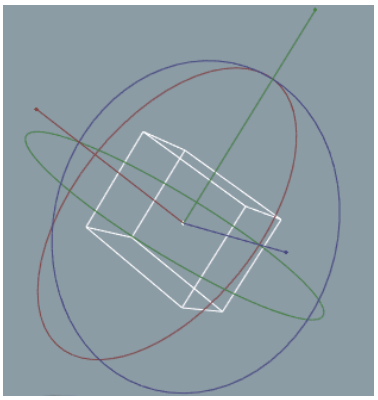
Animation Protected: If set, the object will not record any key frames when modified in animation recording mode. This helps to avoid unintended key frames.

Key frame animation only: This option is similar to the previous one, with the exception that already animated attributes do record new key frames. Unanimated attributes remain unanimated.

Hide Choreographs: Hides all choreographs associated with this object and its sub objects. This option is useful when managing scenes consisting of thousands of animated objects and you don't want to see them all in the Choreography Window.

The Object Space Frame

Object space transformation components. You can see the object space position and orientation on view windows trough the coordinate handle which appears when you select the object. The X axis is drawn in red color, the Y axis in green color and the Z axis in blue color. You can remember the color order easily by checking the view window orientation guides - each axis is labeled there.



Translation: The position of the object space origin (pivot point).

Scale: The total scaling of the object (measured along object space axes).

Rotation: The total rotation of the object. The three values are the heading, pitching and banking angles which rotate the world space XYZ axis system to the current object space axis system as follows:

- Heading angle first rotates the world X and Z axes around the world Y axis.
- Pitching angle then rotates the obtained Y and Z axes around the rotated X axis.

- Banking angle finally rotates the obtained X and Y axes around the rotated Z axis.

Skew: The shearing angles of the object space. If all angles are 0, the object space is orthogonal.

The Lattice Space Frame

The transformation components of a lattice mapped object. A lattice mapped object has an additional transformation matrix which specifies the object orientation in the lattice space. For example, making a curve mapped object to follow the path is simply a matter of animating the lattice space translate x component to increase from zero to one.

Disable Lattice Mapping: If set, modifying a lattice object to which the object is mapped no longer affects the objet.

Translation: Object position in a lattice space. For example, if you place the pivot point of an object to the beginning of a path curve and then map it, the lattice translation will be (0,0,0). Lattice mapping an object to the middle of a mesh will set the translation to (0.5, 0.5, 0).

Scale: Object scale in lattice space.

Rotation: Object rotation in lattice space.

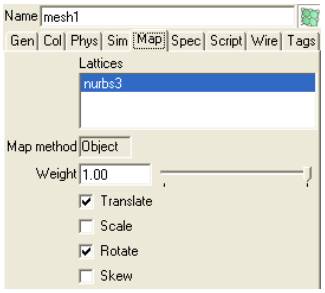
Skew: Object shear in lattice space.

Mapping Tab

The Map tab shows the lattice mapping summary for the object.

Lattices: The list of objects to which the edited object is lattice mapped to. You can click a list item to view the properties of the corresponding lattice binding at the bottom of the property window. The list also includes a popup menu 'Remove' for deleting undesired lattice connections one by one.

Map method: The type of the lattice mapping, Pointwise or Object. The pointwise mapping means that each geometry point is treated individually in lattice mapping. Object mapping controls the object via its object space that is attached to the lattice.



Lattice mapping properties of an object following a directed path

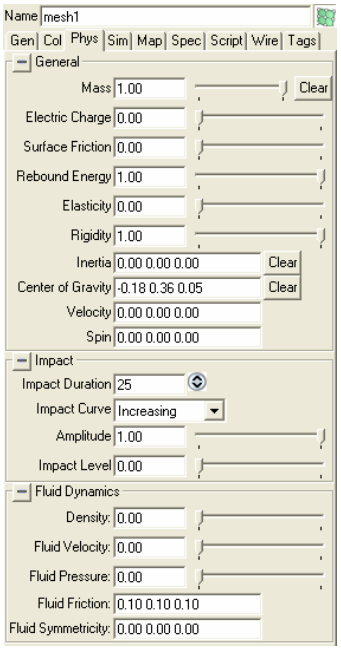
Weight: Defines how strongly the lattice mapping affects the target. Weight one means that the control is total i.e. the mapped object fully obeys the lattice. A weight zero disconnects the object fully from the lattice. When using several lattices, a weighted average is computed. Note that in pointwise mapping, also the weight is pointwise. To control pointwise weights, put the mapped object to the point edit state, select some points and enter the desired weight.

Translate, Scale, Rotate, Skew: The transformation components of an object space type lattice mapping. If only Translation is set, the lattice controls only the object space position of the mapped object. If also Rotate is set, the lattice controls the orientation of the target. The Scale component has importance when the lattice is two or three dimensional. For example, if an object follows a mesh and the scale option is enabled, the object will shrink together with the mesh. Finally, the Skew component allows a non orthogonal lattice control making the object to shear together with the lattice.

Physics Tab

The 'Phys' tab includes controls for physical object properties which are used in simulations.

General Properties



Mass: The mass of an object (in kilograms). The default mass is 1 kg. A level object's default mass is the sum of the mass of sub objects. Any other mass can be defined here.

Clear Mass: Clears a custom mass definition so that the default mass computation will be used.

Electric Charge: The amount of electric charge (coulomb). Can be positive or negative.

Surface Friction: A surface friction coefficient 0..1.

Rebound Energy: Describes how well the object keeps its kinetic energy in collisions. The value range is 0 .. 1. If the value is 1, the object keeps its kinetic energy and just changes the direction of its motion in a collision.

Elasticity: Objects ability to maintain its shape in a deforming situation such as a collision. If elasticity is zero, there is no force that restores the shape after a deformation.

Rigidity: Describes how rigid the object shape is. The lower the value, the more freely object matter can react to forces and the less internal friction resisting the shape deformations exists. If the rigidity is zero, object geometry points act independently in simulations (elasticity defines a string force that maintains the shape).

Inertia: Describes how the mass is distributed in object volume. If the mass is concentrated in the middle of the object, the inertia is small and the object has a tendency to spin easily in collisions. If the mass is concentrated at outermost edges (say a hollow lead sphere), the object has a large resisting momentum and does not spin easily. The program computes a default estimate from the object mass and bounding box dimensions. Be careful when changing the default momentum; entering a value that is physically impossible will invalidate the collision impact calculations.

Clear Inertia: Removes a non default inertia definition. The value estimated by the program will be used.

Center of Gravity: The program estimates the center of gravity as an average of the surface geometry, but you can change it here if necessary.

Clear COG: Removes a non default COG definition. The value estimated by the program will be used.

Velocity: Defines the direction and speed of object's motion.

Spin: Defines how the object rotates around its object space axes. The unit is the selected angle unit per second. For example, (0, 180, 360) degrees per second.

Impact Properties

The impact system maintains an object attribute Impact Level, which can be used for key driven animations launched by collisions. For example, you can key frame a collision behavior and bind it to the Impact Level attribute.

Impact Duration: The duration of impact wave (measured in the selected time units - frames, seconds etc.). The impact wave runs linearly from zero to Amplitude during the duration time.

Amplitude: The maximal value of the Impact Level attribute during the impact wave processing.

Impact Level: The current value of Impact Level. You can use this control to test a choreography driven by the Impact Level attribute.

Fluid Dynamics Properties

Density: Describes how dense a matter where simulation happens is. A typical value range is 1..10. The denser the matter, the more it resists motions (or creates them in case the matter has a moving flow such as wind). This property is attached to objects that define a fluid such as the fan object.

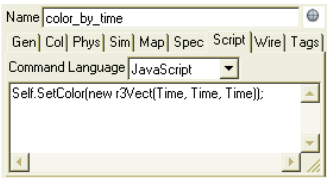
Fluid Velocity: The speed of fluid flow as meters per second (the flow direction is defined by the fluid creator geometry, for example a fan axis). Also this property is attached to objects that define a fluid flow such as the fan object.

Fluid Pressure: The pressure of fluid. This value affects e.g. forces generated by a non symmetric fluid flow.

Fluid Friction: Object's friction coefficients in object space directions. Many objects are strongly non symmetric in this respect: for example, an airplane is designed to have a small friction in its natural peak-tail direction, but it has quite a strong friction against the wing planes. This property controls objects moving in fluid, not the fluid creator.

Fluid Symmetricity: Fluid flow symmetry with respect to object space axes. The value zero means symmetric flow. Non symmetric flows generate a pressure difference at the opposite sides and hence a force along the axis in question (for example: an airplane wing).

Scripting Tab



It is possible to control object behavior in animations using scripts. The Script tab contains an editor for this purpose. Scripts entered through the Script tab are executed last in the construction stack so they can be used for controlling other animation effects easily.

The **Command Language** gadget shows all installed script languages. With it, you can choose which language to use for the object.

You can enter the actual script to the text editing area below the language selector. For example, the following JavaScript program:

```
Self.SetCenter(new r3Vect(0, 0, 0));
```

sets the center point to zero. The variable 'Self' refers to the object the script is associated with and SetCenter() is a method defined by the class in question.

Consult the Realsoft 3D JavaScript referece manual for more information about Realsoft 3D classes and the methods they define.

The following script controls the size of the sphere using the color attribute.

```
color = Self.GetColor();
Self.SetTotalTranslate(color);
```

To make sure the object don't go below 'y=0' level:

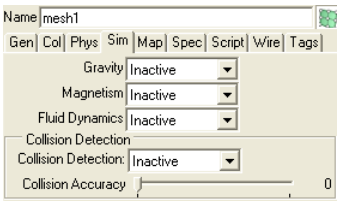
```
t = Self.GetTotalTranslate();
if(t.y < 0) {
    t.y = 0;
    Self.SetTotalTranslate(t);
}
```

'Time' variable holds the current time allowing you to create animations easily. For example, to control the object's color in function of time:

```
Self.SetColor(new r3Vect(Time, Time, Time));
```

Simulation Tab

The 'Sim' tab includes switches for turning various simulation calculations components on and off.



Gravity: Enables Gravity force for the selected objects. There are 4 options available:

- *Inactive* - The object is not considered in gravity computations.
- *Both* - The object creates a gravity field and is pulled by gravity forces.
- *Cause* - The object generates a gravity field. It does not react to gravity forces.
- *Affected* - The object does not create gravity, but is pulled by gravity forces.

Magnetism: Enables Electro Magnetic simulation. Magnetism is different from gravity in that it is a signed quantity: similar charges repel each other, opposite charges attract each other. The four magnetism alternatives - Inactive, Both, Cause, Affected - work the same way as in the Gravity setting above.

Fluid Dynamics: Enables fluid simulation. See 'Gravity' above for the option types. Typically only fluid specialized objects such as a fan have the 'Cause' option set i.e. they generate a fluid environment. Other simulated objects use 'Fluid Dynamics = Affected' option.

Collision Properties

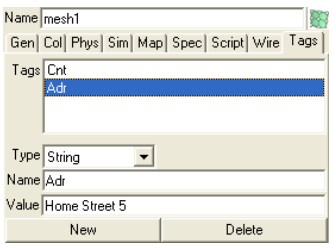
Collision Detection: Enables Collision detection as follows:

- *Inactive* - The object does not collide or make other objects collide.
- *Both* - The object creates collisions and collides itself.
- *Cause* - The object creates a collision impact when hitting other objects. However, it does not react itself to the collision.
- *Affected* - The object reacts to the collision impact. However, it does not change the motion of other objects in a collision.

Collision Accuracy: Defines how accurately the collision point is examined. The higher the value, the sooner collision (= object volume penetrating another object volume) is detected. High values naturally increase computation time. If the object shape is complex (includes sharp extrusions etc.), a high collision accuracy may be necessary to prevent object volume overlapping.

Tags Tab

You can attach additional information to objects using tags. Tag information is visible to the animation system, so you can key frame your custom tags or use them to parametrize animations.



The **Tags** gadget shows all user defined tags of the object. Clicking a tag item activates it showing the tag properties at the bottom of the window.

The **Type** gadget sets the data type of the selected tag. All standard data types such as integer, floating point, color, vector, string etc. are available.

The **Name** field sets the tag identifier. A tag identifier is a three digit character sequence. New tags use a dummy identifier 'XXX', which can be renamed as appropriate.

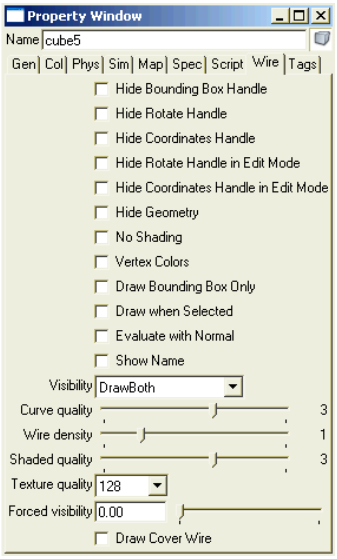
The **Value** field controls the tag value. The gadget adapts automatically to the selected data type.

The **New** button adds a new tag item to the tag list.

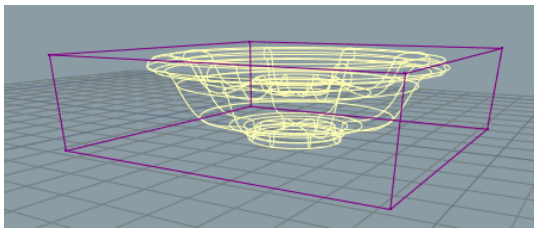
The **Delete** button removes the selected tag item from the tag list.

Wire Tab

The Wire tab contains object settings that are related to real time drawing, both in GDI wireframe and using shaded/unshaded OpenGL. Most of these settings are common to all objects. Some object types insert additional controls to the bottom of the window; check the object specific documentation for more information.



Hide Bounding Box: Hides the bounding box edit handle of this object.



Bounding box handle drawn with a purple color



Note

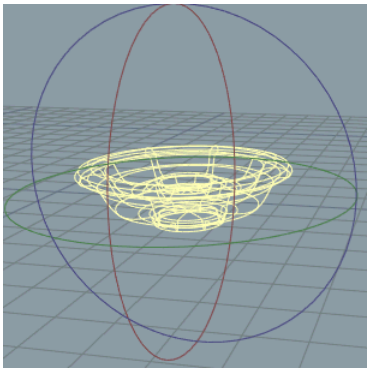
Bounding box handles do not show unless their drawing is enabled from the view popup menu.

Hide Rotate Handle: Hides the object space rotate handle of this object.



Note

Rotate handles do not show unless their drawing is enabled from the view popup menu.



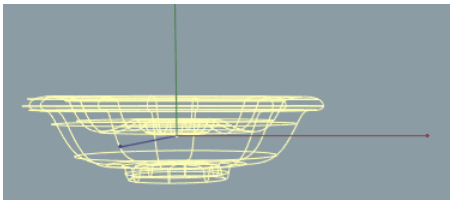
Rotate handle shown, other handles hidden

Hide Coordinates Handle: Hides the object space scale and translate handle of this object.



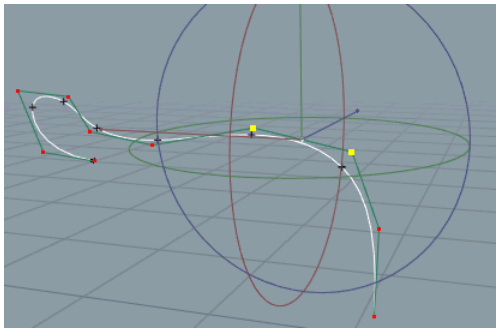
Note

This handle type does not show unless the rotate handle drawing is enabled from the view popup menu.



The Coordinates Handle

Hide Rotate Handle in Edit Mode: Hides the rotation handle of this object in point editing mode. Some objects, such as the Subdivision Surface, display so many other handles that hiding the default handle may be necessary to avoid excessive handle wireframe. With the rotation handle, as well as the Coordinates Handle described below, you can edit the selected points in object's space. The handles are positioned in the average point of the selected points. Rotate handle drawing must be enabled from the view popup menu, otherwise the handles won't be drawn.



Point editing a NURBS curve. The rotate and coordinate handles are enabled.

Hide Coordinates Handle in Edit Mode: Hides the scale/translate handle of this object in point editing mode. You can edit the selected points in the object space with this handle. Rotate handle drawing must be enabled from the view popup menu, otherwise the view window will not draw the handles.

Hide Geometry: Makes the actual object geometry (surface, curve, etc.) invisible in real time drawing. Handles and possible sub objects will be drawn normally.

No Shading: Forces wireframe drawing also in shaded OpenGL drawing.

Vertex Colors: If set, colors from materials are evaluated per mesh vertex. This overrules Texture Quality setting - no texture bitmaps are used. This is sometimes the most suitable and quickest way to roughly show the surface colors.

Draw Bounding Box Only: Hides the actual surface and shows only the bounding box of the object. Sometimes it is enough to see only the objects under editing using accurate geometric representation. Other objects can be drawn using simple reference boxes. Using a bounding box instead of accurate wireframe or shaded surface naturally speeds up model drawing and reduces visual complexity of a detailed scene, thus making editing the scene easier.

Draw when Selected: If set, object wireframe becomes visible only when the object is selected directly (not as a part of selected parent level). This is useful for example for material mapping objects, whose wireframe does not clutter the scene, but which become visible when it is time to adjust the size or proportions of the mapping.

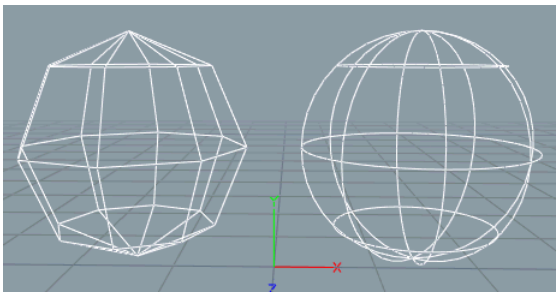
Evaluate with Normal: When set, shader conversion to OpenGL textures computes surface normal. This gives more accurate result when shaders (or material maps such as the roll mapping) depend on the normal.

Show Name: If set, the object name is displayed on view windows.

Draw Only Profile: If set, only wireframe of sharp edges and object boundary is drawn. The option works best with dense wireframes.

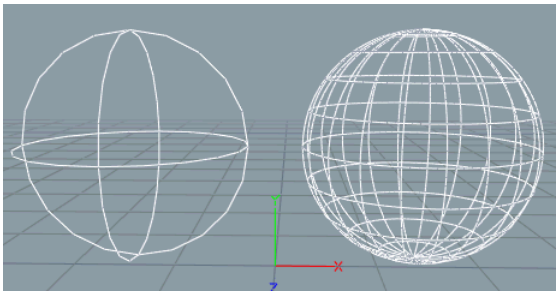
Visibility: Selective visibility in shaded/unshaded drawing. For example, the standard shaded mesh along the object's surface can be used for OpenGL drawing and another simplified 'symbolic' wireframe for GDI drawing.

Curve quality: The density of wireframe curve subdivision. Higher values display curves more accurately, but increased density makes drawing slower.



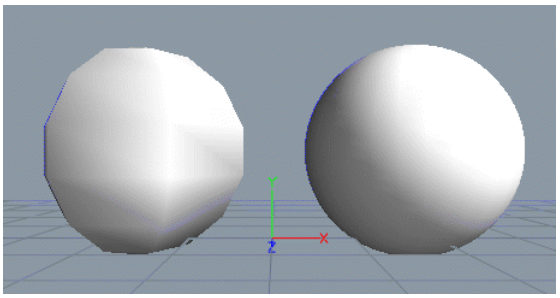
Two identical spheres. The sphere on the right is drawn using high curve quality.

Wire density: The density of wireframe mesh i.e. the density of wireframe curves drawn on surfaces. This setting has no effect on 1D curves.



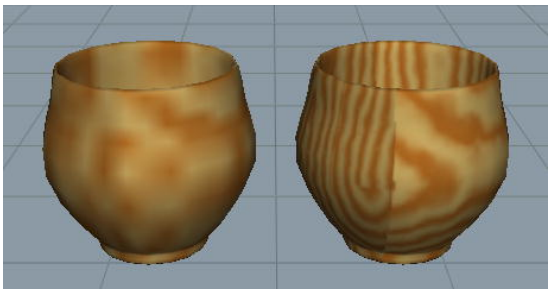
Two identical spheres. The sphere on the right is drawn using high wire density.

Shaded quality: The surface drawing quality in shaded OpenGL. Increasing the value makes surface tessellation to polygons denser, so that the surface appears smoother.



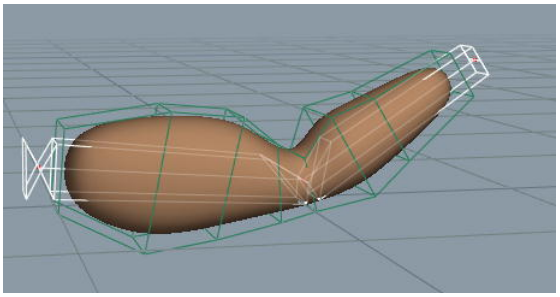
Two identical spheres. The sphere on the right is drawn using high shaded quality.

Texture quality: The quality of material evaluation for shaded OpenGL. The material system of Realsoft 3D supports blending of multiple (mathematically defined or image mapped) materials. The OpenGL drawing system can evaluate the result as a single texture map that colors the surface. The values of this control describe the pixel resolution of the result texture. Evaluation of complex 'final rendering' oriented shaders can be time consuming, so be careful when applying high quality values. Note: 'direct' textures assigned in the 'Col' tab can be drawn very quickly and therefore such textures are drawn in the original resolution (regardless of the texture quality setting in the wire tab).



Two cups with a procedural wood material. The left one has a texture quality 32, the right one 256.

Forced visibility: When using shaded OpenGL drawing, it is often important to see certain items even when they are behind surfaces. For example, object editing handles are always somewhat visible through obscuring surfaces. Sometimes complete objects must be seen as well: a skeleton is placed inside a character object. Its visibility is important when editing the character pose. A suitable forced visibility value makes the skeleton always visible.



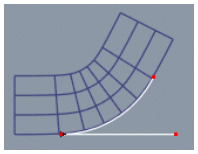
A skeleton forced to show through the flesh

Constructors

Bend Object

Bend deformator is a constructor based deformator. It can be used for generating circular bend effects to free form objects.

The tool asks the user to define an axis with specified length. This axis can then be bent and all the target points will be deformed accordingly.



A bend deformator applied to a nurbs mesh

To bend a nurbs mesh:

1. Create a nurbs mesh consisting of 10 x 6 control points

2. Select the mesh and activate the bend tool. Then define bending axis along one of the edges of the nurbs mesh.

3. Switch the created bend object to point edit mode and drag the end point handle to bend the axis. The nurbs mesh is bent with the axis.

Bending radius can be specified numerically through the property window.

Displacer

Point Displacer deforms geometric objects using material mapping objects.

The displacement effect is defined by one or several material mapping objects, which are sub objects for the displacer.



A displaced mesh consists of a mesh and a displacer object

When using multiple materials for displacement, the selected materials are responsible for combining the computed channel values. A working combination could be:

```
material1                              material2
   surface properties                     surface properties
      color=*noise(map coords)               color=*curve(map coords)
```

The shaders below will not combine well (only the latter material mapping counts):

```
material1                              material2
   surface properties                     surface properties
      color=noise(map coords)                color=curve(map coords)
```

Point Displacer has the following controls:

Channel: Selects the material channel, which defines the displacement.

Space: Orientation for moving the target points.

- *Absolute* - The world space defines displacement direction. If the 1D option is set (or the selected channel is one dimensional), displacement happens along the Z direction of the world space. Otherwise the first sub channel component defines X displacement, second sub channel Y displacement etc.
- *Object* - The object space of the point displacer itself (not of the target mesh!) defines the orientation for displacement. For example, if 1D option is set, the target mesh is displaced along the blue Z axis of the displacer space.
- *UVW Space* - The surface orientation of the original non-displaced mesh defines the displacement direction. In 1D case, the target surface is displaced perpendicularly i.e. along its normal.

1D: - If set, the target mesh is displaced only in Z axis/Surface normal direction. If the selected channel is 3 dimensional, the sub channel average defines applied 1D displacement values. 1D displacement is usually the most intuitive and easiest deformation available via this constructor.

Carriage Object

Carriage object is defined by a coordinate system. The coordinate system defines a linear transformation that is applied to the attached target objects.

Carriage object defines the following attributes:

Origin: The origin of the carriage coordinate system.

X, Y, Z: Three axes of the carriage coordinate system.

Drag Object

To use the tool, multi select an animated skeleton, a "floor" object and enter two points through the view window to define gravity. The drag object will then study the movements of the skeleton and translate the skeleton accordingly. Bodies you bind to the skeleton will move with the skeleton.

Or drop the skeleton and the body into a single level and apply the drag tool to the level. In this case also the body points will be taken into account when the drag object computes translation.

Drag object uses collision detection system to determine how much the target object should be moved based on the pointwise translations defined by the target. For example, animate the legs of a robot and drop the robot on a floor. Drag object will make the robot walk on the floor according to the pointwise velocities of the legs.

Drag object affects only the translation of the target objects. It doesn't generate spin to the target objects.

Drag object defines the following attributes:

Rebound: Specifies rebound energy for the collision detection. If the rebound energy is zero, object will not rebound from the surface.

Friction: Surface friction. Setting friction to zero results totally slipping feet effect. Setting friction to max (1) will eliminate slipping feet effect completely.

Inverse Kinematics Object

This is a special purpose geometric sphere that applies inverse kinematics method to the joint it is associated with.

The idea is that one can define independent, reusable motion libraries and associate them with different type of skeletons.

To use this object:

- Select a skeleton and create I.K object over a joint. The created object is automatically bound to the joint inside it. Note that binding is based on inside/outside test rather than selected joints.
- You may also create and animate a set of I.K objects first, then match any skeleton to them and click 'Bind Siblings' tool to set binding with a single click.
- For easy matching, radius can be controlled in edit mode via special purpose 'radius handle'.

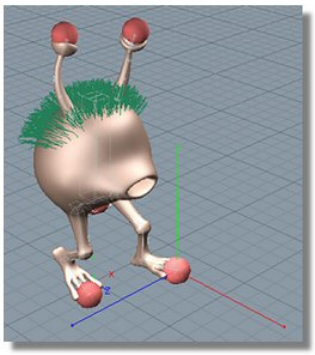
The tool is construction stack based so it can be combines with other type of animation effects. For example, use keyframing to fine tune the motions generated by I.K objects. Add noise to make the character to move and shake, etc.

Inverse Kinematics object defines the following attributes:

Position: Specifies the center point of the Inverse Kinematics object.

Radius: Radius of the object. This can be changed in the property window or by editing a special purpose radius handle in edit mode.

Snap To: If set, the joint is placed exactly on the center of the object. If not set, the initial displacement is maintained.



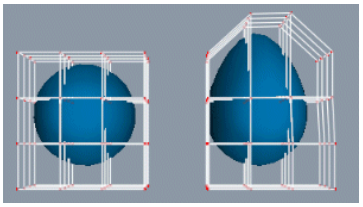
Five I.K objects attached to a body

Lattice Object

The Lattice object is a constructor based deformator. It is a three dimensional grid which is typically used for deforming freeform objects, such as SDS and NURBS surfaces.

The lattice object defines only attributes which specify the number of control points in each dimensions.

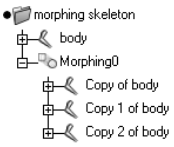
When a lattice object is created, the initial state of it is taken as the state of no deformation. Any modification applied to the lattice object will then deform the target objects accordingly.



The effect of editing a lattice object attached to an SDS sphere

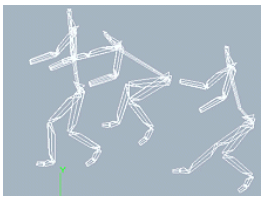
Morphing

The morphing object is a level object, which can contain two or more sub objects. The morphing object interpolates these sub objects and assigns the result to the target object. The target object must be at the same hierarchy level as the morphing object itself and the structure of the target object and the parameter objects must match.



The morphing object defines the following attributes:

- Interpolation - Defines the continuity of morphing. Discrete option jumps from key state to another. Linear morphing follows straight lines between key position. The Cubic option selects smoothly continuous interpolation.
- Extrapolation - Defines what happens when time runs outside the range of defined key states. The following choices are available:
  - Extrapolate - The morphed motion continues outside the key range to the direction defined at end keys
  - Close - Morphing is automatically closed to interpolate back to the first key state. There should be some timeline left after the last key to allow the closing of the loop.
  - Hold - Morphing stops at extreme key position and remains static outside the key range.
  - Repeat - Morphing repeats the modification cyclically. Note: keys are not auto closed to a loop.



The target object 'body' controlled by a morphing object consisting of three key object

- Relative - Selects a relative morph effect. Then changes between the morph keys are added to the existing state of the target. In non-relative morphing, the morphed value replaces the existing target value. Relative morphing allows more flexible control of the target. The disadvantage is that the target object and the morph keys can lose the exact match at key frames (this happens e.g. when the first key is edited after setting up the morph system), which can make key editing afterwards less intuitive.

- Time values - The knot line can be used for controlling the timing. By default, uniform timing is used, that is, the object distributes time evenly over the key objects.

By default, morphing controls only the common geometric attributes. For the sake of efficiency, it does not try to morph all attributes of key objects. However, all attributes that are already set animated at the time of morph binding will be morphed. So, to add color to morphed attibutes, do as follows:

- Create morph keys as usual, and apply the morph tool

- Select the created morph modifier, and select the pull down menu Methods/Constructor/Unbind siblings

- Open the Choreography window. Select the desired component(s) of the morph target object from the list of choreographs, find and click the Color attribute from the Animateable attributes list, and set the Animated check-box. Close the choreography window.

- Morph modifier still selected, apply the menu Methods/Bind siblings

- Edit the color of the morph key objects

Move Constructor

Move constructor translates the target objects based on the specified translation vector.

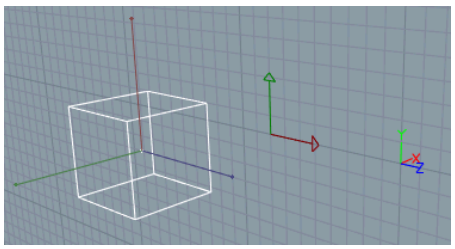
Packer

The Packer object is the key element for parametric object modeling. It arranges its child objects into a row and scales them to fill the space of the packer using a variety f options. 2D and 3D object arrays can be built by stacking packers hierarchically.

The property gadget of packer has the options listed below.

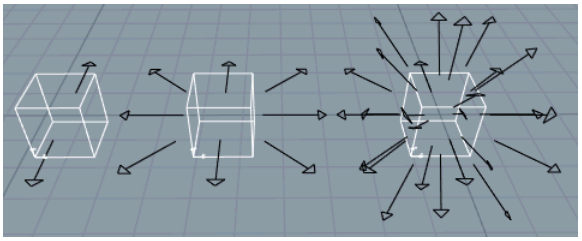
Drag And Drop Snapping options define what happens, when a packer object is dragged from the object browser window and dropped to a view window. The packer object has an ability to search closest other objects and snap into them. Snapping is based on collision detection. You can use the Hit Invisible property of geometric primitives to control and optimize snapping.

- Space: Snapping orientation.
  - Object: Snapping search directions (see Dir1, Dir2 and Dir3 below) are defined in the object space of the packer.
  - World: Snapping directions are defined in world space.
  - IPlane: Snapping directions are defined in the current input plane of the view window.



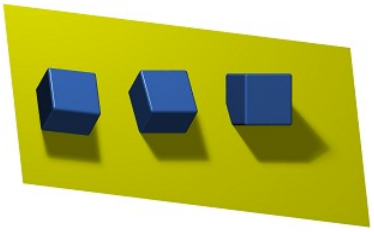
From the left: Object, Input Plane and World spaces

- Dir1: The primary snapping direction. Snapping can move the packer to the direction of a fixed axis, in a 2D plane with a given density (see Sampling below), or in all directions. Note that snapping directions are examined starting from the pivot point of the packer; this is sometimes a relevant factor. Setting Dir1=None disables the snapping functionality (other directions are not considered).



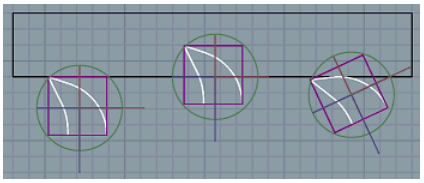
From the left: Z, XZ and XYZ(=all) snapping directions

- Dir2: The optional secondary snapping direction, which takes place after the primary snapping has been processed. For example, a kitchen wall cabinet can first snap to a nearest wall and then move side by side to a closest other cabinet. Note that if snapping space is set to Object, the secondary snapping happens in already rotated object space (see the Rotate option below).
- Dir3: The optional third snapping direction. For example, a kitchen cabinet designed to stand on a floor can first snap to the closest wall, then snap to the floor level and finally snap to the nearest other cabinet.
- Rotate: The packer can be rotated to match the surface tangent at the snapping position. For example, a kitchen cabinet should turn its backside against the wall to which it snaps. Rotate options define the 'route' of rotation as follows:
  - Direct: The packer axis defined using the Axis option below is rotated to match the surface tangent using a single (shortest route) rotation. If the target surface has unusual orientation, this may tilt the object.
  - H, P: The object is first rotated around its vertical Y axis (Heading angle) and then around the axis perpendicular to the selected fitting axis (Pitching angle). Two-step rotation eliminates tilting (i.e. Banking angle remains zero).
  - H: The packer is rotated only around the Y axis.



Three cubes snapped to a tilted plane using Direct, HP and H (=heading only) rotation

- Axis: The axis which is rotated to match the surface tangent. The 8 main axis directions +-X, +-Y, +-Z of the selected snapping space are available. The option Any selects the axis that matches the tangent initially best i.e. selects the minimal required rotation automatically.
- Fit: Defines what is snapped to the found target point.
  - BBox: The bounding box of the packer is fitted to the snap target point.
  - Pivot: The pivot point of the packer is moved to the snap target point.
  - Surface: The surface of the child objects of the packer is fitted to the snap target point using bi-directional collision detection.



Bounding box, Pivot and Surface fitting

- Range: A distance threshold for snapping. Surfaces further than this are not considered in snapping.
- Sampling: Defines how many directions are examined in 2D and 3D snapping cases. For example, if Dir1=XZ and sampling is at the minimal level 1, four directions X, X-, Z and Z- are examined. If level is 2, 8 directions at 45-degree intervals are examined.

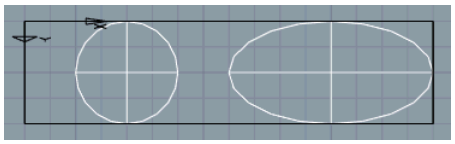
Geometry Managing options:

- Width: The current width (x dimension) of the packer object
- Height: The height (y dimension) of the packer

- Depth: The depth (z dimension) of the packer
- Min. Width, Min. Height, Min. Depth: You can set a minimal size for the packer object using these controls. For example, a kitchen cabinet assembled from 16 mm plate must be at least 32 mm wide (in practice at least 10 cm wide). The minimal size is initialized to the size of the packer at its creation time.
- X, Y, Z Modulo: If X modulo is nonzero, Width of the packer is automatically rounded to a multiply of the modulo value. Y and Z dimension work in a similar way. In other words, Modulo values define an object specific snapping grid for the size. The feature is useful when resizing the object manually using the edit handles of the packer. For example, allowed standard width values of kitchen cabinets can be multiplies of 10 cm.

Packing options:

- Orientation: Defines the direction in which the packer arranges its child objects.
- Target: Selects a child object whose packing is controlled using the options below. If target=All, you can set packing options to all child objects at once.
- Expand: Expand the packing space reserved for the selected target to fill the packer.
- Unmanage: Disables packer management for the selected target. For example, material map objects are usually unmanaged children.
- Fill X, Fill Y, Fill Z: The selected target is stretched in the selected direction X, Y and Z to fill the packing space reserved for it.
- Pad X, Pad Y, Pad Z: Adds extra padding space around the selected target. The padding space is not filled even when the respective Fill option is enabled.



Two spherical objects packed. Both have Expand enabled and hence occupy an equal amount of space. The sphere at the right side has also Fill X enabled.

- IPad X, IPad Y, IPad Z: Expands the packing space reserved for the selected target.
- Anchor: The position of the selected target inside the packing space. Not relevant when the packing space is as large as the size of the target (Expand disabled or Fill options enabled to fill a dimension)

Path Level

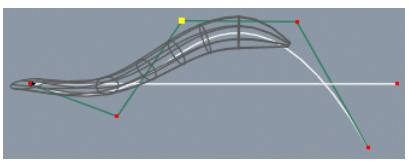
Path Level object can be used for bending geometric objects using a nurbs curve, or any other geometric object. The actual geometric object defining the deformation is created inside the path level object.

Path Level can also be used to create path animations, by setting the Animator option, or by applying it to a moving object.

If the Animator option is not set, then the object uses the defined axis to represent non-deformed initial state and the defined curve to define deformed state.

If the Animator option is set, then the object uses the first defined curve segment to represent the deformed state and the rest of the curve is used for generating the path animation effect.

To create a wriggling snake, just apply the tool to a snake object with the Animator option set, or:



1. Create a horizontal cylinder mesh representing a snake.

2. Animate the snake to move horizontally (move the time slider ot the last frame and translate the object in anim. record mode).

3. Activate the Path Level tool, make sure Animator option is not set and define a the first two curve points along the tube. Play the animation to see a wriggling snake.

A suitable banking option can be selected through the property window.

Point Translator

Point Translator is a constructor based deformator which can refer to number of geometric points in its target objects.

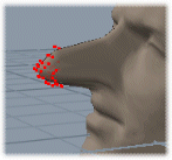
It can be used, for example, to define facial expressions.

The point translator object can be modified or animated similar to any other geometric object, in which case the effect is applied into the actual target points.

To use the point translator:

1. Select desired target points and click the Point Translator tool.

2. Activate the Construct option for the object and modify the translator object. The associated target points are modified accordingly.



Moving a point translator which deforms a nose

The Point Translator object defines Weight attribute, which can be modified in the Spec tab of the Property Window.

A point translator object can refer to any number of objects and points.

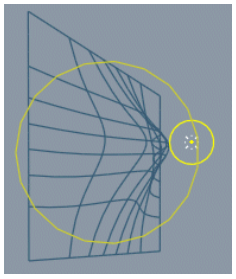
Radial Deformator

Radial deformator is a constructor based deformator. The shorter the distance to the center point of the deformator the stronger the deformation effect.

The initial position and radius of the deformator represents the state of no deformation. The desired deformation effect can then be achieved by editing the center and the radius of the sphere.

All points inside the initial radius will be influenced by the deformator. All points outside the sphere will not be affected by the deformator.

By moving the center point, one can achieve a magnet like effect. All points are moved with the sphere. The higher the distance of the point and the center of the initial sphere the smaller the translation.



A radial deformator applied to a nurbs mesh

To deform a nurbs mesh:

1. Create a nurbs mesh consisting of say 10 x 6 control points

2. Activate the radial deformer tool and specify two points defining the center point and radius attributes for the deformer.

3. Switch to edit mode and drag the center point or the radius handle of the deformer. The target points are deformed accordingly.

Setting the radius attribute to a smaller value generates a gravity like effect. In other words, target points are scaled towards the center point rather than translated with it.

The strength factor can also be controlled by defining a deformation curve through the property window. The default curve is linear. The beginning of the curve represents the maximum distance of influence and the end point corresponds to the zero distance.

The deformator also supports fractal noise deformation effect. The strength and density of the noise effect can be defined through the property window. This option is useful for creating turbulence effects. For example, a star ship can be moved through a noise field.

Rotate Constructor

Rotate constructor rotates the target objects about the specified axis.

To rotate the selected objects by three full circles:

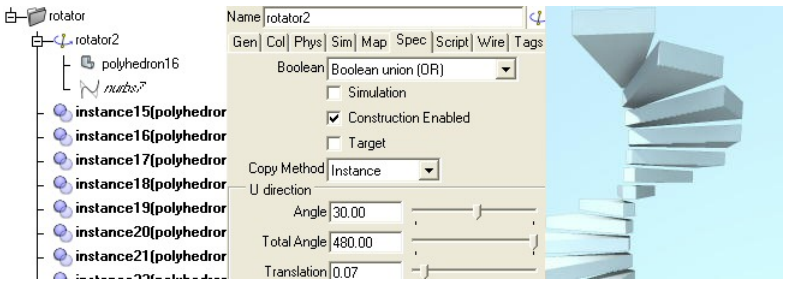
1. Click the Rotate constructor tool to activate it:

2. Enter two points in the view window to define rotation axis.

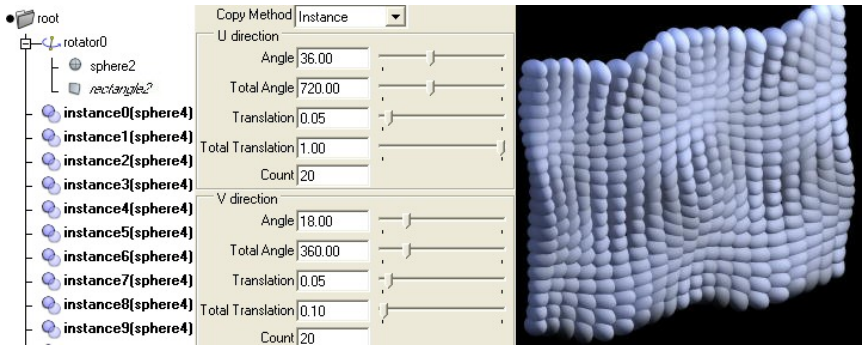
3. Specify the desired rotation angle through the property window. To make the target object to rotate three full cycles, enter 3 * 360.

Rotator

A rotator object produces duplicates or instances of its first sub object by rotating and/or translating the copies into regular rows and circles. If the second parameter object is one dimensional (a NURBS curve), the result is a sequence of objects. If the second sub object is two dimensional mesh, the result is an array of objects.



Spiral stairs created by a 1D rotator system



A sphere wave created by a 2D rotator system

The following set of controls is available. The 'U direction' controls are applied for all kind of parameters. The V direction controls become relevant when the 2nd parameter is two dimensional. Both sets of controls work in a similar way.

Copy Method: The object duplicates can be either real objects or instances. Instancing saves some memory especially when the sample object is complex.

Angle: Defines how much a new duplicate is rotated from the previous duplicate around the current parameter object evaluation position.

Total Angle: The same as the previous one, but expressed as the total rotation for all duplicates.

Translation: Defines how much a new duplicate is moved along the 2nd parameter U/V dimension. The amount is measured in parameter space i.e. value 1 corresponds to full parameter length.

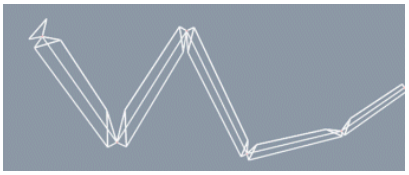
Total Translation: Amount of total translation from the sample object to the last copy/instance.

Count: Controls the amount of resulting objects (or object rows/columns in 2D case).

Skeleton

Skeleton is a deformator. Geometric points are bound to the joints of a skeleton through a set of binding tools.

In addition to this, it is also possible to use the lattice mapping system to attach desired objects to skeletons.



Skeleton properties are described below. For more information, see also Skeleton Tool documentation.

Default Joint Action

This option determines the action to be taken when the user drags a joint. The following options are available:

- Inverse Kinematics: Inverse Kinematics is used to move the selected joints. I.K propagates symmetrically to both directions translating joints according to their properties. Alt key can be used for enabling cumulative mode where the original joint positions are not restored in each mouse move.
- I.K Head, rigid tail: Inverse Kinematics propagates only to the joints at the head side. The tail side remains rigid.
- I.K Tail, rigid head: When set, Inverse Kinematics propagates only to sub joints. The head side remains rigid.
- Forward Kinematics: Applies forward kinematics to the joint. The dragged joint rotates about the previous joint.
- Move joints: The selected joints are translated, as if the skeleton was completely flexible. Angle constraints are expanded, if needed, to allow the defined translation.
- Modify lengths: Dragging a joint moves the joint closer/farther away to/from the previous joint.

Preferred Anim. Method

Defines the preferred method for key framing:

- Angles - the key framer records the bone angles for each joint.
- Inverse Kinematics: The key framer records the position of the selected joint only. When the animation is played back, the joint in question is positioned using Inverse Kinematics. Because of this, the state of the skeleton is not well defined, but depends on the initial state.

I.K Sampling

This option specifies the level of sampling for Inverse Kinematics. The higher the value the better I.K works.

I.K Parents

Specifies the number of parents I.K propagates to. If the value of this attribute is zero, I.K will never propagate to the parent object but affects only the current skeleton.

Mapping Volume

Mapping volume is a control hull with a circular cross section. It is used when one needs to bind a skeleton to an object to be deformed. Points inside the mapping volume will be bound to the skeleton and points outside the bounding volume will not be bound.

In hierarchical skeletons, mapping volumes typically overlap in which case certain point may get bound to more than one skeleton.

There are two options controlling mapping volumes:

- Mapping volume On/Off - to enable/disable mapping volume property
- Mapping Radius - radius of the mapping volume for the skeleton

Parent Joint

Skeletons can be grouped to a hierarchical tree where each skeleton may have one or more sub skeletons attached. For example, a hand skeleton may have five sub skeletons representing fingers.

This option specifies the parent joint to which the skeleton in question is attached to. Bone angles for such a skeleton are defined in the space of the parent joint.

Rotation from Parent

If this option is set, both the translation and the rotation of the skeleton is defined in the parent bone's space. If the option is not set, the rotation is defined independently from the orientation of the parent skeleton.

This option is effective only for skeletons attached to their parent skeletons via the 'Parent Joint' option.

Displacement

This option specifies how much the root joint of the skeleton is displaced from the joint of the parent skeleton. This option can be modified simply by moving the skeleton with the Move tool, or by defining displacement through the property window.

Interpolation

Realsoft 3D key framer supports two interpolation methods for key framing rotations.

- Angle Space
- Quaternion space

Angle space interpolation simply interpolates recorded rotation angles. The drawback of this method is that the two dimensional angle space has two singularity points. Any path passing near these points typically cause problems.

Quaternion interpolation solves this problem by using four dimensional angle space for interpolation. This is an advanced method and results very nice rotations. However, the problem is that the resulting animation curves are somewhat difficult to control.

Angle System

Skeletons support two kind of angle systems:

- HPB - Heading Pitching Banking
- HPT - Heading Pitching Twisting

HPB is the standard HPB angle system (stands for Heading, Pitching and Banking). The first two angles define polar coordinates for the bone. The third angle 'Banking' is defined so that as long as the 'x' axis of the bone lays on the ground plane, banking is said to be zero. This makes the method particularly suitable for controlling cameras. With zero banking, the x axes of the subsequent joints lay on the common plane.

HPT is a new angle system developed by Realsoft Graphics Oy.

The difference to the standard HPB system is that the third angle is defined in such a way that it minimizes the surface tension of the attached body rather than trying to keep the bones horizontally aligned.

Think about a striped flexible plastic pipe. Bend it to any direction and the stripes along the body tell you intuitively whether there are twisting or not. If the stripes appear spiral like, there is twisting involved. If stripes along the body follow straight lines, the twisting is zero.

In other words, when there is no real twist on the body attached to the skeleton, the third angle 'Twist' is said to be zero. What could be a more natural way of defining bone angles!

This system has several advantages over the standard HPB angle system. First, you typically have to edit only two bone angles: heading and pitching. You need to touch twisting angle only when you really need to get some real skin twisting introduced. In the standard angle systems, the banking angle has nothing to do with real twisting. In animations, you have to edit the banking angle all the time to eliminate unwanted twisting from the body. Another advantage is that I.K. can now operate based on heading and pitching angles only. This makes the I.K. tool much more useful.

The third and significant advantage of the new HPT angle system is that it helps to eliminate the biggest problem with angular spaces. In any angle space, there are two so called singularity points - the poles. When you animate, say, a camera over these poles, you'll get sudden jump at the north pole. The new HPT system eliminates this problem completely. In this respect, HPT angles are almost as good as Quaternion angles, and easier to handle in the curve editor.

Joint Angles

The orientation of each joint (or bone) can be controlled by three angles, named Heading, Pitching and Banking.

The joint angles are defined in the space of the previous joint. There is no previous joint for the root joint, in which case the angles are defined in the object space.

Skeleton has so called native orientation, in which the skeleton can be bent +/- 180 degrees. This corresponds to the 'a' (heading) angle of a bone. Joint representation visualizes this heading angle and intuitively shows you how much the joint in question can be bent.

In pitching angle, skeleton should only be bent less than +/- 90 degrees to avoid those problematic singularity points of the angle space. You can bend the skeleton beyond this limit, but in animations the skeleton may not behave as you expect. The reason for this is that the angle space has two so called singularity points at pitching = +/- 90. If you think the angle space as the surface of the earth, these singularity points correspond to the north and south poles. There, any latitude value will give you the same point (the pole). Therefore, you should try to create a skeleton so that the direction in which it bends the most, corresponds to the heading angle. This problem is not specific to Realsoft Graphics Oy's implementation but is a fundamental problem related to polar angle systems.

If you need to create a joint which bends over +/-90 degrees in both directions, you may construct them from two joints close to each other. In fact, many real world joints are constructed this way. Another solution is to use quaternion angle system.

Constraint Angles

Constraint angles can be used for constraining bone angles. If a skeleton is bent over the constraints, the joint friction is internally increased preventing the joint from bending further and keeping the bone angles within the specified min / max constraint angles.

Note: If quaternion angles are used, constraints are effective only when interactively editing the skeleton. Constraints are disabled for animation play back.

Bone Length

Each joint defines a length attribute specifying the distance to the next joint.

Thickness

Thickness attribute specifies the thickness of the skeleton at the joint in question. This attribute is only used for visualization. It can be used for representing the actual body.

Position

Joint position attribute is actually an optional interface to joint angles. Skeleton can be defined by the root position and a set of joint angles, or by specifying joint positions for each joint.

Joint friction

The higher the friction, the more rigid the joint is when editing the skeleton using Inverse Kinematics. If the joint friction is 1, the joint is completely rigid and the two associated bones act as a single curved bone.

Fidelity

Fidelity attribute is based on multi weighted binding. If the fidelity is zero, then the each point is single bound to the bones. If the fidelity is 0.5, then the skeleton handles the binding as if the point was multi bound to subsequent bones.

This option specifies an object wise fidelity value. It is also possible to use point wise fidelities.

Drag

This attribute specifies the drag force between the joint and the medium. The higher the value, the stronger the joint resists attempts to translate it. Infinitely high drag force anchors the joint to a world.

Anchor

Anchor option anchors the joint coordinates in the world space or in the object space when the skeleton is modified using the Inverse Kinematics tool.

If the joint is anchored in the world space, the joint maintains its position even when the entire object is translated.

If the user modifies an anchored directly, the anchore is temporarily disabled for the joint.

Rigid

Setting this option makes the joint rigid. The effect is the same as setting joint friction to 1.

Animate as I.K Target

When this option is set, the key framer records the position coordinates. In animation play back, the joint is moved to the specified point using inverse kinematics.

Animate as Angle

If set, the key framer records bone angles for the joint. This option is typically used when Preferred animation method is set to Inverse Kinematics and when some of the joints need angle based key framing.

I.K Head Stopper

Inverse Kinematics stopper. When one of the joints at the tail side is modified using I.K, this option prevents I.K from propagating to parent bones. The option has no effects when I.K is applied to parent bones.

I.K Tail Stopper

Inverse Kinematics stopper. When one of the joints at the head side is modified, this option prevents I.K from propagating to parent bones. The option has no effects when I.K is applied to tail side bones.

Binding

Skeleton acts similar to other construction based deformators. The main difference is that skeleton must be *bound* to target objects before it can deform any target points.

When the skeleton is bound, its current state is saved. When the skeleton is then modified, a deformation is computed from the difference between the initial state and the current state.

Binding becomes effective only when the *Construct* feature is enabled for the skeleton. Resetting *Construct* option does not discard binding data.

Two methods exist for setting up initial binding.

Single binding binds target points to their nearest joints of the skeleton.

Multi binding uses bounding volume property to determine the skeletons to which the points should be mapped.

It is also possible to set up or fine tune binding manually.

Bind Selected tool binds the selected points to the selected bones. The tool is shown when one target object and one skeleton object is selected. This tool does not set up multi binding. If multiple bones are selected from a single skeleton, point is mapped to the nearest one.

Unbind Selected unbinds the selected points from the selected bones. The tool is shown when one target object and one skeleton object is selected.

Unbind tool detaches all target objects from the skeleton. All binding specific data is freed from the skeleton.

Select tool allows the user to select the points bound to the selected joints. This tool selects all points bound to the currently selected bones. The tool is only shown when both the target object and the skeleton object are selected. The tool is useful, for example, when one wants to view the points bound to a certain bone. Or, when some of the points got bound to a wrong bone and you want to unbind them.

Unselect tool unselects all points bound to the selected bones. The tool is only shown when two objects are multi selected: the skeleton object and the target object. The tool is very useful when binding points manually to complex hierarchical skeletons because it allows the user to exclude points which are already mapped to other skeletons.

Handles

Skeleton supports a set of handles through which the user can control the skeleton.

Handles can be switched on / off through the Wire tab of the Property Window, or through the compass menu.

Joint Handles

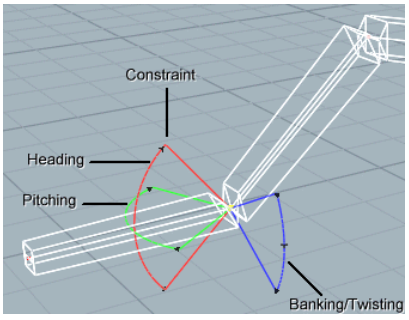
Each joint acts as a handle activating the tool specified by the 'Default Joint Action' option. The default tool is Inverse Kinematics.

Angle Handles

Angle handles can be used for controlling bone angles and constraint angles.

Dragging an angle handle modifies the corresponding bone angle, such as Heading, Pitching or Banking.

The constraint handles are marked as black arrow heads at the both ends of each angle handle.

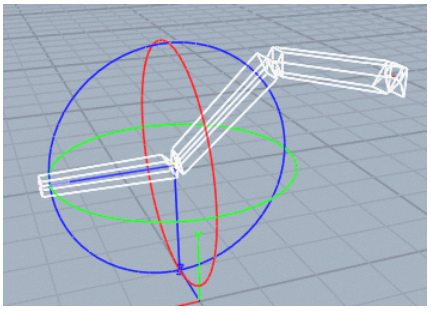


Angle handles

Rotation Handle

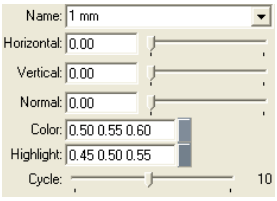
The rotation handle rotates the selected bones exactly the same way as the standard rotation handles rotate regular geometric objects. The difference to the bone Angle Handles is that the rotation handle rotates the bone about the main axes of the bone whereas bone Angle Handles control certain bone angle directly. In order to get the desired rotation applied, rotation handles typically need to change all the three bone angles.

The rotation handle can be used for rotating the selected bone. Constraints can be disabled by holding down the Alt key.


Bone rotation handles

Grids

Grid object has the following properties:



- Name: The name of the grid object
- Horizontal, Vertical, Normal: sets the grid dimensions, which define the distance between snap positions. Usually, all three fields are set to the same value. If you wish to enter small values like 0,001, make sure the value of the options/window/metrics tab is set to match.
- Color: The base color used for grid drawing.
- Highlight: A second color used for drawing the grid lines, whose frequency is determined using the 'Cycle' slider.
- Cycle: Defines the density of grid lines that are drawn with the highlight color.
- Dynamic: Automatic adjustement of snapping interval.
- Dynamic Color: Automatic adjustement of grid color for dynamic grids. The shorter the snap interval, the lighter grid is drawn.

You can activate the desired grid by dragging and dropping it into the view window. This sets the current grid for the view window and activates both 'snap to grid' and 'show grid' options. You can also use view property window's 'Input' tab for controlling view's grid snap and draw options.



Note

Do not mix the grid with the ground plane visualization in perspective views. The ground plane has nothing to do with snapping and is provided as an orientation guide only. It is automatically turned off when grid drawing is enabled.

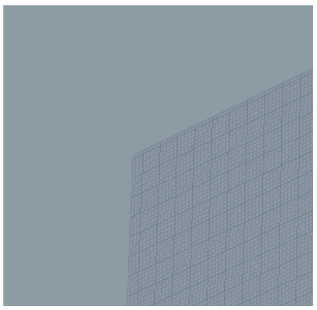
The view control bar allows you to switch snap to grid and show grid options on/off quickly any time.

  *Draw Grid and Snap to Grid icons*

The grid drawing fills the entire view window. If you zoom out the view window so that the grid appears denser and denser, the grid system stops drawing the base colored lines at some point, because the too densely drawn lines would not serve any useful purpose. Drawing only the highlighted lines keeps grid visualization useful. If you continue zooming out, also highlighted lines get too dense and their drawing will be stopped.

The same density control is used when perspective projection is enabled and view camera is not locked to the grid plane. Only the part of the infinite grid plane, which is not too dense, is visualized.

The automatic density control does not affect grid snapping, only grid drawing.

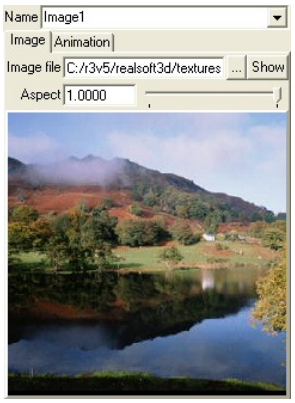


Too dense part of the grid is not drawn in perspective mode

Images

An image object displays a disk image file or a frame of an animation file (e.g. an AVI file). You can use it for viewing purposes. For example, when 3D painting an object, the system automatically adds an image object which displays the painting result in real time.

The image object properties are:



- **Name**: A symbolic name for the object.
- **Image file**: The disk file name. The path may be omitted if image search paths include the parent folder.
- **Aspect**: Pixel aspect ratio height/width. With this control, you can fix stretching of images coming from devices, which have non-square pixels (such as popular TV standards PAL and NTSC).
- **Animation**: Frame index controls. With these, you can activate automatic image updating by animation time. For example, if you want to synchronize a 3D animation to a live video stream, just create an image object that uses an avi file or a frame sequence, activate Automatic animaton indexing from the Animation tab and drag&drop the image object to a view window. The view window backdrop will automatically update whenever you change the animation frame. For detailed description of these controls, see the Animation tab documentation.

Materials

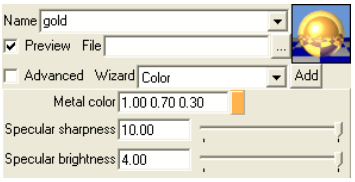
Material Types

Realsoft 3D has an open architecture, which means that a material object can have any kind of property or interface. This chapter gives an overview of the material types included in the program. You can find the collection of installed material classes by opening the 'New' sub menu from the popup menu of the select window's material section.

VSL Material

This is the most powerful and commonly used material type of the program. All materials, which you can find in the default material library, are of this type. The reason why VSL materials are so powerful is that the 'low level' interface of the VSL materials is a programming language. Advanced users can therefore implement unusual and customized material effects. Don't be alarmed by the term 'programming language', you do not have to write complex formulas and algorithms, because VSL materials can be built using wizards and an easy drag and drop interface. Consequently, it is easy to get started. Nevertheless, in order to fully utilize the possibilities of VSL, some programming experience is required.

Using VSL materials is very easy. VSL makes it possible to build an optimal interface for each material type. You can see this by examining the materials of the default library in the property window.



A VSL material in showing its 'easy to use' interface

The following control groups are available for VSL material building and editing:

- Top Level Controls
- Advanced Controls
- Object Specific Tab
- General Tab
- In/Out Tab

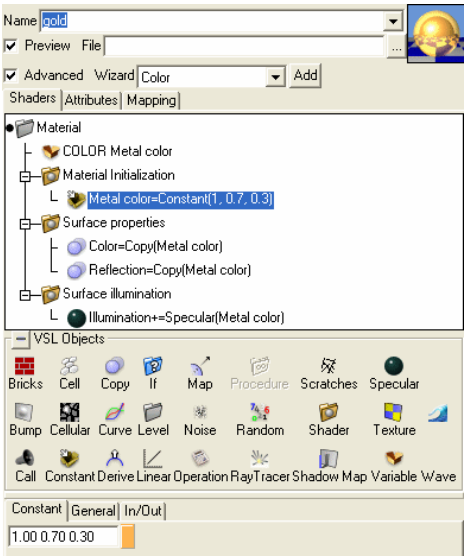
Top Level Controls

The topmost area of the VSL material editor shows a set of gadgets that are displayed both in the easy to use interface and in the advanced editing state. These controls are:

Name: The name of the currently edited material. The editing target can be changed by choosing another material from the list.

Preview: This checkbox enables/disables the material preview feature. When enabled, the program automatically renders a test image of the edited material after each change. The preview is rendered using a test scene defined by the current material preview project. It is also possible to assign a special preview project to the material by selecting a disk project file using the 'Test Scene' gadget at the right side of the 'Preview' check box. The rendered preview image automatically updates the material icon in the select window (if preview icon feature is enabled).

Test Scene: A custom preview project. If a project file name has been defined to this field, the program automatically loads the project and uses it for material previewing whenever the material is edited in the preview state. The idea is that some materials may require a certain kind of preview project which shows the characteristics of the material well. For example, a transparent material requires some objects behind it so that refraction can be tested. The 'previews' folder in the Realsoft 3D home directory contains some predefined preview projects. You can create more of them by editing the default material preview project and saving the result to the 'previews' folder with a suitable name.



A VSL material in the advanced editing state

Preview Window: The squared little window at the top right edge of the material editor is actually a View Window. Therefore, you can rotate, zoom and pan it with the right mouse button and Alt, Shift and Ctrl modifier keys. The window also has a small popup menu:

- *Reset*: Resets the viewing position back to the default front view.
- *light properties, material preview, ...*: The rest of the menu items show the contents of the 'preview' folder. You can choose a suitable preview project here instead of browsing from the 'File' gadget.

Disabled: Turns off all properties of the material. If the material is mapped to a geometry, the geometry renders as if the material was not mapped at all, or deleted from the material library.

Advanced: This checkbox brings up the advanced VSL editing controls.

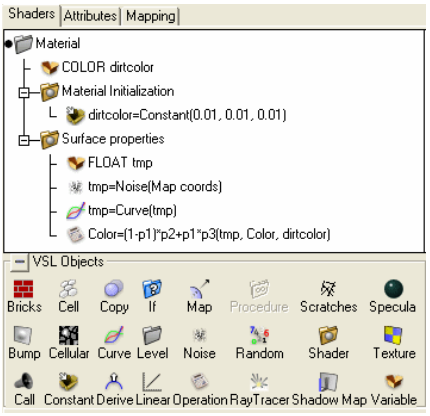
Wizard: A list of VSL wizards, which automatically add some commonly needed material appearances and features to the edited material. For more information, check the wizard list.

Add: Adds the currently selected wizard elements to the edited material. As a result, a set of new VSL objects and top level controls appear to the material. Note that the same wizard may be applied several times to the material; the program does not check or prevent this. For example, a material may include several texture maps which are blended together.

Advanced Controls

The advanced VSL controls are divided into the following three tabs:

Shaders Tab



Shader Tree - The tree diagram shows the complete structure of the 'shading program'. The top level objects, 'Shaders', are executed in a predefined fixed order, regardless of their order in the constructed VSL tree. Inside the shader levels, the execution proceeds in top-down order. Variables can be placed freely into levels - position does not matter - but is is a good programming practice to introduce them as the first objects in a level. VSL objects can be inserted and moved around using drag and drop. Ctrl-dragging creates a duplicate and moves it. The shader tree gadget has the following popup menu (it is context sensitive and changes depending on the VSL object selection):

- *New* - Contains a list of available VSL objects. Selecting a sub menu item adds the corresponding VSL object to the currently edited VSL level.
- *Cut* - Cuts the selected VSL object (and its possible sub objects) into Realsoft 3D's VSL clipboard.
- *Copy* - Copies the selected VSL object into Realsoft 3D's VSL clipboard.
- *Paste* - Adds the contents of the Realsoft 3D VSL clipboard to the end of the edited VSL block.
- *Duplicate* - Duplicates the selected VSL object to the end of the same level.
- *Delete* - Removes the selected VSL object.
- *Output* - You can select the output channel/variable of the selected VSL object from this menu.
- *Operator* - The mathematical operation which is used when assigning the result of the evaluation of the selected VSL object to its output parameter.
- *Input0, Input1 ...* - You can set the input parameters for the selected VSL object with these menus.
- *Rename* - Renames the selected VSL object (slow double click also works). VSL objects can be renamed to make the code better understandable for other users.



Note

Renamed objects no longer update their names automatically when the input and output parameters change.

- *Disabled* - Removes the selected VSL object from rendering computations. This feature is useful when building and experimenting with the VSL code.
- *Balanced Layout* - The VSL hierarchy will be displayed using horizontal tree layout.
- *Properties* - Opens a floating window which shows the object specific attributes of the selected VSL object.

VSL Objects - An array of currently installed VSL objects. You can drag and drop items from this array to the VSL shader hierarchy. Clicking a VSL object icon adds it to the end of the level where the currently selected VSL object is located. Note that VSL object hierarchy follows some syntax rules, and a certain object may not fit into some parts of the hierarchy. For example, the root level of the hierarchy can contain only variables and shaders, and a shader cannot be placed into another shader. When using drag and drop, the mouse pointer symbol tells you where you can drop an object. If you click a VSL icon in the array and nothing happens, the reason is most likely that the current destination is not allowed.

Attributes Tab

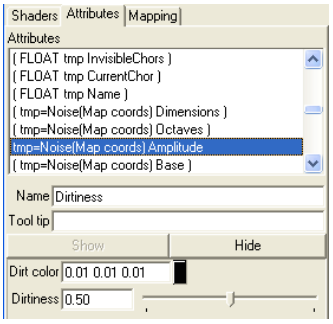
The Attributes tab contains the tools for creating an easy to use interface for VSL materials. The controls are:

Attributes - A list of all attributes of all VSL objects in the current material. You can pick an item from this list and make it visible in the easy interface with a suitable name. The list includes a simple popup menu, whose items 'Show' and 'Hide' serve the same purpose as the two buttons described below.

Name - Displays the name of the selected attribute. By default, attribute names describe their general nature. For example, 'Amplitude' defines the magnitude of the 'Noise' object's value variation. You can change the name to describe the particular purpose of the attribute in the edited material. For example, 'Wave Height' might be a better name than 'Amplitude' for a sea water material.

Show - Adds the currently selected attribute to the easy interface.

Hide - Removes the currently selected attribute from the easy interface.





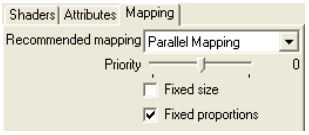
Note

Attributes of deleted VSL objects automatically disappear from the constructed interface.

At the bottom of of the Attributes tab, you can see the defined GUI for the material. The same interface will become visible when you turn off the advanced editing state.

Mapping Tab

The rightmost tab of advanced controls defines how the created material should be mapped to target objects. The specifications made here are used by the drag & drop system. The user can overrule them by selecting another mapping type from the mapping creation tools.



The following controls are included:

- **Recommended Mapping**: Select the preferred mapping type from this gadget.
- **Priority**: Evaluation order recommendation. By default, materials are evaluated in the order they appear in the hierarchy, deep sub level materials last. The hierarchy defined evaluation order can be overruled with the mapping priority setting. The higher the priority, the earlier the material will be evaluated.



Note

This means that the material will have a weaker effect! Later evaluated materials can overwite the result. Some special materials, such as a global illumination shader, may work best if it is evaluated last. Because most materials have the default priority 0 and only some built in shaders use evaluation priority -2 .. +2, a value -5 is usually sufficient to guarantee that the material is evaluated last.

- **Fixed size**: If set, the size of the mapping geometry does not automatically resize to match the size of the mapping target object.
- **Fixed proportions**: If set, the proportions (width, height and depth) of the mapping geometry do not automatically stretch to match the size of the target object. For example, parallel map always becomes a symmetric cube if this option is set.

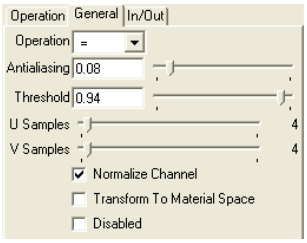
If the recommended mapping is set to 'Any Mapping', drag & drop system creates a mapping that best matches the geometry of the target object. Spheres will get a spherical mapping, cylinders gets cylindrical mapping, etc.

Object Specific Tab

The leftmost tab on the bottom of the VSL editor window shows the object specific attributes of the selected VSL object. Therefore, the tab contents changes from object to another. Object specific VSL attributes are described in VSL objects.

General Tab Options

The majority of objects have several common controls, shown in the 'general' tab of the VSL editor.



Operation:

This option defines how the value computed by the object is assigned to the output channel/variable. If the output channel has 3 subchannels, the same assignment is considered for each sub channel. Operation can also be changed using the popup menu of the VSL editor.

=	Default assignment, overwrites the old value with the new one
+	Adds the new value to the old value
-	Subtracts the new value from the old value
*****	Multiplies the old and the new value
/	Divides the old value by the new one
Min	Assigns the minimum of the new and the old value to the output channel
Max	Assigns the maximum of the new and the old value to the output channel
Abs	Assigns the absolute value of the computed expression to the output channel

Normalize channel:

This option is provided for convenience. Many of the available channels are such that they must be normalized whenever their value is changed. The term 'normalize' means that the value of the channel is divided by its length, the result being always a unit vector. For example: normalization of vector (3, 4, 0) equals (3/5, 4/5, 0). The channels, which are (and should be kept) in normalized form are:

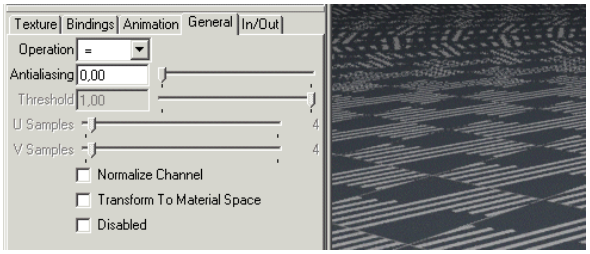
- Surface normal
- Bump normal
- Ray
- Reflected ray
- Refracted ray

Transform to material space:

This option performs a 3D transformation to the orientation defined by map x, map y and map z channels

(x,y,z) -> x*map x + y*map y + z*map z

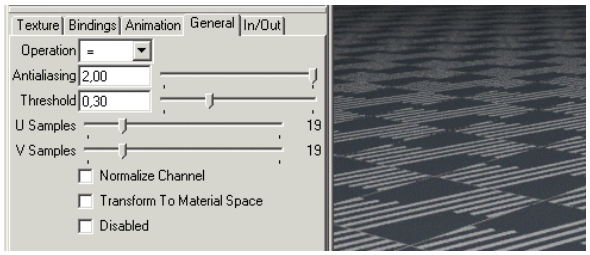
The orientation defined by these vectors is often suitable for surface normal modifications, such as bump mapping.



Antialiasing, Threshold, U-samples, V-samples: These settings control material anti-aliasing. Generally speaking, anti-aliasing is a method of increasing computation accuracy because of high 'information density' or 'level of detail'. In other words, if an examined pixel contains plenty of details, the rendering system needs to analyze that point carefully to achieve proper image quality.

Antialiasing defines the relation of how quickly computation accuracy is increased when level of detail grows. If the value is zero, anti-aliasing computations are switched off. A good default value is 1.

The Threshold setting controls the level of detail when the anti-aliasing starts. The lower the value, the easier antialiasing starts. The minimum value is 0.



The effect of VSL anti-aliasing in a texture map. The quality of the first picture (no anti-aliasing) is not sufficient for animations because of intolerable amount of flickering. The second picture shows the improved quality after activating the anti-aliasing feature.

U-samples and V-samples define how many times the object is maximally evaluated for anti-aliasing purposes. The system is designed so that by setting the u & v samples to the level that corresponds the 'native' level of detail of the object (for example: pixel resolution of texture map object), anti-aliasing factor 1 and threshold value of 1 are optimal. However, because of speed reasons, it is not always possible to use high enough sample values (1000000 samples for a 1000*1000 pixel texture map is too slow), so sometimes lowering the trigger value and increasing the antialiasing value may be necessary.

Note also that sometimes it may be necessary to use higher u-v-sample values than the texture resolution: for example, a simple 2*2 pixel checker pattern may require 4*4 samples because of the random nature of the anti-aliasing process.

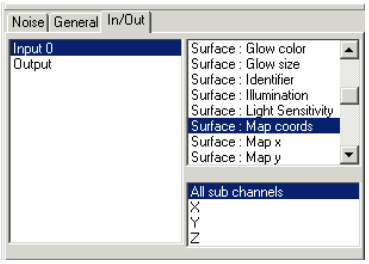
Disabled

This option disables the object in ray trace rendering. The result is the same as if the object were deleted. The option is useful when experimenting with the materials.

Tip

The Tip string can contain some optional information about the purpose of the VSL object, instructions how to adjust the shader or other user-defined information. The Tip string control is the preferred way to add such comments, because renaming a VSL object stops the automatic name updating when input or output parameters or other attributes affecting the name of the VSL object change.

In/Out Tab



VSL object parametrization can be controlled using the 'In/Out' tab of the property window.

The list on the left side shows all input and output parameters of the selected VSL object. The list on the top right side shows how the parameter is assigned. The list includes all channels and variables available in the current shader.

The third list is for sub channel selection. The following examples illuminate this feature:

- Parametrization by all sub channels: myparam=linear(map coords)
- Customized sub channel parametrization: myparam.x=linear(map coords.y)

VSL Procedure Library

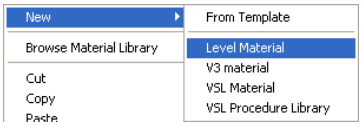
A VSL procedure libary is an object which can contain a set of general purpose VSL functions. For example, an advanced user may create a VSL procedure library called 'Noise Functions' and place frequently needed noise related VSL algorithms there.

When a procedure library is selected in the VSL editor, only Procedure objects can be added at the top level of the VSL hierarchy. Each procedure defines a symbolic name, a parameter list and some VSL code. Note that VSL objects in a procedure can refer only to the defined input parameters and to the output value (procedure's return value). This supports good programming. All required data must be passed in as parameters. Input parameters are used by address, not by value, so assigning a value to an input will change the original parameter defined in the call. This also means that a procedure can return multiple output values.

All defined procedures in all memory-resident procedure libraries can be called from the usual VSL materials. A procedure call is defined using the Call VSL object. A call to a non-existing (unloaded or deleted) procedure does nothing.

Level Material

The Level material allows you to combine existing materials to construct more complex materials. Thanks to this, you can map several materials to an object using a single mapping object. The tutorial 'Blending Materials' already showed one typical use of the level material: combining a material and a 'scope mask', which defines where the material has influence.

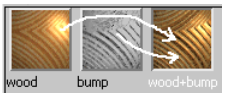


Create a Level Material

Creating level materials

Level materials can be created from the select window's New menu (when the Materials tab is selected).

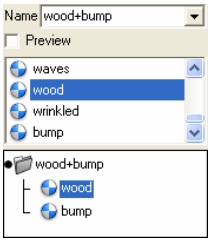
This material can refer to any number of other materials. The sub materials of a level material can be selected simply by dragging the desired materials into the level material in question.



Wood and bump are dragged and dropped into a level material. The result is a bumpy wood material

Modifying level materials

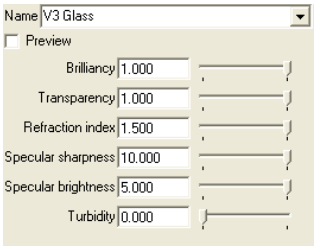
Level materials can be edited as usual: just double click the desired level material in the select window and the property window is opened allowing you to edit the material. Also the property window supports drag&drop. Drag the desired materials from the upper material list into the level material.



A level material consists of two sub materials: wood and bump

V3 Material

This material type implements the same kind of controls for optical material properties as Real3D v.3. 'Brilliancy' property defines the amount of mirror like reflection. The amount of diffuse shading is automatically computed from brilliancy: if brilliancy is zero, diffuse shading is at its maximum and vice versa. The material can be used for basic mirror, glass, etc. and it renders quickly.





Note

The more general VSL material allows all effects that 'V3 material' can define, but V3 material is slightly faster to render.

Samples

The channels are not used 'one by one' in the rendering engine; instead, they are managed as sets of channels. One such set contains space to store all the available channels. A channel set is called a 'sample'. Because all channels are included in a sample, it can store all attributes of an object in a point of space.

One sample is not enough to store the data for the entire shading process. In order to fully utilize the capabilities of the shading system of Realsoft 3D, one must know the relation between the sample objects the rendering engine uses and the shaders.

The sample objects are named according to the purpose they are used for. In some shaders, only one sample is available; some shaders provide two samples (perhaps even more in future releases or plugins for the software). For example, in the surface illumination phase, the first sample describes the properties of the surface, and the second sample describes the properties of the light. In such a situation, the channel names are prefixed by a name that specifies to which sample the channel belongs. For example, Light:illumination is the channel that contains the color of the light from a light source, and Surface:illumination is the channel that contains the total color emitted form the surface.

Below is a list of sample objects the rendering engine uses to carry out the shading process.

Light sample

This sample contains the properties of a light ray (from a light source). The sample is used first in the light properties shader, next in the surface and volume filtering shaders and finally in the surface/volume illumination shaders.

Surface sample

The most frequently used sample. Describes the properties of a surface (boundary of a geometry). The surface sample is available in many shaders and it is used to carry the result information through the rendering system. The ray tracer computes its default shading effects to the illumination channel of the surface sample. It contains the 'result' color of the rendering process at the end of the ray tracing.

Surface filter sample

This sample is used when examining shadow casting properties of a surface in the surface filtering shader.

Particle sample

The particle sample is used to examine properties of particle objects that are drawn in post processing. The sample is used only in the post particle phase.

Ray hit sample

This sample is used with ray-surface hit events. The main purpose is to modify channels that affect the shape of the surface. The most important of such channels is the distance channel. This sample is used only in the ray hit shader.

Volume sample

The Volume sample is used to examine properties inside objects (the surface sample is used on the boundary). It is used first in the volume properties shader, then in the volume illumination and volume shading shaders.

Volume filter sample

This sample is used when examining shadow casting properties of a volume in volume filtering shaders.

Shaders

The highest level of a VSL material hierarchy consists of Shaders. A Shader is a VSL object that has only one property: 'Type'. 'Type' has values like 'Surface properties', 'Surface illumination' and 'Volume properties'. These names, quite obviously, reflect the actions that the rendering engine performs.

The order in which the shaders appear in the hierarchy is not significant. For example, the rendering engine needs to know surface properties before it starts computing surface illumination, and therefore the corresponding shaders are executed in this order, although the user may have inserted them in reversed order.

The set of channels, which is available for use in each shader type, varies. This helps to understand when channels become initialized and which channels are sensible to change. For example, in the 'Volume properties' shader, the surface normal channel is not available at all. This is because it does not make sense to deal with a surface attribute inside an object. In this way, the shader structure guides the user and helps to understand the rendering process.

The shader system is expandable. Plugin modules may insert new shader types.

The default shader types are listed below. These shaders describe the steps of the rendering process quite accurately, so rendering can be customized quite thoroughly using the shaders. For example, the default surface lighting model (standard diffuse lighting) can be redefined completely using suitable objects in the 'Surface illumination' shader.

Included is also a list of 'system' channels that are initialized by the rendering system particularly for that shader, or by an object, which is typically evaluated before the shader. In addition to these, other channels may also contain initialized information written by user defined special operations, materials, plugin classes or by other means.

Material Initialization

This is a special shader in that it is executed only once; at the beginning of rendering. Therefore, it is suitable for initializing constant data, which is stored in root level variables.

Initialized channels:

Time	The global 0..1 time

Common Properties

The VSL code included in this shader is always executed with all other shaders of the material. You can place general-purpose operations and preparations that affect all other shaders into this shader. A typical use of this shader is defining the scope of a material - how much the material affects the target object.

Note, that the position of the common properties shader in the VSL hierarchy defines its evaluation time: if it is placed above other shaders, it is evaluated first. If it is at the end of the VSL tree, the code is evaluated as the last action of every other shader.

If the material contains no other shaders as the common properties shader, the evaluation defaults to the surface properties shader.

Initialized channels:

Scope	The channel defining how strongly the material affects the targets.
Map coords	Mapping coordinates is the usual parameter for texture maps, etc.
Time	The global 0..1 time.

Output channels:

Map coords	You can compute a global change of mapping coordinates in this shader.
Scope	Scope defined in this shader affects all other shaders.

Surface Geometry

This shader can be used to modify surface geometry related attributes.

Initialized channels:

Ray	The direction from which the surface point is looked from.
Distance	The distance from the ray source (camera, etc.) to the surface.
Coordinates	The point of the surface being examined.
Time	The global time.
Normal, U direction, V direction.	Local surface orientation
Map coords	Mapping coordinates.
Map x, Map y, Map z	Mapping orientation.
Antialiasing	Also computed by most mapping objects.

Channels used by the rendering system after the shader:

Bump height	Modifying this channel creates bump map effects. The height is expressed as meters. Therefore the values are typically quite small.

Surface Properties

This is probably the most commonly needed shader, because the appearance of most objects is defined by their surface. Normally, properties such as color or transparency are defined in this shader.

Initialized channels:

Ray	The direction from which the surface point is looked from.
Distance	The distance from the ray source (camera, etc.) to the surface.
Coordinates	The point of the surface being examined.
Color	The default diffuse color (white) or object's color.
Time	The global time.
Normal, U direction, V direction	Local surface orientation.
Bump normal	Possibly modified (bumped) surface normal.
UV Coords	Local surface coordinates.
Map coords	Mapping coordinates.
Antialiasing	Information density computed by most mapping objects.
Map x, Map y, Map z	Mapping orientation.
All 'Exterior' sample channels that are initialized in the 'Volume properties' shader.	These channels describe the properties of the volume from which the ray hitting the surface arrives. For example, Optical thickness and user defined channels can be utilized for optical effects. For more information, see 'Volume properties' shader below.

Channels used by the rendering system after the shader:

Color	The value defines diffuse shading.
Bump normal	Modifying this channel creates bump map effects.
Ray	Used in various shading actions.
Reflection	Amount of mirror reflection.
Transparency	Amount of transparency.
Volume sampling	Values greater than or equal to one invoke volume sampling.

Volume Properties

This shader is used when examining interiors of objects. The shader is needed when creating fog effects, including volumetric light effects (fog glowing in light, shadows casted in fog). Volume sampling slows down rendering, and therefore there is a switch on render settings, 'Volume sampling', which must be enabled in order to let volume properties shaders to operate.

Initialized channels:

Volume:Distance	The distance between samples (=diameter of space area the sample represents).
Volume:Coordinates	The examined point.
Volume:Ray	The direction of the examined ray.
Volume:Time	The global time

Volume:Map coords, Volume:Map x, Map y, Map z, Volume:Antialiasing	The mapping specific information. See 'Surface properties' above.
Ray:Distance	The total distance ray travels in the volume.
Ray:Illumination	The result of ray tracing so far.

Channels used by the rendering system after the shader:

Volume:Color	The diffuse color of fog. This property reacts with lighting.
Volume:Illumination	The 'ambient' or 'self-illumination' color of fog.
Volume:Turbidity	Density of the fog. Defines overall visibility of volume effects as a function of distance.
Ray:Illumination	It is possible to leave the 'Turbidity' channel to zero and compute the volumetric effect directly to this channel.

Light Properties

This shader defines the properties of light emitted from a light source. For example, it is possible to define light falloff rate, color, additional light property channels (a light source casting gamma rays), etc.

Initialized channels (all in Light sample):

Distance	The distance from the light source to the illuminated point.
Coordinates	The illuminated point (NOT the position of a light source!).
Ray	The direction of the light ray (from the light source to the illuminated point).
Illumination	The color of light.
Time	The global time.
Map coords, Map x & y, Antialiasing	Mapping information (see 'Surface properties' above).

Channels used by the rendering system after the shader:

Illumination	The color of light. If black, no illumination is computed.
Distance, Coordinates, Ray	These will be used in shadow computations.

Surface Filtering

This shader controls the shadow creation. A ray tracer executes the shader when a light ray coming from a light source hits an object. The purpose is to find out if the material applied to the object lets the light go through and how the light changes. The decision is made by examining the transparency channel. If the color is white (1.0, 1.0, 1.0), the light continues in an unmodified state, whereas black (0,0,0) stops the light and causes a full shadow. Materials can define custom filter effects by computing suitable filter color values in the filtering shader.

A Ray tracer always executes the 'Surface properties' shader before 'Surface filtering' shader. Therefore, transparency defined in 'Surface properties' is conveniently used as a default value for 'Surface filtering'.

Initialized channels:

All Light sample channels that were initialized in the Light properties shader	See 'Light properties' above.

All channels of the Filter sample that were initialized in the Surface properties shader	See 'Surface properties' above.
Light:Color	Unshadowed, original amount of light.
Light:Illumination	The current amount of light (possibly reduced by other shadows).
Target:Antialiasing, Target:Distance etc.	Target refers to the surface receiving the shadow. Some channels initialized in the Surface properties shader of the shadow Target are available mainly for advanced antialiasing purposes.

Channels used by the rendering system after the shader:

Light:Illumination	The current amount of light. You can change it directly for custom effects.	
Filter:Transparency	The rendering engine computes the amount of shadow from this channel. White = No shadow.	
Filter:Fade	Also the fade channel contributes to the shadow transparency. Fully faded objects do not cast a shadow.	

Volume Filtering

This shader is executed to examine how a volumetric material shadows the light. The sampling density channel defines the accuracy of computation, just like in other volume effects. Furthermore, the 'volume shadows' option in render settings needs to be switched on. The transparency channel defines the amount of shadow. Transparency has the default value defined in the 'Surface properties' shader, which was evaluated when the light ray entered the volume.

Initialized channels:

All Light sample channels initialized in the Light properties shader	See 'light properties' above.
Light:Color	Unshadowed, original amount of light.
Light:Illumination	The current amount of light (possibly reduced by shadows).
VFilter:Coordinates	The examined point.
VFilter:Distance	The distance between samples (=diameter of space area the sample represents).
VFilter:Map coords, VFilter:Map x, Map y, Map z, VFilter:Antialiasing	Mapping specific information.
VFilter:Ray	The direction of the examined ray.
VFilter:Time	The global time.
VFilter:Transparency	Initialized to the value defined by Surface properties.

Channels used by the rendering system after the shader:

Light:Illumination	The color of light. Can also be modified directly.
VFilter:Transparency	The rendering engine computes the amount of shadow from this channel.

Begin Illumination

This shader starts processing surface illumination by light sources. It is executed once, and after that usually multiple Surface illumination evaluations follow. In this shader, you can initialize channels for collecting lighting information.

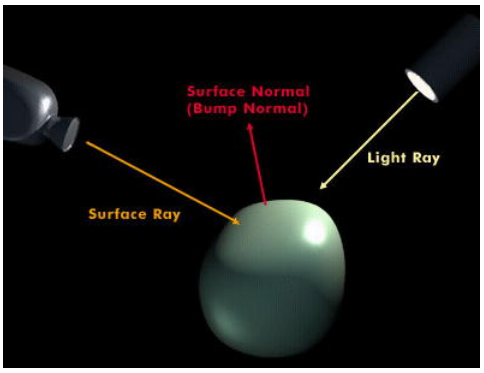
Initialized channels:

All Surface sample channels that were initialized in the Surface properties shader	See 'Surface properties' above.

Surface Illumination

This shader is executed for examining the influence of a light ray to a surface. This is a suitable shader for defining custom lighting effects. It is possible and also common to define illumination specific information already in the 'surface properties' shader, such as the usual diffuse color. Nevertheless, at this shader, the properties of the light ray are available (intensity, color, direction, etc.), and therefore, more detailed and advanced effects can be defined.

The default actions after the surface illumination is computing the diffuse shading using the color channel and adding the result to the illumination channel, which describes the total result of the shading.



Surface illumination

Initialized channels:

All Light sample channels that were initialized in the Light properties shader	See 'Light properties' above.
All Surface sample channels that were initialized in the Surface properties shader	See 'Surface properties' above.
Light:Color	The original amount of light sent by a light source.
Light:Illumination	The amount of light which reached the surface.

Channels used by the rendering system after the shader:

Light:Illumination	The amount of light.
Surface:Color	Diffuse lighting depends on this channel.
Surface:Illumination	The total result of rendering. Add custom illumination to it.

End Illumination

This shader finishes processing surface illumination by light sources. You can examine and store the total influence of light sources to the surface in this shader. For example:

```
Begin illumination /* Store illumination level before light source contribution */
    surface:my_lighting_channel = copy(surface:illumination)
Surface illumination
```

```
    .... /* Lighting actions here */
End illumination /* Compute how much illumination increased */
    surface:my_lighting_channel = substract(surface:illumination, surface:my_lighting_ch
```

Initialized channels:

All Surface sample channels that were initialized in the Surface properties shader and modified by Surface illumination shaders.	

Surface Finishing

This shader is the very last step of surface shading. In this shader, one can perform actions that control the total surface shading. For example, one can compute all illumination contributions to a buffer channel and then define the actual illumination using this shader.

Initialized channels:

All Surface sample channels that were initialized in other surface shaders	This is the last surface shading step, so most channel values have already been computed.

Channels used by the rendering system after the shader:

Illumination	Illumination channel contains the result of ray tracing.

Volume Illumination

This is similar to the surface illumination, except that this time, illumination of an interior point of a volume is examined. By default, ray tracer does the diffuse shading effect by the given color channel value: light makes the fog glow if the color channel is not black. The diffuse shading depends only on the intensity of the light in the examined point, it does not depend on the lighting direction.

Initialized channels:

All Light sample channels that were initialized in the Light properties shader	See 'Light properties' above.
All Volume sample channels that were initialized in the Volume properties shader	See 'Volume properties' above.

Channels used by the rendering system after the shader:

Light:Illumination	The amount of light.
Volume:Color	Diffuse lighting depends on this.
Volume:Illumination	The result of volume illumination. Add custom illumination to it.

Volume Shading

This shader is executed as the last step of volume processing. At this point, the Volume sample contains the fully illuminated fog in the 'Volume:Illumination' channel. The remaining step is the default shading, which blends the fog illumination with the result of ray tracing using the distance and turbidity. The thicker the turbidity and the longer the

distance, the more fog illumination affects the result. VSL code in 'Volume shading' can make the final adjustments before the default shading. It can also compute completely customized fog shading and then set the 'Turbidity' to zero to eliminate default shading.

Initialized channels:

Volume:Distance	The distance between samples (=diameter of space area the sample represents).
Volume:Illumination	The result of fog effects.
Volume:Turbidity	The density of fog.
Ray:Distance	The total distance the ray travels in the volume.
Ray:Illumination	The result of ray tracing before fog effects.

Channels used by the rendering system after the shader:

Ray:Illumination	The result of ray tracing before fog effects. You can compute a custom effect and blend it to this channel.
Volume:Turbidity	The density of fog. Defines the strength of fog effect. Set to zero to eliminate the default fog shading.
Volume:Illumination	The result of fog effects.

Initialize Ray

This shader is called when tracing of a new ray (reflection, GI ray etc.) starts. It is not called for rays originating from the camera. In this shader, you can move data from the originating surface into the traced ray.

Initialized channels:

Traced ray:Coordinates, Traced ray:Ray	The origin and unit direction of the traced ray.
Surface sample channels initialized by the surface properties and surface illumination shaders	



Note

If the ray is generated by a user defined raytrace object, the set of initialized channels depends on the location of the raytrace object in the VSL shader tree.

Channels used by the rendering system after the shader:

Traced ray:Coordinates, Traced ray:Ray	The origin and direction of the ray to be traced.

Secondary Ray

This shader is executed when tracing of a ray is finished and the remaining step left is to add the new information to the overall result of shading. For example, fading of reflection by distance can be done in this shader.

Initialized channels:

All Traced ray channels that were initialized by other shaders encountered while tracing the ray.	See the shaders above.

Surface sample channels initialized by surface properties and surface illumination shaders.	See 'Surface properties' and 'Surface illumination' above.

Channels used by the rendering system after the shader:

Surface:Illumination	The result of rendering so far.
Traced ray:Illumination	The result of 'child' ray tracing.

The default shading operation, which combines the two generations of rays, is simply
Surface:Illumination += Traced ray:Illumination*Signalcolor

Here 'Signalcolor' is the channel that created the new ray, such as transparency or reflection.

Ray Hit

This shader is quite a special one. The ray tracer executes it when computing distances to a surface. By modifying the distance, one can distort the surface. If the distance is set to a negative value, the surface disappears and so-called clip mapping effect is achieved.

Initialized channels:

Coordinates	The hit position.
Distance	The distance from the ray source to the hit position.
Map coords	Mapping coordinates.
UV coords	Surface coordinates.

Channels used by the rendering system after the shader:

Distance	Defines visibility of the surface.

Ray Termination

This shader can be used for some fine detail effects for transparent objects. The ray tracer executes it when the tracing of a ray is stopped because of finite recursion depth. Usually, one could add the average of the color of the overall scene to the illumination channel.

Initialized channels:

All surface sample channels that were initialized by the Surface properties and surface illumination shaders.	See 'Surface properties' and 'Surface illumination' above.

Channels used by the rendering system after the shader:

Illumination	Defines the result of tracing the current ray.

Scanline

This shader allows rendertime displacement of scanline objects using materials. The density of points which are passed for this shader depend on the particle type. For example, a particle curve evaluates point displacement densely along the curve, and this shader can be applied to make it curly. Particles may also support other control channels (diameter channel etc.)

Initialized channels:

Coordinates	The position of a geometry point on a particle.
Map coords	Mapping coordinates for displacement parametrization.
UV coords	UV coords run along the surface of each particle, describing its geometry.

Channels used by the rendering system after the shader:

Coordinates	Change this to achieve displacement.  **Note** Use 'Transform to material space' option when computing the displacement vector. Rendering space relative displacement will not behave consistently in animations.

Post Particles

This shader is executed when rendering particles in the post-processing module. For example, it is possible to define the color of particles in this shader. The purpose of this shader is similar to that of the 'surface properties' shader, but as the particle rendering in post processing is a much simpler operation than surface rendering in the ray tracer, less channels are available.

Control channels:

Color	The default diffuse color (white) or particle's color.
Fade	Controls transparency for most effects. Fade 1.0 makes particles totally invisible.
Coordinates	The position of the particle, all particles compute this. Change it to displace the particle.
Map coords, Map x, Map y, Map z	Mapping coordinates and orientation.
U direction, V direction, Normal	Additional information of particle geometry. See the particle specific documentation for details.
Time	The global time.

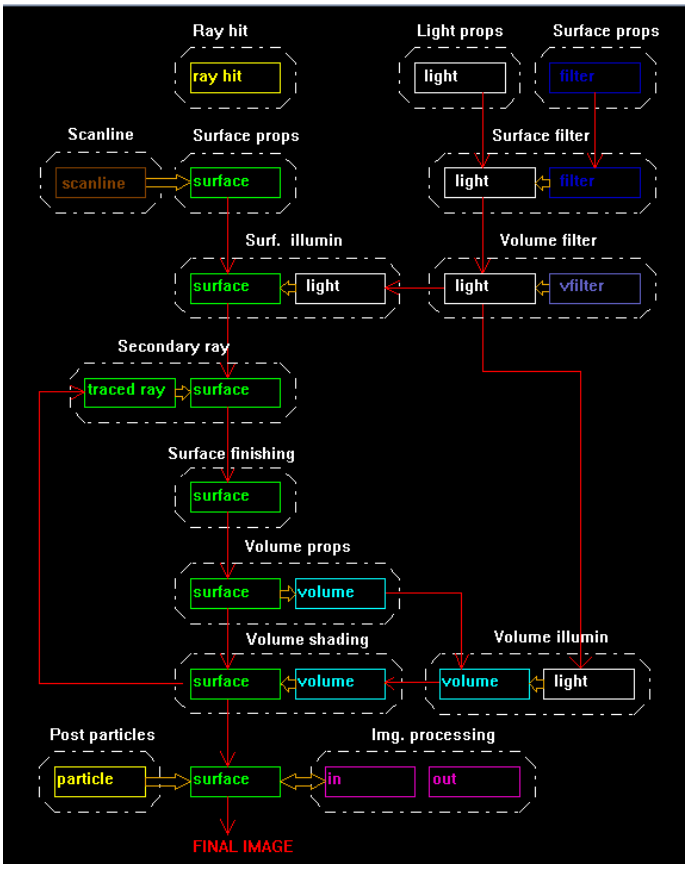
Image Processing

This special shader is plugged in by one post processing effect, 'material effect'. The purpose is to let the user manipulate pixels of the rendered image by a shader system. As described before, the material editor of the program includes a powerful programming language, and this shader allows the user to define custom image processing effects with the language. All the channels are available in this shader.

Input channels	All channels.
Output channels	All channels.

Rendering Diagram

The picture below shows how the rendering system moves data between the samples. Thick orange arrows represent internal default shading operations between samples, filling the gaps in the data flow.



VSL Objects

VSL objects compute and assign values to channels. A VSL object can be a mathematical formula, transformation curve or an algorithm, to name a few examples.

There are hierarchical, conditional and procedural objects available, as well as local variables. Also recursive structures are possible. The included object set provides the user with a visual programming language, which is specialized in creating algorithms for shading effects. There are tens of channels available, and an extensive set of objects with a variety of options; the number of possible combinations created using these elements is unlimited.

Most built-in and also user defined custom objects, if implemented properly, are 'reusable'. For example, a texture mapping object can be assigned to the transparency channel as well as to the color channel. Nevertheless, some objects have quite a special purpose: for example, the computations made by the bump map object are specifically designed for modifying surface orientation. One can assign the bump map object to the color channel, but the result hardly makes sense.

Bump

A Bump object uses a bitmap image to define a 'height transformation' to its output channel. This transformation is computed is such a way that, when assigned to the 'Bump normal' channel, the surface appears bumpy. The controls of a bump map are:

- *File name*: The name of the image file used for bump mapping.
- *Tile X* and *Tile Y*: These controls repeat the bump map pattern on the surface of the target object.
- *Flip X* and *Flip Y*: These options work in combination with the tile options. If Flip X is set, every second tiled bump area is inverted horizontally, so that the left and the right edges of adjacent tiles always match. Flip Y does the same in the vertical mapping direction.

- *Bleed X* and *Bleed Y*: Bleed option extrapolates edge values outside bump mapped area.
- *Interpolation*: Interpolation type defines the shape of the bumps. Linear interpolation produces sharp ridges and peaks, whereas Cubic produces very smoothly varying bumps. Bi-linear and Quadric are between these two.
- *Compute Direction/Height*: Compute Direction is the suitable operation mode when a bump map is used in a Surface properties shader for direct bump normal manipulation. Compute Height is designed for use in a Surface geometry shader.
- *Bump height*: Defines the height of bumps. In Compute Height mode, the measuring unit is meters, and therefore quite small values are often sufficient.
- *Smooth*: A smoothing filter. This feature eliminates pixelization of high resolution bump maps. The measuring units is the proportion

  radius of the smoothing filter/image width

  Therefore, quite small values are usually sufficient. Level 0.01 smoothing in a 500*500 pixel image uses a 10*10=100 pixel filter for smoothing.
- *Binding*: Binding controls are included in a separate tab when a bump object is selected. The tab displays a list of data channels, which are included in the selected image. By default, the first data sub channel (red in a usual color image) defines the bump height, but other channels can be selected as well.

Animation tab: This tab includes controls needed when using an animated image sequence for defining the bump map (or any other texture map type). This means that the program automatically loads a new bump map in every rendered frame.

- *Animation Type:* This option defines whether the bump map animation is on or off, and how the image index should be computed. The following alternatives are available:
  - *None:*Animation disabled. The first frame of an animation file or the specified image without index changes is used.
  - *Automatic:* The image index is computed using 'Repeat', 'Base index' etc. options automatically. By default, the full image cycle is used repeatedly.
  - *Key Frame:* The image index is computed from user defined key frames. Index does not change before a key frame sequence is recorded using the 'Current Index' field.
- *Current Index:* Shows the texture index which will be used in the current animation frame. If 'Key Frame' Animation type is selected, you can enter indices to this field in animation recording mode and hence key frame an arbitrary order for texture indexing.
- *Base index:* Defines, what is the index of the first image of the bump map sequence (usually 0 or 1). The value can also select the start of a sub range of a bump map sequence.
- *Frames:* Number of map images used for the animation (usually the total number of images in the map sequence). The images are selected starting from the base index. For example, consider a ten image sequence named as 'map0', 'map1', ..., 'map9'. If 'Base Index' is 3 and 'Frames' is 5, the index sequence will be 3, 4, 5, 6, 7, 3, 4, …
- *Repeat:* Activates repeated cycle of bump map images. First image is loaded again after the last defined image.
- *Ping Pong:* Activates 'ping-pong' style repeating of bump map sequence: bump index increases until the last index is reached, then it starts decreasing until the first index is reached, and so on.
- *Reverse:* Inverts the direction of bump map sequence.
- *Format Type:* Defines how names of separate images are indexed (a sample name is displayed below). When 'Default' indexing is set, the number of digits can be defined with the 'Digits' slider. When 'Custom' indexing is selected, any kind of indexing convention can be defined using the 'Format string' field.

- *Digits:* Number of decimal digits for image index. The value controls zero padding for small indices.
- *Format String:* Custom indexing format. The string follows the standard C language 'printf' conventions.



Note

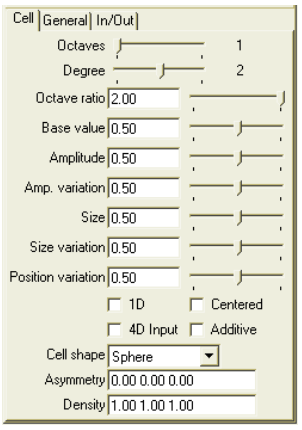
Automatic texture indexing changes the index in every frame, field and motion sample, so that texture animations are motion blurred as well. Other kind of behavior can be defined by animating the texture index by key framing. The choreography window makes it possible to modify the animated indexing in a highly flexible manner: the index curve can be point edited, the time line can be moved in time, scaled, repeated, etc.

Call

This object can be used to execute a user-defined VSL procedure (see also VSL Procedure Library). The Attribute tab of the Call object displays a list of available user-defined procedures for selection. After selecting the call target, the user can set the respective input and output parameters.

Cell

Cell noise is a versatile VSL object, which is especially useful in bump mapping. Cell noise is a combination of 'noise bubbles' in 3D space. It has the following options:



- *Octaves:* The amount of iterations in cell noise computation. The final cell noise is a sum of several iterations of cell computation. Every iteration has 'Octave ratio' times higher frequency and lower amplitude.
- *Base value:* Defines the base value of the noise signal.
- *Amplitude:* Defines the maximum range from the base value.
- *Amp. Variation*: The amount of random variation in amplitude between cells. 0 = All cells have equal amplitude, 1=Amplitude varies from zero to maximum amplitude.
- *Size*: The relative size of cells. The size 0.5 produces cells that do not overlap but slightly touch each other (if variations are zero).
- *Size variation*: The amount of variation in cell size. 0=all cells have equal size, 1=size varies between zero and the maximum value defined by *Size.*
- *Position variation*: The amount of random variation in cell positions. 0=all cells are in a regular 3D grid, 1=two adjacent cells may overlap completely.
- *1D*: The cell noise is one dimensional. 1D noise is faster to compute than 3D cell noise.
- *4D Input:* Selects four dimensional input for the noise. When the option is set, the cell object gets a second input channel, whose first sub channel is used for the additional 4th dimension. Typically the 4th dimension is animation time. This is useful for creating animated turbulent fog, fire and other similar phenomena.
- *Additive*: If set, the noise is sum of adjacent cells. Otherwise, the maximum value of adjacent cells defines the noise.
- *Centered*: If set, Amplitude varies symmetrically around the Base value. If cleared, Amplitude is added to the Base value.
- *Cell shape*: Sphere - spherical cells, Cube - box like cells, Rounded cube - Cell shape is a box with rounded edges.
- *Asymmetry*: Defines how symmetric the cells are. If cell shape is sphere and asymmetry is zero, all cells are spheres. Non-zero asymmetry produces elliptic cells. The value 1 makes the proportions of cells fully randomized.

The value of this attribute is a vector, and therefore variation of each cell dimension width/height/depth can be defined independently.

- *Density*: Scales the input parameter. The higher the density, the smaller (=denser) patterns the cell noise creates.

Comment

The purpose of this simple object is to add some optional information into the VSL shading tree. It does not perform any operation during rendering.

The comment string can be changed by a slow double click over the name string or by applying the Rename tool of the popup menu of the VSL tree gadget.

Constant

A Constant object assigns a constant value to its output channel. The type of its value is automatically chosen according to the output channel type.

Copy

This is a 'data mover' object. Copy object copies data from its input channel to the output channel. It is possible to choose which input sub channel is copied to which output sub channel. It is also possible to leave some output sub channels intact.

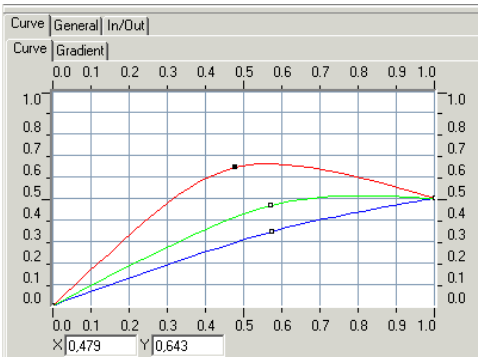
Curve

A Curve object allows the user to define an arbitrary transformation, defined by easily editable curves, between the input and output channels.

The number of displayed curves depends on the dimension of the output channel. A float output channel can be transformed by only one curve, whereas a color or a vector channel requires three curves, one for each sub channel.

Curves can be edited easily. The object automatically inserts a new point to the curve, if user clicks a curve in such a location where there is not yet a point. Existing points can be selected by clicking and multi-selected by shift-clicking or dragging. Points can be deleted using the popup menu of the curve gadget. For more information about curve editing, check the curve editor documentation.

There are five curve types available:



Editing a curve object. The red curve maps the first sub channel value 0.479 to a new value 0.643.

- *Linearly interpolated.* Points are connected using straight lines.
- *Boolean*. The y value of the curve changes only at control points.
- *Nurbs*. The standard cubic Nurbs curve.
- *PNurbs*. This interpolation method produces a stable curve also from badly oscillating data. Nevertheless, if interpolation data is bad, continuity will not be quite as good as when using regular Nurbs interpolation which is always C2-continuous.
- *Bezier*. The standard cubic Bezier curve. This curve type can represent both smooth sections and sharp angles.

The input and output range of the curve transformation can be adjusted using the menu 'Set minimum and maximum values' from the popup menu. The default range is 0-1. The value of the transformation outside the defined input range is defined by the values at the ends of the curve.

The Gradient option defines another kind of interface for curve editing: the result vector of a transformation is displayed as an RGB color. This is useful when designing color gradients and other RGB transformations.

Derive

Mathematically speaking, the object computes u and v partial derivatives of the norm of the output channel and assigns the computed two values to the output channel. Then it transforms the result to the surface orientation and normalizes the result. Practically speaking, these are exactly the steps that transform any function to a bump-map suitable form.

Usually, at least some of the objects use the same input channel (i.e. the channel with respect which the derivative is computed) as their parent 'Derive' object. Also, at least one of the sub objects should assign a value to the same output channel the derive object has (otherwise the derivative would be null).

The Derive object is mainly included for backward compatibility reasons. It has its use when computing a bump normal modification in the 'Surface properties' shader, but the same effect can be achieved in a simpler manner using a 'Surface geometry' shader.

Else

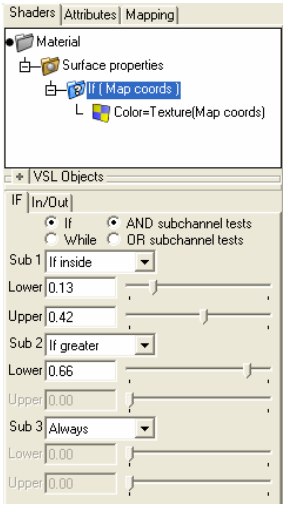
Else object defines a VSL level, whose sub objects become evaluated when conditions of a preceeding If object were not set. An Else object without an If object placed before it at the same hierarchy level does nothing (its children are never evaluated).

Else object does not have to be immediately after the If object: other VSL objects can be evaluated between the two.

If

This is the conditional VSL object. It is a 'level' type of object: other VSL code can be placed inside it. The object defines a condition, and sub objects are executed only when the condition is met.

If object can be paired with an Else object, that follows the If object in VSL hierarchy. Its contents are evaluated when If condition are not met.



If-object can be used in two ways: for single execution of sub objects ('If 'option selected) or for continuous execution ('While' option selected). The condition is defined for each sub channel separately. It has five alternatives:

- *Always true:* the sub channel does not set any condition
- *If inside:* true if sub channel value is inside a given range
- *If outside:* true if value is outside a given range
- *If greater:* true if value is greater than or equal to the given value
- *If less:* true if value is less than or equal to the given value

The channel sub tests are combined using a suitable logical operation:

- *AND operation:* The sub objects are executed if all sub channel conditions are true.
- *OR operation:* The sub objects are executed if at least one sub channel condition is true.

Level

The level object can be used in collecting VSL objects into hierarchical groups. A VSL level also defines a scope for variable objects inserted into it. A level object has no options.

Linear

This object computes a usual 'linear' transformation: it multiplies the input by a given factor and adds a constant value to it.

Map

Map object computes the mapping coordinate projection from the given parameter. The object is compatible with all material mapping objects which use a 3D geometry to define the mapping function. Surface defined coordinates (vertex variable interpolation etc.) are not supported.

One important use of the Map object is to provide a consistent parametrization for randomized sampling. For example, a volumetric fog shader may change the sampling position sligthly to examine the surrounding volume better:

```
volume properties
  vector variable pos
  vector variable mc
  vector variable delta
  delta = random(map coords)
  pos = add(coordinates, delta)
  mc = Map(pos)
  turbidity = Noise(mc)
  ...
```

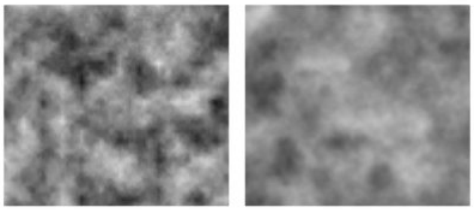
This kind of fog density definition appears consistent for a moving camera, self-shadowing fog etc.

Noise

Noise is a very useful object, which can be used in simulating various 'fractal' phenomena.

It has the following controls:

- *Octaves* defines the amount of iterations used in computing the fractal noise. The higher the value, the more detail the noise has (but it may take slightly more time to compute it).



Linear and Smooth 1D Noise assigned to the Color channel

- *Octave ratio* field defines the fractal dimension value: total noise = noise(input) + noise(input*dimension)/dimension + …
- *Base value* defines a base for the noise signal.
- *Amplitude* defines how much the noise values range from the base value.
- *Density*: Scales the input parameter. The higher the density, the smaller (=denser) patterns the noise creates.

- *Smooth* option computes smoothly continuous noise. This makes computation slightly slower, but the result is more suitable, for example, for bump mapping.
- *1D noise* computes one dimensional output values. If the output channel is a vector, all sub channels get the same value. 1D noise is faster to compute than 3D noise.
- *4D Input* selects four dimensional input for the noise. When the option is set, the noise object gets a second input channel, whose first sub channel is used for the additional 4th dimension. Typically the 4th dimension is animation time. This is useful for creating animated turbulent fog, fire and other similar phenomena.
- *Centered* If set, Amplitude varies symmetrically around the Base value. If cleared, Amplitude is added to the Base value.

Operation

This object offers a wide collection of mathematical formulas and functions, including the following (the first input is denoted as p1, the second as p2, etc.):

- abs. value - absolute value, inverts sign of negative values
- add - addition of two channels
- arc cos - usual trigonometric inverse functions
- arc cot
- arc sin
- arc tan
- ceiling - rounding up to integer value
- cos
- cotangent
- cross product - computes cross product of inputs, the result is perpendicular to both
- divide - p1/p2
- dot product - (x1, y1, z1) dot product (x2, y2, z2) = (x1*x2, y1*y2, z1*z2)
- exponent - exp(p1)
- faceforward - computes a vector which is parallel to p1 and faces against p2. In other words: v=faceforward(unit_normal, ray) means that v dot unit_normal = +-1 and v dot ray is less than 0.
- floor - rounding down to integer value
- invert - 1/p1
- length - the length of the input vector = square_root(x*x + y*y + z*z)
- logarithm - natural logarithm
- max - maximum(p1, p2)
- min - output = minimum(p1, p2)

- modulo - p1%p2 (floating point modulo)
- modulo by 1 - p1%1, the decimal part of p1
- multiply - p1*p2
- negate - -p1
- normalize - (x, y, z) - (x, y, z)/squareroot(x*x + y*y + z+z)
- polar - transforms cartesian coordinates to polar coordinates (heading, pitch, distance)
- power - p1 to the power of p2
- reflect - reflects a ray p1 with respect to a unit normal vector p2.
- refract - computes light refraction refracted ray=refract(exterior thickness, interior thickness, incoming ray, surface normal)
- round - rounding to the nearest integer
- sign - -1 if parameter is negative, 0 if zero, +1 if positive.
- sin
- sqrt - square root
- subtract - p1-p2
- tangent
- 1-(1-p1)(1-p2) - some useful formulas
- (x,y,z)-xyz
- 1-p1
- (1-p1)*p2+p1*p3 - linear interpolation
- (p1+p2)/2 - average of input channels
- (x,y,z)-(x+y+z)/3 - sub channel average. Can be used in color to gray scale conversion, etc.

Procedure

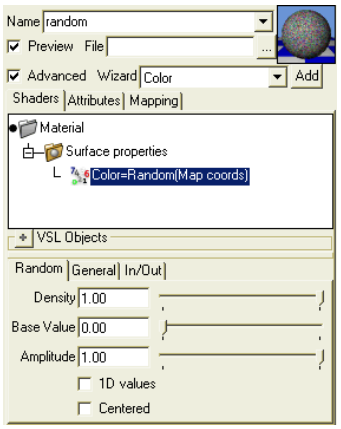
This VSL object defines a general purpose VSL function which can be called from other VSL code. Procedures can be created only into VSL Procedure Library objects. A procedure has the following properties:

- **Name** - A name for the procedure. You can set it to describe the purpose of the included VLS code. For example "Gamma Correction", "SuperNoise" etc.
- **Parameters** - Procedures cannot refer to any external data, all channel information must be passed in and out using parameters. The Parameter tab of the VSL editor has the following controls:
  - *Parameter List* - The complete list of procedure parameters.
  - *Add* - Adds a new parameter to the procedure.

- *Delete* - Removes the selected parameter. Note: If there is already VSL code calling the procedure, their Call parameters will not be automatically remapped.
- *Parameter Name* - It is a good idea to give descriptive names to parameters, especially if the procedure has many of them. You can then easily regognize the parameters later when defining calls to the procedure.
- *Variable Type* - The data type of the selected parameter. You can guide the program to use appropriate gadgets for parameters by setting the type. For example, if the procedure defines a color transformation, you may set its input parameter type to Color. Then for example Constant and Linear VSL objects show you a color gadget when you assign a value to the parameter.

Random

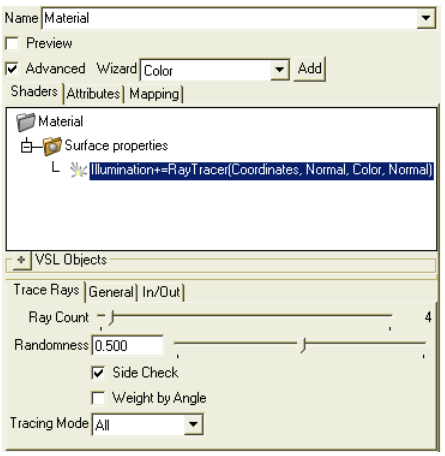
This VSL object computes random values. The following controls are provided:



- *Density* defines a probability threshold for the random computation. If the density is 0.1, only 10 % of evaluations produce a value other than the random base value. If the density is 1, all evaluations of the object compute a random value.
- *Base value* defines the value, which is output if the density threshold is not triggered. It also defines the base level for random values.
- *Amplitude* defines how much values range from the base value. If the base value is 2 and range is 1, the random values vary between 2 and 3 (when Centered is cleared).
- *1D values* option selects 'gray scale' output: all output sub channels get the same random value.
- *Centered* If set, Amplitude varies symmetrically around the Base value. If cleared, Amplitude is added to the Base value.

Raytrace

Raytrace object gives an access to the actual ray trace engine of Realsoft 3D. It is possible to create customized ray tracing effect using this object.



Some examples:

- Blurred reflections by sampling 5-10 rays around the reflected ray
- Global illumination by sampling some random rays around the surface normal
- Even fog can reflect the surrounding illumination. Each fog volume sample can trace some rays to examine what is around the position.

Raytrace object requires 2-4 input channels:

- input1: ray origin (usually 'Coordinates' channel)

- input2: ray direction (e.g. 'Reflected ray' or 'Normal'). Does not have to be a unit vector.
- input3: weight color (signal strength of the traced rays). The illumination value of the traced ray is multiplied with this value.
- input4: an optional half space selector for keeping randomized rays in the appropriate side of the defined direction. The 'Surface:Normal' channel is usually suitable for this purpose. Input4 is also used when 'Weight by Angle' is set, in which case the value should have unit length.

The raytrace object outputs the result of ray tracing which is usually added to the contents of the illumination channel.

The object has the following options:

- *Ray Count:* defines how many rays are traced.
- *Randomness:* defines how much the traced rays are scattered randomly around the defined main ray direction. 0=no randomness, 1=fully randomized ray direction (up to 180 degrees from given ray direction)
- *Side Check:* if set, the ray randomization is done so that the rays remain on a given half-space (e.g. on the same side of a surface). The side is defined by the parameter input2 (ray direction) and input4 (side selector)
- *Weight by Angle:* If set, the contribution of each ray is weighted by the dot product sampled_ray*input4. This is a useful option for illumination effects, input4 being the surface normal.
- *Tracing Mode:*
  - *All:* Performs full recursive ray tracing and stores the computed illumination to the output channel.
  - *Distance:* Finds the only the nearest ray hit and stores the respective distance to the output channel.
  - *Occlusion:* Stores the ratio *hit_rays/all_rays* to the output channel. The value can be used by Ambient Occlusion like illumination shaders.



Note

Occlusion and distance rays do not interfere with other shaders, which process or manipulate traced rays.



Note

Trace Range channel controls all tarcing Modes of the Raytrace object.

Shader

This object has the special task of dividing the execution of the VSL material into parts. Material properties are needed in various situations, and evaluating all properties every time would be slow. Therefore, the highest hierarchy level of a VSL material consists of shaders, which are levels under which the actual VSL code is placed. The type of shader defines when the included code is executed. More information can be found in the chapter 'Shaders'.

Shadow Map

Shadow map object is the same object which the render engine uses for mapped shadows. Render engine generates the required maps automatically as a part of the rendering process. With this object, you can do the same 'manually'. For example:

- Create a camera that matches the beam of a spot light

- Render an image file (preferably using Realsoft 3D's native multi channel format) that contains transparency and distance channels
- Map the image to a light source using the Shadow Map shader and a pyramid mapping which matches the camera geometry

Shadow Map has the following controls:

- *File* defines the name of the shadow map file.
- *Smooth* defines how much smoothing is applied to the shadow map. The value defines the proportion smoothing width/image width, so quite small values (0.01 - 0.1) usually suffice.
- *Bindings/Transparency channel* selects the image channel from which shadow transparency is read.
- *Bindings/Distance channel* selects the image channel which defines nearest shadow distance for each map pixel.

Specular

This object computes a specular illumination effect. It has two controls:

- *Sharpness* defines the size and sharpness of specular illumination effect
- *Brightness* defines the brightness of specular illumination effect

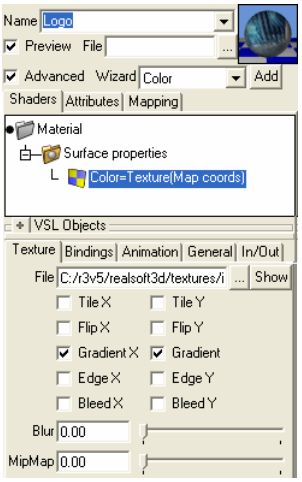


Note

Operation += is automatically selected when a specular shader is dropped to the VSL hierarchy. This is because of the additive nature of specular illumination.

Texture

This is the common texture map object. Itscontrols are:



Texture mapping in action

- *File name:* defines the name of image file.
- *Tile X* and *Tile Y*: these controls repeat the texture map on the surface of the target object.
- *Flip X* and *Flip Y*: If Flip X is set, every second tile is inverted horizontally, so that the left and the right edges of adjacent tiles always match. Flip Y does the same in the vertical mapping direction. The option can also be used without tiling to mirror texture maps.
- *Gradient X* and *Gradient Y*: Smooth interpolation between pixels. Boundaries of pixels of texture image become blurred, which in many cases improves quality.
- *Edge X* and *Edge Y*: By default, gradient interpolation is extended over the edge of the texture map to the background color. Edge X and Y eliminate gradient on the edges.
- *Bleed X* and *Bleed Y*: Bleed option extrapolates texture edge colors outside texture mapped area.
- *Blur*: A smoothing filter. The measuring units is the proportion

  radius of the smoothing filter/image width

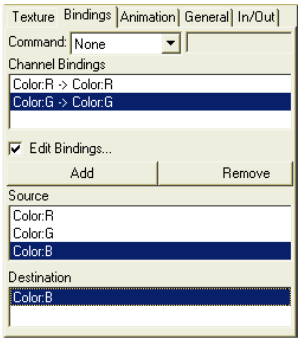
Therefore, quite small values are usually sufficient. Level 0.01 smoothing corresponds one percent of the image width which usually blurs the image quite a lot. Note, that 'Gradient' feature also smoothens the pixel boundaries, but smoothing uses only 2*2 pixel blocks. Blur smoothing is not limited, it can use e.g. 10*10 pixel area or more for blurring.

- *MipMap*: A special antialiasing technique for texture maps. If the value is greater than zero, the program computes a sequence of mipmap images. Each image halves the resolution of the previous one. The last image is only one pixel wide and contains the average color of the whole original image. The render engine uses interpolated colors from mipmaps whose resolution is suitable for the image area being computed. The higher the MipMap value, the more low resolution mipmaps contribute into the result.

The Bindings tab of the texture object gadget includes tools for specifying how to use the image data. It displays so called channel bindings, and allows the user to redefine them.

The Edit Bindings part of the tab shows the available image channels and the sub channels of the output channel. The existing channel bindings can be selected and then removed using the Remove button. New bindings can be created by selecting an image channel and an output sub channel and by pressing the Add button.

The Animation tab of the Texture object is the same as the Bump object has. See Bump documentation above for details.



Editing sub channel bindings

Variable

The 'Variable' object defines a 'local VSL channel'. A variable is visible everywhere within its parent level and in all the parent's sub levels. Variables can be used to store temporary results.

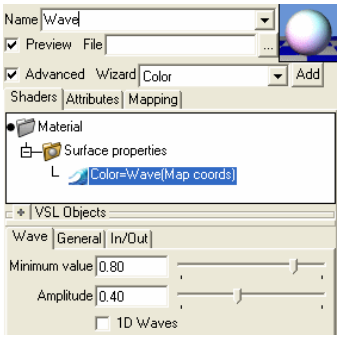
Variables have two attributes: name and type (float/vector/color).

The root level variables are different from other variables in that they are 'static'. This means that the storage for them is allocated once in the beginning of rendering and it does not change during successive evaluations of the material. This means that root level variables can be used to store constant data, which is initialized only once at the beginning of rendering (in a 'Material initialization' shader). Actually this also means that root level variables can pass data between shaders, but this capability should be used with care, because it may be difficult to ensure that the material is not evaluated again between writing to and reading from the variable. Ray tracing is a recursive process and the same material may be encountered again when following a reflected ray, etc. Therefore, user channels are a better alternative for this kind of purposes.

Wave

This object computes a trigonometric sine wave function. Its controls are:

- *Base value* of the wave. The output value on the bottom of the wave.
- *Amplitude* defines the height of the waves.
- *1D values* option selects 'gray scale' output: all output sub channels get the same wave height value, computed from the first input sub channel.
- *Centered* If set, Amplitude varies symmetrically around the Base value. If cleared, Amplitude is added to the Base value.



Color waves

VSL Wizards

The easiest way to create new kinds of VSL materials is to use VSL wizards. These wizards are available in the VSL editor. Wizards construct the necessary material structure automatically, and insert suitable controls for easy use.

Bump map	Creates a standard bump map that makes a surface appear bumpy. The controls available are: • Bump Height • Tile X: Horizontal tiling • Tile Y: Vertical tiling • Interpolation: Type of bumps. Linear=sharpest bump shapes, Cubic=smoothest bump shapes. • File name: The name of the bump map file
Chrome	The Chrome wizard creates a mirror like, shiny material. The only control available is the color of the material.
Clip map	This wizard creates the clip map effect. It is possible to define the color range, which removes the underlying surface. Inserted controls: • File name: The name of the clip map file • Sub channel 1 min & max: These two values define the clipped red signal range • Sub channel 2 min & max: These two values define the clipped green signal range • Sub channel 3 min & max: These two values define the clipped blue signal range Note that all the conditions must be satisfied before the clipping happens (red inside the red clip range AND green inside the green clip range AND blue inside the blue clip range)
Color	This wizard adds an object that controls the usual diffuse color. A new color value control is inserted for setting the color.
Density	This wizard controls material's mapping parameter density. Texture patterns, bump maps etc. effects can be scaled using it.
Fade map	The Fade Map wizard maps the inverted alpha channel of an image file to the fade channel: fade = 1 - image.alpha. The wizard adds some frequently needed texture map controls for tiling and pixel interpolation: • Tile X: Horizontal tiling • Tile Y: Vertical tiling

	• Grade X: Horizontal smoothing • Grade Y: Vertical smoothing • File name: The name of the texture file used for fade mapping.
Fog	This wizard creates a fog material that has transparent surface and constant density. The fog has 'plasma' behavior, in other words, its illumination does not depend on lighting conditions but is self illuminated. The fog attributes are: • Fog color: Defines the color of fog. • Fog density: Defines the density of fog. The higher the density, the more visible fog appears, and the more it hides objects inside and behind it.
Glass	The Glass wizard creates quite realistic glass. The surface of the material has a varying amount of subtle reflection, and the material also has a semi-transparent, 'caustic' shadow. Controls are: • Glass color: The color of the glass • Refraction: The amount of refraction. 0=no refraction, 0.4=standard glass, 0.9=very strong refraction.
Glow	Glow wizard adds two controls for defining glow: brightness and size. Glow brightness should usually match object color, and glow size, being measured as meters, is typically quite small (default 0.05). The glow effect must be included in the post effect configuration when using glow materials.
Illumination map	The Illumination map wizard creates a texture map object that defines object's self illumination. Note that usually illumination mapped objects need a black diffuse color; use the Color wizard to add it, if necessary. The wizard inserts the following texturing controls: • Tile X: Horizontal tiling • Tile Y: Vertical tiling • Grade X: Horizontal smoothing • Grade Y: Vertical smoothing • File name: The name of the texture file
Reflection	This wizard adds an object that controls mirror like reflectivity of the material. The reflection color can be set using the inserted color gadget. Black reflection color removes

	all reflections, whereas white reflection color produces chrome like, clearly visible reflections.
Self Illumination	This wizard adds a given RGB amount of self-illumination (object-specific ambient lighting) to the material.
SpecularColor	Inserts VSL objects for specular illumination effect. The controls are: • Sharpness: Defines how small and sharp-edged the specular illumination effect appears. • Brightness: Defines the maximum brightness of specular illumination.
Steel	This wizard simulates satellite metal, which is only partially reflecting and whose surface is slightly rough. The included controls are: • Metal color: Color of the material. grayish blue=steel, dark orange=copper, etc. • Roughness: Defines how smooth the surface is. • Sharpness: Sharpness of specular lighting effect • Brightness: Brightness of specular lighting
Texture map	The Texture wizard creates a standard texture map object that defines object's color. The controls available are: • Tile X: Horizontal tiling • Tile Y: Vertical tiling • Grade X: Horizontal smoothing • Grade Y: Vertical smoothing • File name: The name of the texture file
Transparency	This wizard adds an object that controls the transparency of the material. The level of transparency can be defined using the inserted color gadget. Black transparency color means full non-transparency, white transparency produces glass-like transparency. Note that usually the color wizard should be used in combination of this wizard, because glass-like objects have (nearly or totally) black diffuse color.

Output Objects

Output Objects define a rendering target, such as a file format. Such objects are needed in file rendering. The program includes various output object types, which have different properties: JPG allows good compression, AVI objects can store several consecutive frames, Targa supports alpha channel etc. The property window controls described below reflect these differences.

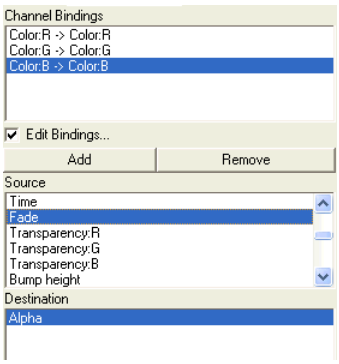
The common properties of all output objects are:

Name: A symbolic name for the object. It usually reflects some characteristic properties and the file format of the object: Jpeg good quality, Preview window, Targa RGBA etc.

Command: A script language or OS command line string that will be executed after the image data output to the object has been finished. The string can include any of the following tokens (square brackets must be included), which the program replaces with the actual used names before executing the command:

- *[file]* - The total file name, including path, possible frame index and extension.
- *[path]* - The path to the output file.
- *[base]* - The output file name without the path and the extension.
- *[ext]* - The file format extension without the dot separator.

The output object command feature can be used for automatic file format conversion using a command line based image processing program such as Image Alchemy.

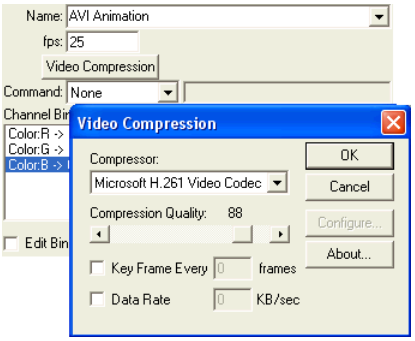


Channel Bindings: Bindings define what rendering data is placed into the channels of an output object. Each output object initializes reasonable binding defaults at the object creation. For example, the red, green and blue components of the total rendered illumination of each pixel go to the corresponding components of the usual color channel available in all output formats. This default behavior can be customized with the binding editor, which has the following controls:

- *Channel Bindings* - This list shows the current set of bindings. Items at the left side of each line are Realsoft 3D's data channels defined by the rendering modules and shaders. At the right side of the arrow is the sub channel of the image format to which the data is written.
- *Edit Bindings...* - Displays the binding editor where channel bindings can be changed.
- *Add* - After selecting an unused image format sub channel and a Realsoft 3D rendering data sub channel, you can press 'Add' to create the new channel binding.
- *Remove* - To remove an existing binding, activate an item from the 'Channel Bindings' list and click this button.
- *Source* - The list of all rendering data sub channels of Realsoft 3D.
- *Destination* - The list of all (not yet bind) image sub channels. When you bind an image sub channel, it disappears from this list. Deleting a binding brings the sub channel back to this list, so that it can be bound to another rendering data sub channel.

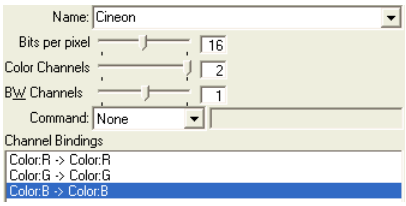
AVI

The AVI specific options are:



- **fps**: The preferred playback speed as frames per second.
- **Video Compression**: Opens a dialog where the preferred compression method and its settings can be selected. The compression types and settings available depend on the operating system configuration and other installed software packages.

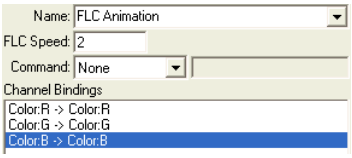
Cineon



The Cineon format has the following options:

- **Bits per pixel**: The data bandwidth as bits per sub channel.
- **Color Channels**: The amount of RGB channels (1 or 2).
- **BW Channels**: The amount of single component channels (1 or 2).

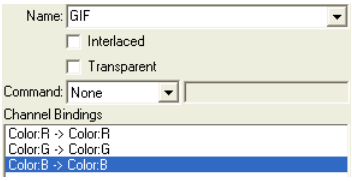
FLC



A FLC object has only one format specific option:

FLC Speed: The preferred playback speed as frames per second.

GIF



A GIF object has the following options:

- **Interlaced**: If set, the image data will not be stored linearly but in an interlaced manner.
- **Transparent**: If set, the black color will be set as the transparent color of the gif file. In later use of such a GIF image, pixels having the black color will appear transparent - the background shows through them.

ICO

The options of the Microsoft .ico format are:

Icon Type:

- 24 bit RGB - stores the colors
- 32 bit RGBA - stores the colors and the alpha channel
- Both 24 and 32 - this dual mode format includes both icon versions

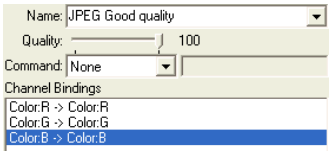
FFMPEG

The FFMPEG options are:

- **FPS**: Playback rate (frames per second)
- **Bitrate**: Data compression rate (kb/s). The higher the value, the better playback quality and bigger file size.
- **Format**: List of available formats, such as avi, mov or mpeg
- **Codec**: List of available codecs

Any codec can be used with any format but not all players support all format / codec combinations.

JPG



Jpeg files have only one format specific setting:

- **Quality**: Controls image compression and quality. The higher the quality, the bigger the output file size. High compression (=low quality) may cause bad errors in the image.

OpenEXR

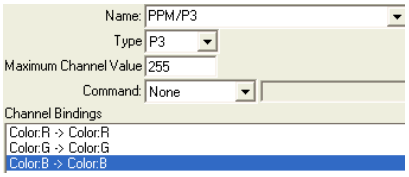
Any number of channels can be written to an OpenEXR file. To create an OpenEXR output, use the Add button to add suitable channels to the format, and rename the channels if necessary. All data types supported by OpenEXR are available: 32 bit integer, 16 bit float (so called Half type) and 32 bit float. Channel type can be selected per channel.

For convenience, Realsoft 3D automatically maps the common channel names as follows: Color.R is renamed as R in OpenEXR writing, and Color.G and .B respectively. Alpha is renamed as A. This helps other applications to identify the data.

PNG

The PNG format has only one format specific option: a switch for turning Alpha channel on/off.

PPM

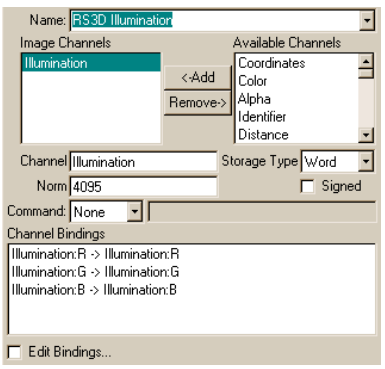


The PPM format has two special options:

- **Type**: the PPM type selector (P3=ascii, P6=binary).
- **Maximum Channel Value**: Defines the integer value which represents the norm signal level 1.0 (white).

Realsoft 3D

This is the native file format of Realsoft 3D. It supports arbitrary amount of channels, and each sub channel can use any of the standard data types from a single BYTE up to 64 bit IEEE floating point format. This format implements practically unlimited data bandwith. It is especially suitable for storing intermediate deep images which will be post processed later in high quality. The controls are:



- **Image Channels**: A list of channels that have been included into the image file format.
- **Available Channels**: A list of all rendering data channels available in the system.
- **Channel**: Name of the selected channel. You can give descriptive names to outputted channels, such as 'Floor alpha', 'Unshaded color' etc.
- **Storage Type**: Defines the data type of the selected image channel.

- **Norm**: Defines a conversion factor from floating point to integer form. The smaller the value, the wider value range the channel can store but with smaller precision. See the notes below.
- **Signed**: Enables/disables negative values for the channel.
- **Add**: Adds the selected channel from the system channels into the image format.
- **Remove**: Removes the selected image channel.



Note

For integer based data types (8 bit BYTE, 16 bit WORD, 32 bit INTEGER), the user can also define how floating point values are converted to the fixed point number form. The **Norm** value defines the conversion factor: integer value = float value*norm value. For example, for an usual byte based color channel, a suitable norm value is 255. Because this is also the largest value which 8 bits can represent, such a byte channel can only store float values between 0 - 1. Value -0.5 will be truncated to zero and 1.5 to 1. This is perfectly OK for the image color, but unsuitable for e.g. XYZ components of a surface normal vector.

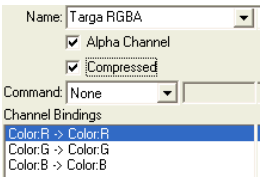
Another related control is the **Signed** option. Turning it on makes it possible to store negative values to the channel. Channels themselves define convenenient defaults for datatype management. Therefore, changing the Norm and Signed controls is seldom necessary.

Some examples:

- The default norm value of a word based color channel is 4095. Because the largest value which a 16 bit unsigned word can store is 65535, the maximal signal which you can store to the channel is 65535/4095 = 16. In other words, this kind of channel chan store 16 times brighter values as the standard white signal (1,1,1). It is also 4095/255 = 16 times more accurate as a byte channel. Summary: of the extra 8 bits of a word compared to a byte, 4 bits are used for the higher precision and 4 bits for the wider value range.
- If you wish to use a word channel to store the Z (distance) buffer and you know that the scene diameter is 100 meters, set the norm value to 65535/100 = 655. Then you do not need to rescale distance values at rendering time to prevent data overflows.
- The suitable byte options for storing a surface normal are: Signed enabled, Norm value=127. X, Y and Z components of an unit vector run from -1.0 to 1.0, so the value range available must be split between negative and positive values. For example, you can store the bump normal needed for Global Illumination processing to a byte channel without noticeable loss of quality and save lots of memory.

Targa

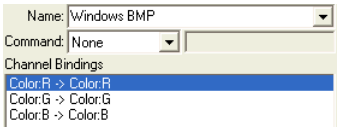
Targa is a popular output format in video industry because of its alpha channel support. It has two options:



- **Alpha Channel**: If set, an additional byte per pixel is allocated for storing the alpha values. Alpha channel usually contains the transparency information: value 0 = fully transparent, value 255 = fully non transparent.
- **Compressed**: Enables RLE compression, which can reduce the file size, especially when the image contains large constant color areas.

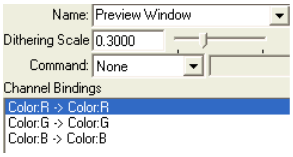
Windows BMP

Windows BMP does not have any format specific options. It is the native image file format of the Windows operating systems and therefore supported by most applications, such as Windows Paint.



Window

The Window output is exceptional in that it is the only output target that is not disk file based. You can use it for previewing file rendering progress in real time. It has one option:



- **Dither Scale**: Defines the amount of RGB dithering. The dithering value is scaled automatically to match the bit depth of the current display mode. The window output specific dithering operates on the already byte converted data stream, and therefore the dithering accuracy is maximally 1/255 units. Therefore, it is irrelevant for 24 bit display modes. The default value 0.3 is optimal for 16-bit graphics.



Note

You can use render settings dithering for high accuracy dithering below 1/255 unit boundary.

Post Particle Effects

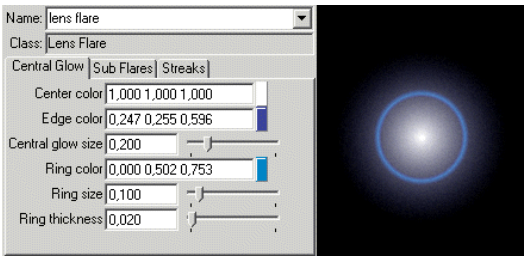
Lens Flare

The 'Lens Flare' effect simulates optical phenomena caused by bright light in a camera's lens system. Lens flare has many controls, which allow for a wide variety of different effects. Several program modules utilize this effect:

- **Lens flare particle effect** (described below in detail). Particles define the position of flares.
- **Lens flare of a camera object**. Light sources, which illuminate the camera, create lens flares.
- **Image Flare effect**. Flares are added to the brightest areas of the image.

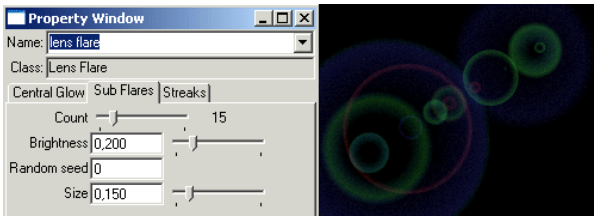
The lens flare controls are divided into three tabs.

'Central Glow' tab:



- *Center color:* The color of the bright glow in the middle of the flare.
- *Edge color:* The central glow is faded away towards the edge through this color.
- *Central glow size:* The diameter of the central glow. Value 1.0 is the full image width.
- *Ring color:* The color of the thin ring around the central glow.
- *Ring size:* The diameter of the ring (1.0 is the same as image width). Set to zero to remove the ring.
- *Ring thickness:* The thickness of the ring.

'Sub Flares' simulate internal lens reflections. The flares make a row passing through the center of the image.

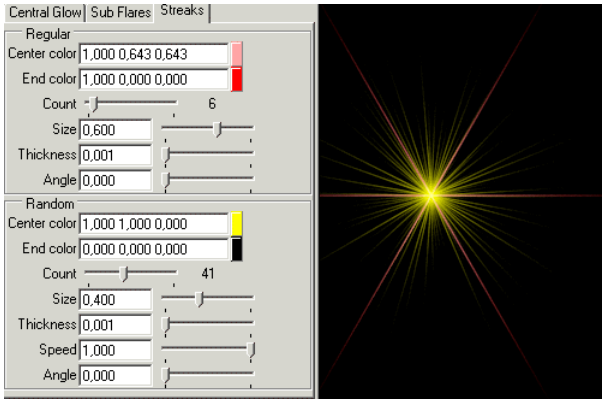


- *Count*: The total number of circular sub flares. Set to zero to disable this part of the effect.
- *Brightness*: Defines how bright the sub flares are. Usually low values 0.05 - 0.2 give the best results.

- *Random seed*: Selects a sequence of random colors and sizes for the flares. Each seed value produces a completely different looking flare sequence.
- *Size*: The maximum diameter of sub flare rings.

The Streaks tab controls drawing of stars. Regular streaks have the same length and they are drawn at regular angle intervals. 'Random' streaks have varying lengthsand random directions.

- *Regular & Random Center color*: The color of streaks in the middle of the flare.
- *Regular & Random End color*: The streaks fade away towards the end via this color.
- *Regular & Random Count*: The number of streaks.
- *Regular & Random Size*: The length of regular streaks and the maximal length of random streaks.
- *Regular & Random Thickness*: The thickness of streaks in the center of the flare.



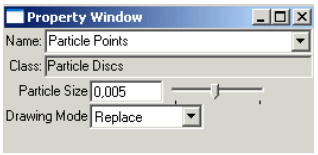
- *Regular & Random Angle*: Defines the orientation of streaks. Changing the value rotates the streaks (which naturally can be animated).
- *Random Speed*: The random streaks appear and fade away in animations, creating an impression of light radiation. This control defines how quickly the streaks change. Value zero freezes the random streaks to be static.

Particle Disks

This effect draws a filled circle to represent each particle. The size of all the disks is constant and defined by the effect. The colors of the disks are defined by the particles' colors. You could, for example, texture map the particles using a 'Post Particles' shader (see the VSL manual for details). The edges of the rendered points are always faded to full transparency in order to guarantee smooth blending with the rendered image. Assigning a suitable 'Fade' channel value to the particles controls the overall transparency of the disks.

The effect is simple and quick to render when the disk size is small. Therefore it is quite suitable for representing smoke, dust, fire and other particle effects which can be defined by a large flock of particles.

The effect has the following controls:



- *Particle Size*: The common diameter of all particles. The size is measured as real world units; typically a value such as one millimeter is quite suitable. The effect scales the size of particles by taking the perspective projection properties of the camera into account. Zero size has a special treatment: each particle is drawn by one single pixel, which is a very fast operation.
- *Drawing Mode*: Defines how particles are drawn into the image. The drawing modes are listed below.

The drawing modes are:



'Add' sums the color of each new particle to the previous result. It is a suitable mode for drawing 'glowing' particles, because many overlapping particles produce a very bright result.



'Max' uses the maximum color of the previous result and the new particle. This method also produces a relatively bright result.



'Min' mode uses the minimum color of a particle disk and the background image. Therefore, the particles darken the image.



'Mul' mode multiplies the particle and background image colors. This usually darkens the image.



'Replace' mode is the default mode. It is suitable for representing non-glowing effects such as fog, smoke or dust.



Note

Particle disk positions can be randomized using the following kind of material:

```
Particle displacer material
    Post particles
        Coordinates += Noise(map coords)
```

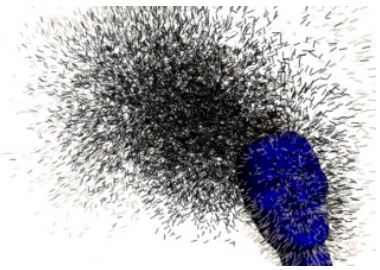
Particle Lines

This effect draws a line to represent each particle. The effect is therefore very suitable for rendering 2D particles (quickly falling raindrops etc.).

The thickness of the drawn lines is defined by the particle geometry. Color, fade and other rendering properties, as well as the drawing mode setting, are the same as in the Particle Points effect above.

The following channels can be modified using a Post particles shader to control particle line drawing:

- *Color* - particle color
- *Fade* - particle transparency
- *Coordinates* - start point of a particle
- *U direction* - end point of a particle
- *V direction X and Y* - two first sub channels define the radius values at the end points





Note

Mapping for the VSL material which assigns the above mentioned shader must appear before the line effect mapping in object hierarchy. Materials and post particle effects are evaluated in the order they appear in the select window.

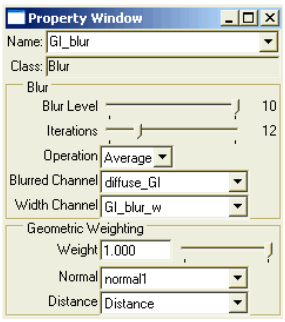
Post Image Effects

Blur Effect

The blur effect smoothes the image by blending adjacent pixels. This produces a smoother image and sometimes can be used to anti-aliase the image. For example texture flickering of insufficiently anti-aliased animations can be reduced this way.

The Blur effect can be controlled using the following settings:

- *Blur Level:* Defines the amount of blur. Increasing the value makes images more blurred.



- *Iterations:* Repeats the effect multiple times. Unusually wide blurs can be created quickly using high iteration values.
- *Operation:*The performed pixel operation.
  - Average - Pixel's new value is the average of the values inside the blur filter
  - Min - Pixel's new value is the minimum of the values inside the blur filter
  - Max - Pixel's new value is the maximum of the values inside the blur filter
- *Blurred Channel:* The image channel to be blurred. Usually the color channel is the most suitable target.
- *Width Channel:* An optional channel for controlling blur size. For example, to obtain selective blurring, the user can add a channel property to part of scene objects and then apply blur to only those parts of the image where the channel values are set. The total blur size is computed as the following product:

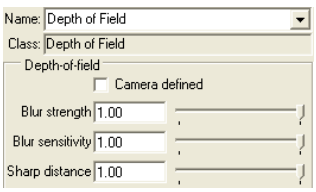
```
blur size = Iterations * Blur Level * Blur Width channel
```

- *Geometric Weight:* If set to one, blurring happens only across connected, continuous surface areas. Dislocated objects will not blur together. This is a useful option when anti-aliasing object specific channel information (such as ambient lighting) by averaging in post processing phase.
- *Normal:* The channel, which defines surface normal continuity.
- *Distance:* The channel, which defines distance continuity.

Depth of Field

The depth of field effect simulates image blurring of photographs. Cameras (except the pinhole camera) use a certain aperture for exposing the image to the film. Therefore, rays coming from different parts of the scene can expose the same point on the film. The greater the aperture, the more blur the photograph will contain.

The effect has the following controls:



- *Camera defined:* If set, the amount of depth of field blur is defined by the settings of the current camera (focal length, F-stop, focusing distance). Experienced photographers can estimate the amount of blur from the camera properties. Other users may find it more suitable to clear this option and use the other settings listed below to directly control the effect.
- *Blur strength:* Defines the maximum amount of blur when 'Camera defined' is off.

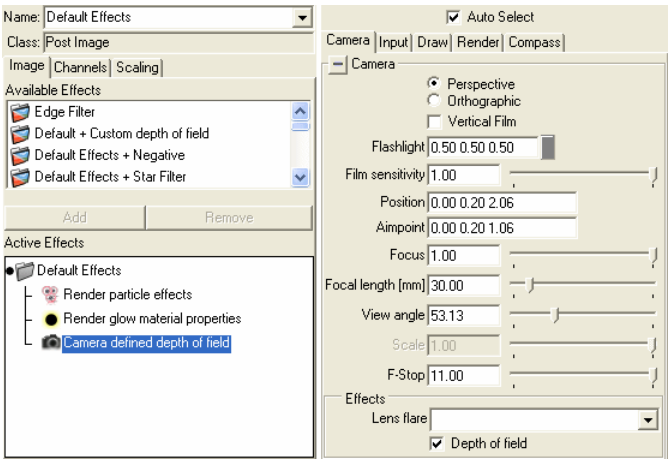
- *Blur sensitivity:* Defines how quickly the maximal amount of blur is achieved away from the sharp distance. Used only if 'Camera defined' is off.
- *Sharp distance:* The distance from the camera where the image is totally sharp (when 'Camera defined' is off).

Camera defined depth of field for view rendering

1. Go to the Post Image Effects tab. Drag and drop the post image you find suitable to the view window (for example 'Default Effects').

2. Open the property window to see the properties of the above mentioned post image. Make sure that a 'depth of field' object is included in the active effects. If not, add one to the active effects (you may first have to create a depth of field effect using the 'New' popup menu of the select window). Select the depth of field effect and verify from the property window that 'Camera defined' is set.

3. Open the view property window. Go to the Camera tab and set the Depth of Field option. Close the window and render the view. You can adjust the amount of blur from the F-Stop gadget of the view property window. The smaller the value, the more blur. Typical real world F-stop values range from 1.8 to 22.

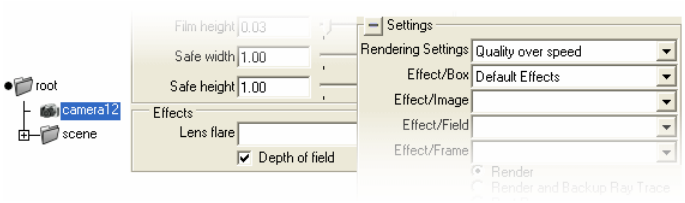


The post effect configuration includes Camera defined depth of field and the view has Depth of field enabled

Camera defined depth of field for file rendering

1. Go to the Post Image Effects tab. Select the post image you plan to use for file rendering (for example 'Default Effects'). Open the property window to see the properties of the post image. Make sure that a 'depth of field' object is included in the 'Active Effects'. If not, add one to the active effects. Select the depth of field effect and verify from the property window that 'Camera defined' is set.

2. Go to the Geometric Objects tab of the Select window. Make sure that there is a camera object included in the scene. If not, create one using the camera tool. Select the current camera object from the hierarchy (or all of them in case you are using multiple cameras for animation). Go to the Camera tab of the property window and set the 'Depth of Field' option. Close the property window.



The scene contains a camera, which has Depth of field option set. File rendering uses a post processing configuration Default Effects, which includes a Camera defined depth of field effect.

3. Open the file rendering window. Prepare settings as usual, but remember to select the post processing configuration which includes the depth of field effect. Render as usual.

Custom depth of field for view rendering

1. Go to the Post Image Effects tab. Drag and drop the post image you find suitable to the view window (for example 'Default Effects').

2. Open the property window to see the properties of the selected post image. Make sure that a 'depth of field' object is included in the 'Active Effects'. If not, add one to the active effects (create a new one using the 'New' popup menu of the select window if necessary). Select the depth of field effect and clear the 'Camera defined' option using the property window.

3. Render the view. Note that a strong depth of field effect may take a long time to render, so it may be a good idea to use 'Render/Define render box' menu of the view popup menu to render suitable test areas only. Adjust the 'Depth of Field' attributes from the property window and re-render (or post process only if backup buffer is enabled) until the result is suitable.

Note that you do not have to pay attention to the view property window's 'Depth of field' option in this case.

Custom depth of field for file rendering

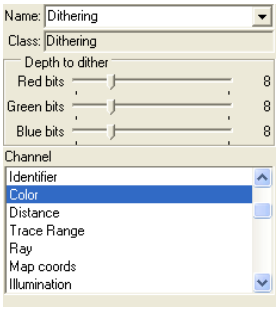
1. Go to the Post Image Effects tab. Select the post image you plan to use for file rendering (for example 'Default Effects'). Open the property window to see the properties of the post image. Make sure that a 'depth of field' object is included in the 'Active Effects'. If not, add one to the configuration. Select the depth of field effect and clear the 'Camera defined' option using the property window.

2. Open the file rendering window. Select the post processing configuration, which includes the depth of field effect prepared above, to the 'Effect/Box' field.

3. Render as usual.

Dithering

This effect adds dithering to the image. Most commonly used image standards use a limited amount of colors to represent images. The human eye is very accurate in detecting differences between colors and therefore even 16 million colors is not always enough. Dithering improves image quality by blending available colors.



The effect has following controls:

Red bits, Green bits, Blue bits: The suitable amount of dithering depends on the accuracy, which is used for storing the image. You can select the data type accuracy using these sliders. Value 8 is suitable for 24 bit output (16 million colors). Value 5 is suitable for 32 000 color output (16 bits).

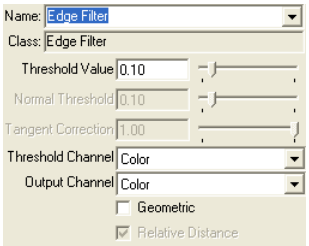
Channel: The effect can be used for dithering any channel. Color channel is the usual target of dithering, but you can select another channel from this list.



Note

View window has its own dithering mechanism and the ray tracer output can also be dithered before post processing starts (while the data is still in accurate floating point form). Therefore dithering post effect is seldom needed.

Edge Filter



The edge filter effect finds object silhouettes, edges and other areas where contents of the image change rapidly. The edge pixels are set black and other pixels white.

The controls of the effect are:

Threshold Value defines the amount of change between adjacent pixels which is required from edge pixels. The smaller the value, the more edges appear to the image rendered by the effect. A full value 1.0 usually generates an empty white image (when examining the color channel). In geometric filtering, the threshold value is applied to the Distance channel.

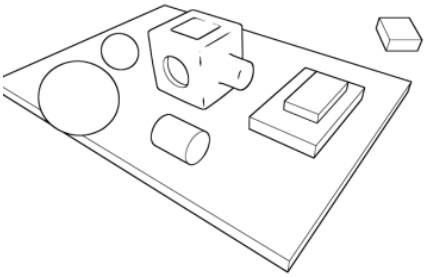
Normal Threshold is a second threshold which is used in geometric filtering. It defines the limit angle between two surface normals triggering edge drawing. Threshold value 1 corresponds to the 90 degree angle, 0 to zero angle. A suitably small threshold adds edges also to rounded but tightly bending corners.

Tangent Correction increases distance threshold of geometric filtering at low (tangential) viewing angles. The full correction value 1 is recommended for most scenes.

Threshold Channel selects the data channel, which is used for detecting edges. By default, color changes are examined. You can also use 'Distance' and 'Ray Normal' channels for detecting edges - or any other channel.

Output Channel defines to which channel the black and white result image is written. The default 'Color' channel overwrites the original shaded image. You can write the output to another channel and for example blend it with the shaded output using a VSL effect.

Geometric selects edge detection that is based on geometric continuity. Sharp angles and sudden distance changes generate edge pixels. Threshold channel will not be used.



Geometric outlines

Relative When enabled, distance discontinuity tests are made relative to the size of the scene. Furthermore, the threshold values are automatically adjusted by the output resolution and post image scaling. Distance and normal variations between two adjacent pixels are naturally smaller in a densely sampled high resolution image as in a low resolution preview image. This option automatically compensates the difference and helps to keep the result image similar across output resolution changes. Unfortunately identical detection cannot be guaranteed when resolution changes significantly.

Relative filtering is easier to set up, but the non-relative option provides more accurate control. For example, if a 5 mm thick plate on a table does not show up, reducing the Threshold to a value below 0.005 ensures that the edge will be detected.



Note

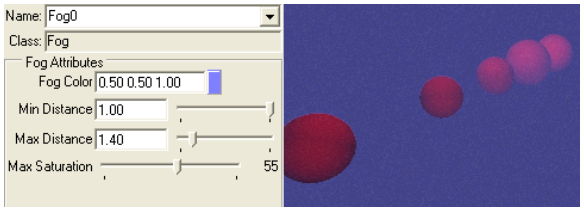
Surface normal based detection needs often some experimentation with the threshold value. An example: a small sphere viewed from a far distance may introduce significant normal direction change between every adjacent pixel. Edge filter which just sees the pixels cannot know if the surface is a 200 face polygonal sphere (=lots of edges) or a perfectly smooth round sphere (=no edges other than the round profile).

Fog

The Fog post processing effect provides a quick way to add global fog to the scene. The fog controls are:

Fog Color: This is the color of the fog.

Min Distance: Fog does not affect objects closer than this distance from the camera.



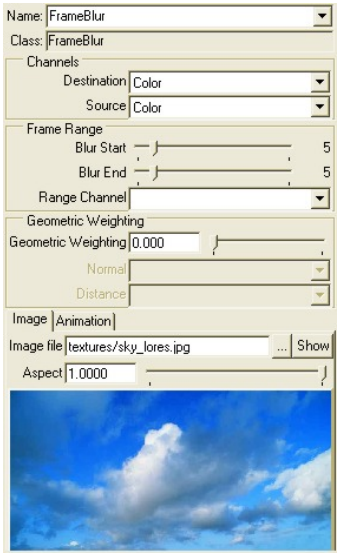
Max Distance: The distance from the camera which is needed for full fog effect. If saturation is 100, objects further than this distance are fully obscured by the fog.

Max Saturation: Defines how strongly the fog effect at its full level obscures objects. If the saturation is 100 %, the fog can fully hide objects. 50 % saturation blends maximally half of the fog color to the original image.

FrameBlur

The FrameBlur effect works similar to the normal Blur effect, except that it blurs over a sequence of images. This effect is useful for special effect purposes and for filtering image data over time.

The FrameBlur effect has the following controls:



- *Destination:* The post image channel to which the result is stored.
- *Source:* The image file channel to be blurred.
- *Blur Start:* Defines how many frames before the current image the blur starts. Value 1 starts the processing from the previous image.
- *Blur End:* Defines how many frames after the current image the blur ends. For example, if the current image index is 5 and Before=2, After=3, then the result will include the blurred result of images 3, 4, 5, 6, 7 and 8.
- *Range Channel:* An optional channel, which controls blur range. When a range channel is selected, the actual range is Before*range_channel ... After*range_channel. For example, if Before is set to 8 frames and the range control channel has a value 0.4 in a certain pixel of the current frame, blurring starts 8*0.4 = 3 frames before the current frame, provided that the respective pixels in preceeding frames have a range control value, which reaches the current frame. By setting the range channel to 0 or 1, one can obtain selective blurring i.e. only selected image areas will be blurred.
- *Geometric Weighting:* If set to one, blurring happens only across connected, continuous surface areas. Dislocated objects will not blur together. This is a useful option when anti-aliasing object specific channel information (such as ambient lighting) by averaging in post processing phase.
- *Normal:* The image file channel, which defines surface normal continuity for geometric blurring.
- *Distance:* The image file channel, which defines distance continuity for geometric blurring.
- *Image:* Here you can define an image file sequence (or avi file) from which the blurred data is read. The controls are the standard image object controls. If geometric weighting is active, the image file sequence must be in the native Realsoft 3D image format, and normal and distance channels must be included.



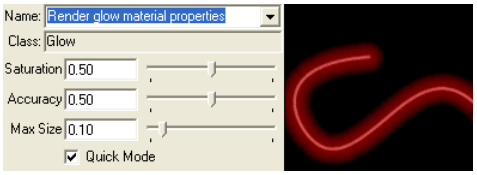
Note

The blur index range may exceed the actual source image range. The effect blurs the images it can find, and missing images do not cause an error. For example, at the end of the animation After indices usually refer to nonexisting images but that is perfectly OK.

Glow

The Glow effect is a part of the post effects 'Default Effects' configuration. It processes the image using two glow specific properties:

- Glow Size
- Glow Color

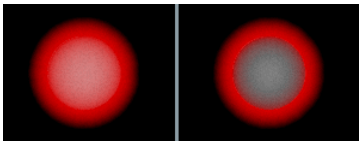


Glow effect

These can be defined just like any other material property such as color.

The glow effect examines the current scene and detects any use of the above mentioned channels. If they are not used, the effect is not executed. Therefore, you don't have to remove glow from the post processing configuration for speed reasons. You only have to remove it from the current post image in case the scene contains glow materials but you still want to disable glow rendering.

Saturation adjustment defines how quickly maximal glow brightness is reached in volume with respect to the distance. If the saturation value is zero, the glow looks brighter around the edges of the object - and has less intensity on the object itself, resulting in a kind of 'aura'. If the saturation is set to 1, an equally bright glow is generated in front of the glowing object as well as the edges.



Full saturation level on the left and zero saturation on the right

The **Accuracy** setting is an anti-aliasing type quality adjustment. The default value of 0.5 is suitable in most cases. The value range is from zero (fast, inaccurate) to 1.0 (slower but accurate).

The **Maximal Size** value defines a global limit for channel defined glow sizes. The global control can be used as an easy top level control for the effect, and to speed up the effect. Glow effect must usually render some additional image area outside the visible image to ensure effect continuity at image borders. The global size control instructs the program of what is the required extension area. The smaller the value, the quicker the rendering becomes. The default value is 0.1 (= 10 % of the image width).

Quick Mode uses a simpler solution to generate the glow from interior pixels of a glowing object. If the option is not set, every glow pixel generates a spherical 3D volume of a glowing matter around it, and such glows are blended into the final result. This is a very computation intensive task and takes long time to render. Quick mode uses a simpler approach, but usually the results are very similar - especially when glow size and color values variation is small.



Note

Suitable glow size values are typically quite small, for example 0.01 (1 centimeter) is pretty good for default scale models. Large glow sizes like 1.0 may take a long time to render.

When rendering an image with post effects to an animation, the rendering engine must expand the image internally. This increases rendering time significantly. If you only need to render one single still image of a scene containing many glowing objects, you can often increase rendering speed dramatically by switching 'Post Proc./Safety Area' and 'Distrib/Box Rendering' off from the render settings property window. Alternatively, you may apply glow as a full image effect, not a box specific effect.

Gradient

The gradient filter computes the image gradient i.e. intensity change derivatives. The gradient is useful for image analysis and for special effects.

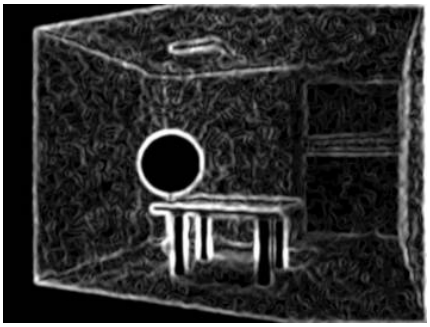
The Gradient effect has the following controls:

Brightness scales the brightness of the computed gradient data.

Input selects the data channel, from which the gradient is computed.

Output defines the channel, to which the gradient data is written.

Monochrome option outputs gradient data as a grayscale image representing the length of gradient values: sqrt(dx*dx + dy*dy). If the option is cleared, dx is written to R component and dy is written to G component. Note that the value range of dx and dy usually includes negative values.



The gradient of noisy global illumination

Image Flare

The image flare effect finds the brightest areas of the image and adds highlight flares to them. It uses the following principles:

- Only places where a given threshold value is exceeded get a flare
- Only one flare is added per a continuously connected bright area
- The size of a flare depends on the brightness of the area generating it

Brightness Trigger defines the threshold value, which must be exceeded before a flare appears. The value is defined as a sum of red, green and blue. A pure white color has brightness 3.0 and pure red has brightness 1.0, etc.

Max Flare Count defines the maximum number of flares added to the image. Only the defined amount of brightest areas will get the flare. This is the maximum amount, not the exact amount. If the picture is too dark, no flares will be added.

Trigger Channel selects the channel which triggers flares. Usually it can be the color channel.

The lower part of the image flare properties window contains the standard lens flare gadget. See the description of the Lens Flare effect for details.

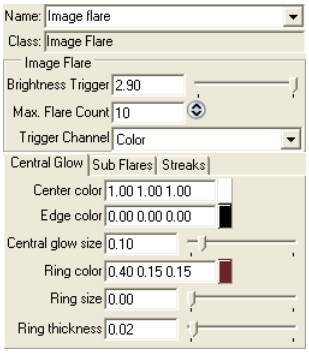


Image flare properties

The effect can extract highlight areas better, if you use 'Illumination' as the trigger channel. 'Illumination' value range is not clipped to 0..1 and its default data type is 'Word'. A byte can only represent a brightness scale from black = 0 to white = 1, whereas 'Word' channels can store values from zero up to 16. In other words, there are plenty of brightness values after white, and the program can locate the maximum brightness spots better.



Note

Do not use box rendering with the image flare effect, or apply image flare as an effect for full images. Render boxes break continuous bright areas and flare computations become inconsistent between boxes.

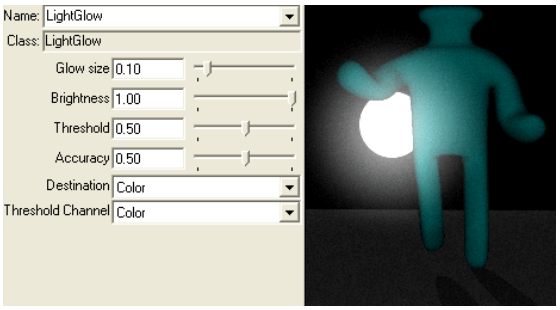
Light Glow

The Light Glow effect analyzes the image and generates an additive glow from bright image areas. Unlike the usual glow, light glow is a lens effect, meaning that it appears in front of all objects regardless of their distance from the camera.

Light glow has the following controls:
Glow size -

Defines how wide the computed effect is.
Brightnes -

Controls brightness of the overall effect. The intensity of the effect depends on the brightness of image areas generating the glow, and this slider provides a top level control to the whole effect.



Threshold -

Only image areas whose brightness exceeds this limit generate glow. Each sub channel is processed independently. If the 'Threshold' is 0.9 and pixel's illumination is red=0.5, green=0.5, blue=1.0, only blue glow becomes added to the surrounding area.

Accuracy -

Sampling accuracy for the effect. Values at 0.1 ... 0.5 range should work well for most cases. The full accuracy 1.0 can make the effect very slow, especially when the image contains thousands of pixels.
Destination -

The channel which glow changes. Usually should be 'Color'.
Threshold Channel -

The channel whose brightness triggers the effect. This can also be 'Color'. However, since color channel values are usually limited to 0..1 range (because of byte storage type), the 'illumination' channel which has a higher dynamic range, allows more interesting effects (with the cost of increased memory usage).

Line Effect

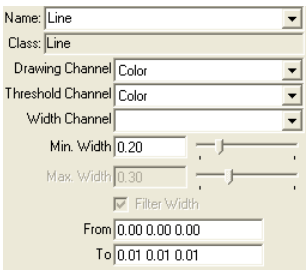
The main purpose of the effect is to filter outline rendering. For example:

1. Ray trace in double resolution and outline rendering mode on

2. Apply Line effect to get thicker lines

3. Scale the result down to 50 % resolution

The result of this process is antialiased outline rendering



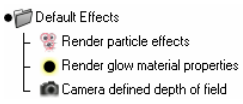
The following options are available:

- *Drawing Channel:* The channel to which lines are drawn
- *Threshold Channel:* The channel that triggers drawing
- *Width Channel:* An optional channel which can be used to achieve variable line width. The channel should have a value range between 0 and 1. Channel value 0 produces line width defined by the 'Min. Width' value and channel value 1 corresponds 'Max. Width' value respectively.
- *Minimal Line Width:* Line width values are expressed using percentage of the full image width. Therefore, small values such as 0.1 are usually suitable.
- *Maximal Line Width:* Used if width channel selected.
- *From* and *To:* Color range which triggers line drawing. For example, if you are rendering black outlines on a white background, the default range (0,0,0) - (0.01, 0.01, 0.01) is suitable. Then only almost black pixels will trigger the effect.

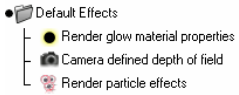
Particle Effect

This post processing object is actually not a real 'effect' - it does not render anything itself. However, it does the important job of managing the rendering of all post particle effects. The position of this object in the post processing configuration defines when post particle drawing takes place.

The configuration on the left shows a situation, where post particle effects are drawn as the first step of post processing. The right side image shows a configuration, which draws post particles last. Note that in the latter configuration, post particles cannot have the glow property.



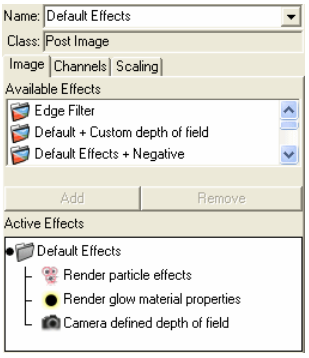
The particle effect has an option for distance sorted drawing. When enabled, particles are drawn in distance sorted order. Sorted drawing is slightly slower than unsorted drawing (typically a couple of seconds per 100 000 objects). Sorting is usually necessary when rendering multi colored particle flocks.



Post Image

The Post Image object has an important role in all post processing configurations: it allocates the image buffer, which other post effects modify. It also acts as a parent effect, having a number of child effects, which modify the parent's image buffer.

A post image can be configured by adding a set of suitable effects to its Active Effects list. The child effects can be added by drag & drop from the Available Objects list or by using the Add popup menu item. Effects can be removed from the Active Effects list by selecting them and using the Remove popup menu item. The child effects are rendered in the order they appear in the Active Effects list - sometimes this has importance.



A post processing configuration defined by a post image

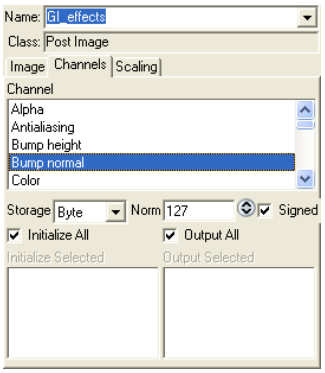


Note

A component of a post effect configuration can be disabled temporarily by selecting it and using the Disable popup menu.

The Channels tab of the post image property gadget contains a list of all installed channels. When you select a channel from the upper Channel list, the Storage field displays the data type, which is used for storing the channel values. Byte takes the smallest amount of memory space but is inaccurate. Float is the most memory hungry but very accurate. The most suitable data type depends on the channel - Byte is usually enough for storing the color, but never accurate enough for the distance channel and cannot represent a wide enough value range. The default types are suitable for most cases, but sometimes using higher accuracy is necessary (see the image flare documentation for an example of this).

Float data type usually gives fastest rendering times, because it is the render engine's native way of representing channel data. Use of other data types leads to data type conversions. However, due to high memory consumption, it is seldom possible to use the float data type for all channels.



Channel controls

For integer based data types (8 bit BYTE, 16 bit WORD, 32 bit INTEGER), the user can also define how floating point values are converted to the fixed point number form. The **Norm** value defines the conversion factor: integer value = float value*norm value. For example, for an usual byte based color channel, a suitable norm value is 255. Because this is also the largest value which 8 bits can represent, such a byte channel can only store float values between 0 - 1. Value -0.5 will be truncated to zero and 1.5 to 1. This is perfectly OK for the image color, but unsuitable for e.g. XYZ components of a surface normal vector.

Another related control is the **Signed** option. Turning it on makes it possible to store negative values to the channel.

All channels define convenenient defaults for these controls, so changing them is rarely necessary. Below are some examples:

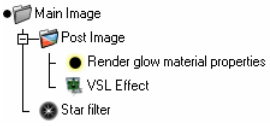
- The default norm value of a word based color channel is 4095. Because the largest value which a 16 bit unsigned word can store is 65535, the maximal signal which you can store to the channel is 65535/4095 = 16. In other words, this kind of channel chan store 16 times brighter values as the standard white signal (1,1,1). It is also 4095/255 = 16 times more accurate as a byte channel. Summary: of the extra 8 bits of a word compared to a byte, 4 bits are used for the higher precision and 4 bits for the wider value range.

- If you wish to use a word channel to store the Z (distance) buffer and you know that the scene diameter is 100 meters, set the norm value to 65535/100 = 655. Then you do not need to rescale distance values at rendering time to prevent data overflows.

- The suitable byte options for storing a surface normal are: Signed enabled, Norm value=127. X, Y and Z components of an unit vector run from -1.0 to 1.0, so the value range available must be split between negative and positive values. For example, you can store the bump normal needed for Global Illumination processing to a byte channel without noticeable loss of quality and save lots of memory.

You do not have to define which channels to include in a post image. The post image can find out which channels are used by child effects. It includes them automatically.

Post images can be used hierarchically. Each effect must be added to a post image, then the post image can be added to another post image to obtain the hierarchy. The example image shows a simple hierarchical post image configuration.



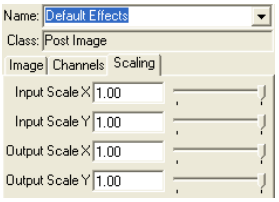
Both 'Main image' and 'Post Image' are post image objects. The sub image 'Post Image' allocates a new image buffer and glow effect is rendered into it. The VSL effect does some custom manipulations after that. The final step is star filter effect rendering into the main image.

If *Initialize All* option (see the channel tab image above) is set, the post image automatically reads the contents of all the channels being used by its child effects from the parent image. In other words, when the effect drawing of the sub post image starts, the situation is exactly the same as in the parent image. You can clear the Initialize All option and define manually which channels should be initialized in the beginning. Other channels will have zero contents. You can add channels to the Initialize Selected list by using the popup menu of the Channel list gadget. Removing channels happens by using the popup menu of the Initialize Selected gadget.

If the *Output All* option is set, all included channels are written to the parent image after sub effect rendering. To overrule this, clear the option and build a list of output channels to the Output Selected list.

Sometimes it is useful to clear the Initialize All option, but leave the initialization list empty. The effects will be drawn over a black background and the result can be, for example, added to the parent image. Anyway, it never makes sense to disable Output All and leave the output list empty - rendering effects without outputting the result somewhere just wastes rendering time.

The *Scaling* tab contains four values for scaling image data. Input scale X and Y rescale the incoming data, and Output Scale X and Y rescale the post processed data before outputting it further.



Examples:

1. If both input scales are 2.0 and output scales are 0.5, the post image doubles the resolution before the post effects are executed. Since these input and output scales compensate each other (2.0 * 0.5 = 1), the scalings have no effect outside the post image in question.

2. If input scales of the top level post image are 1.0 and output scales are 0.5, also the ray tracing resolution is doubled. The resolution drops to half after post processing. This can be used as an antialiasing solution which covers all post effects. Ray tracing antialiasing level can be reduced correspondingly.

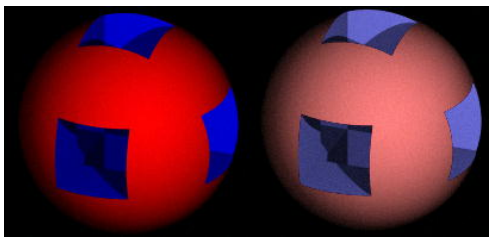
Note, that post images can be stacked hierarchically, and therefore you can change resolution between execution of each post effect.

The *Recommended Use* gadget tells how a post effect configuration should be applied: per box, per image, per field or per frame. The fifth alternative is Not Defined, which usually means that the effect can be applied per box or per image equally efficiently.

The usage recommendation provides information for the user, who is selecting a post effect configuration manually. Secondly, the automatic post effect selection feature at project loading uses the recommendation, when selecting the post effect configuration for each view window from a loaded project.

Saturation

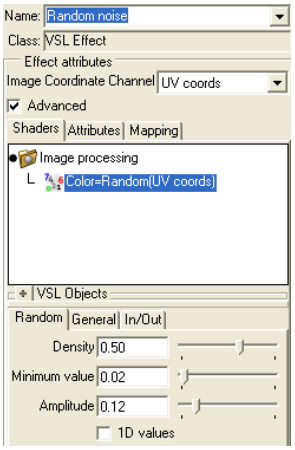
This effect changes the color saturation of images. It has only one property 'Color Saturation'. The lower the saturation level, the less colorful the processed image becomes. Full saturation 1.0 leaves the image unchanged, whereas zero saturation turns it into a gray scale image. Saturation has a maximum value of 1.0.



Saturation level 1.0 on the left, saturation 0.5 on the right

VSL Effect

The VSL Effect can be used for defining arbitrary pixel effects. Using the VSL shading language, the user can move data from a channel to another one, blend channels together, make color corrections, etc.



The **Image Coordinate Channel** defines the channel, which the effect uses to identify the coordinates of the processed pixel. The pixel coordinates are in floating point form: the top left corner has the coordinates (0,0,0) and the bottom right corner is (1,1,0). Use of relative floating point coordinates guarantees that VSL effects are resolution independent. When rendering boxes, the pixel coordinates are scaled to the range, which the box covers of the total image. Therefore, you can safely use box rendering with this effect.

The default image coordinate channel is the 'UV coords' channel. In the example image above, a random noise is added to the rendered image. The amount of noise depends on the UV coords.

The lower part of the VSL effect's property window contains the standard VSL editor. By checking the Advanced option, you get access to the editing tools. See the Materials chapter for VSL programming instructions.

The VSL objects of the image processing tree see all channels of the system and can freely perform any suitable computations using the channels. Because of the auto-configuration property of the post processing system, referring a channel from a VSL object automatically includes the channel in the post processing data flow.

Rendering Settings

A *Rendering Settings* object is a collection of rendering options. Realsoft 3D's photorealistic rendering engine is extremely powerful and designed for maximum realism and image quality. Correspondingly, rendering settings include a large number of options for controlling the entire rendering process.

Rendering settings can be managed through the select window's Render Settings tab. This tab shows the current library of predefined settings, and you can freely add new ones to suit your needs.

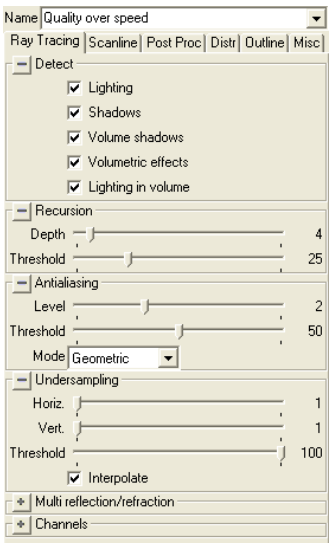
To create a new rendering settings object, select the 'New' pop-up menu. This creates a new object into the select window. As usual, you can use the property window to modify properties of the newly created object.

The rendering engine of Realsoft 3D consists of several sub-systems, such as the ray tracer, the post processor and scan line rendering. The first three tabs of the property window reflect this subdivision. Other tabs include general rendering options. All the available controls are shortly described below.

Ray Tracing

You can control ray tracing settings through the 'Ray Tracing' tab. Ray tracing is the most important part of the rendering pipeline. Accordingly, this tab contains many controls.

The controls are:



Ray tracing settings

Detect

All the options in this frame are set ON by default. The rendering engine examines the scene and switches off internally all unnecessary computations. Therefore, none of these options slow down the rendering, unless the corresponding feature is really used in the scene. These options can be temporarily switched off to speed up preview rendering, for example.

Lighting

If set, the rendering engine computes lighting information. By resetting this check box, one can disable all light sources in the scene.

Shadows

If set, the rendering engine computes shadows.

Volume Shadows

If set, object interiors with suitable volumetric properties cast shadows. For example, a fog cloud can cast a shadow. Not the surface but the actual 'material' inside the volume casts shadows.

This is an advanced feature. The *Volume Filter* shader of VSL can be used for defining shadow casting volumetric materials.

Volumetric Effects

This option enables computation of fog, smoke, gas, plasma, etc. effects (except volume shadows which has a dedicated control as described above).

Lighting in Volume

If set, influence of light sources to fog like materials is computed. Some foggy materials may not be affected by light (self illuminating fog), whereas others glow in light. The latter type requires the use of this option to render.

Recursion

The basic principle of ray tracing is to follow a light source as it travels in a scene, hits a surface, becomes reflected to a new direction, hits another surface, etc. This process requires time consuming computations and must be stopped after a reasonable amount of steps. Otherwise it may happen that the program ends up following a light ray bouncing between two mirrors forever and the image never gets ready.

The settings for controlling the amount of steps are:

Depth

This value defines the amount of surface hits after which tracing of a ray stops. If the value is one, no reflections are rendered. Because the tracing stops as soon as the ray from camera (observer's eye) hits an object. A reasonable

value range for photorealistic rendering is from 3 to 6. The minimal value that is required for a scene of an ordinary glass bottle is 5.

Threshold

Defines the minimal signal strength, which triggers a new ray. You can instruct the ray tracer to ignore very weak reflections or refractions not having much effect on the resulting image. This speeds up rendering. The threshold value is measured using current proportional unit. In the default per mill scale, the value 100 is 10% of pure white color. Low values improve image quality but also increase rendering time.



Note

When rendering transparent objects, this control can have a tremendous impact on the rendering speed. The ray is divided into two sub rays in each surface hit: the penetrating and reflected parts. Therefore, when recursion depth is increased, the amount of required computations grows exponentially, unless weak rays are terminated by a suitable threshold value.

Anti-aliasing

Ray tracer's anti-aliasing is a program feature for improving image quality. Without anti-aliasing, the ray tracer analyzes the contents of each pixel by computing one sample. Anti-aliasing can raise the amount of samples per pixel to ten or even more. Usually the additional samples are taken 'intelligently': the sampling rate is increased if the image data seems to vary a lot. This principle is called *adaptive anti-aliasing.*

Level

Controls the maximum amount of samples per pixel. The higher the value, the better the quality. Level 3 antialiasing is usually sufficient together with geometric anti-aliasing (see the modes below). Stochastic anti-aliasing may require one or two steps higher anti-aliasing level; level 4 is a good default for good quality rendering purposes.

Threshold

This value controls the adaptive nature of anti-aliasing. The ray tracer examines samples around the current sample. If the difference is higher than the threshold value, more samples are computed. Therefore, the smaller the threshold, the better the rendering quality. The value is expressed in the current proportional scale, which can be selected from the preferences window. In the default per mill scale, threshold 50 means that anti-aliasing starts if there is a color difference greater than (0.05, 0.05. 0.05) or (13, 13, 13) in 8-bit RGB scale. A good default range is between 50 and 100 per mill points.

Zero level threshold can be used in conjunction with high stochastic anti-aliasing (level 6) to force maximal antialiasing quality.

Mode

Geometric: additional samples are taken along a regular sub grid of the pixel grid. Samples are used to find the areas where changes happen. This is a good and fast way to anti-aliase images with reasonably low information density (clear geometric forms, etc.)

Stochastic: additional samples are taken randomly inside the pixel. Slower than geometric, but more reliable when the scene contains dense textures, thin objects, etc.



Note

Use stochastic mode for rendering motion blurred animations. Motion blur and stochastic anti-aliasing are designed to work well together.

Undersampling

Undersampling is quite a similar feature to anti-aliasing. It works as follows: A block of adjacent pixels is examined. If the color differences exceed a given threshold, the interior pixels are computed more accurately. Otherwise the color of the interior pixels is interpolated from the edge pixels.

This option is especially useful when rendering previews or high resolution images for printing purposes. Undersampling of 2*2 pixels is usually accurate enough to detect all details of a 2000*2000 pixel image and makes rendering 2-4 times faster.

Horizontal

The width of the pixel group. The higher the value, the faster rendering. This may reduce rendering quality, though.

Vertical

The height of the 'undersampled' pixel group.

Sampling Threshold

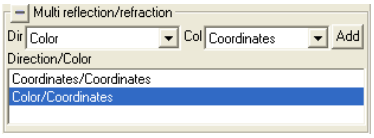
The minimal color difference in current proportional units (default per mill points), which triggers more accurate sampling. The smaller the value, the better image quality. A threshold value 50 is sufficient for reasonable quality.

Interpolate

Activates interpolation of pixel groups where the threshold is not exceeded. If the option is inactive, undersampling generates mosaic like effects (for example, you may try sampling threshold 100 with no interpolation).

Multi Reflection/Refraction

Using controls on this frame, you can create unusual or customized ray tracing effects. For example: some real world materials create a double refraction. A lens of such material shows two different images. You can simulate this (or triple/quad, etc. refraction if you like) in Realsoft 3D.



Defining multiple reflections

Use the controls in the following way:

- Select a channel defining the direction of custom rays to the 'Dir' field. This is usually a user defined custom channel.
- Select a channel defining the amount of custom reflection/refraction to the 'Col' field. This may also be a custom channel.
- Press 'Add' button.
- You can select and remove undesired direction/color pairs from the list using the popup menu.
- Create a VSL material, which computes the direction and the color to the selected channels in the 'Surface properties' shader. The direction must be a unit vector. Black color eliminates the effect and white enables the effect at its full intensity.

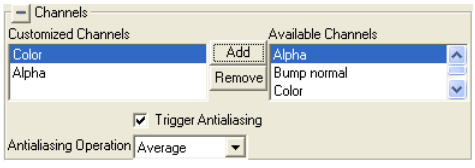
The example project 'tutorprojects/rendering/blurredreflections' shows an example of this feature. Four additional rays are cast randomly around the ideal reflection direction. This simulates a rough surface, which blurs reflections. Thanks to five reflection rays, anti-aliasing of blurred reflections is quite good.

Channels

The settings on this frame define data channel usage on anti-aliasing.

Trigger Antialiasing

By default, only color channel triggers antialiasing and subsampling. Here you can add also other channels that will trigger antialiasing. Especially alpha channel should be included whenever the alpha channel is outputted.



Anti-aliasing channels

Antialiasing Operation

Antialias operation selector. By default, all channels are averaged in antialiasing. Here you can change this to 'min' or 'max' operation. For example, setting the 'Distance' channel operation to 'Min' improves image quality when fog or depth of field post effects are used.

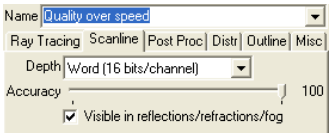
Scanline Rendering

Scanline rendering is the first step of the rendering process. Scanline particles are rendered to an internal 'meta image', which contains the properties of particles in a form which is suitable for the ray tracer. The ray tracer then combines scanline data to other scene data. Scanline rendering takes place before ray tracing and therefore scanline particles can be seen through glass objects, etc.

Scanline is a suitable rendering method for thin and small objects. Ray tracer can miss small details because of its 'sampling' principle.

Depth

Scanline particles define useful UV coordinates. This setting defines how accurately UV coordinates are stored per pixel. The default depth is 16 bit 'Word', but even 8-bit 'Byte' is sufficient for most cases. Low depth saves memory.



Scanline rendering options

Accuracy

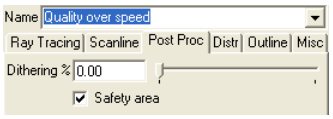
The accuracy of the 'meta image'. Values lower than 100 % can be used for previewing purposes or to save memory.

Visible in Reflections/Refractions/Fog

If this option is set, scanline rendered objects are considered in ray tracing computations. Clearing the option may speed up the rendering.

Post Processing

Post processing is the last step in the rendering pipeline. It has the following settings:



Dithering

A general dithering, which is applied to the accurate 64-bit/channel floating point output from the ray tracer. The human eye is so accurate that it can see banding artifacts in slowly changing color gradients when the usual 24 bit RGB image representation (= 16 million colors) is used. Dithering improves quality by removing the banding. A suitable amount of dithering for a 24-bit image is 1/256 = 0.4 % = 4 per mill points.

Note that some output targets (such as a view window) have their own internal dithering systems. In such a case, the additional dithering is unnecessary. This dithering feature is mainly used with file rendering.

Safety Area

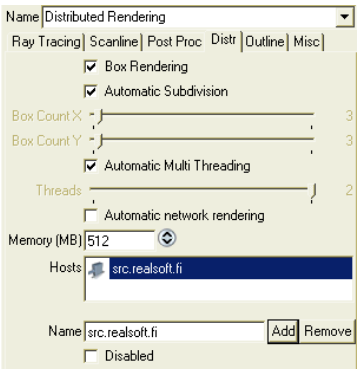
If set, the post processing system uses a larger internal buffer to detect how 'out of camera view' effects affect the visible image. This option has significance only when post effects, which spread their effect over the image area, are being used (glow, depth of field, post particles). For example, the center of a lens flare can be outside the view, but some of the streaks can reach the view. If *Safety Area* is not checked, streaks would not be rendered.

If box rendering is not used and all objects fit inside the camera view (during the whole animation), the option can be switched off. The speed increase can be quite significant.

Distributed Rendering

The options of this tab control distributed rendering. Realsoft 3D can take advantage of processor power available in the workstation in question (Symmetric Multi Processing) as well as other workstations accessible through the network.

Box Rendering



This option activates distributed rendering. If checked, the rendering engine subdivides the image to a given number of sub images. If the rendered scene is complex, this may increase rendering speed. Box rendering adds some overhead and therefore rendering of very simple scenes may be slightly slower than without boxes. Box rendering saves memory (especially when scanline particles and post effects are used).

Box rendering should be set whenever multiple processors or network rendering is used to accelerate rendering.

Automatic Multi threading

Enable / disable automatic thread count detection. If checked, the distributed rendering system will use all the available processors for rendering. The higher the thread count, the higher the memory consumption.

Threads

The number of threads to be used for rendering. For example, if you have a dual processor system, set this slider to value 2. Single processor systems should use one thread. Zero threads can be used if the 'Hosts' list is not empty. The whole rendering is then performed using other computers on the network.

Automatic Subdivision

Enable / disable automatic box count detection. If checked, the distributed rendering system computes appropriate box count automatically based on the number of threads and sub servers used.

Box Count X/Y

These two sliders define the number of boxes in horizontal and vertical directions. The total number of boxes should be higher than the number of threads and hosts used for rendering. If all the systems of the network are equally fast, the box count can match the sum of hosts and threads. (e.g. 6 processors - set Box X to 1 and Y to 6). Higher box counts help the system to adapt to speed differences between processors: slow computers render only one box while fast computers render many. However, box rendering naturally involves some internal managing work, and at some point increasing box count starts decreasing the speed.

Automatic Network Rendering

Enable / disable automatic network rendering. If checked, the distributed rendering system uses all available rendering servers in your local network.

Note that this option won't use the rendering server running on the same system as the actual client program. The reason for this is that the software does not need network rendering to take advantage of the local CPUs.

Memory

If *Automatic Network Rendering* option is used, this field can be used for specifying minimum memory requirement for the servers to be used. Only those servers which have more memory than specified by this field will be used.

Hosts

Host names or IP addresses of machines on the network to be used for distributed rendering.

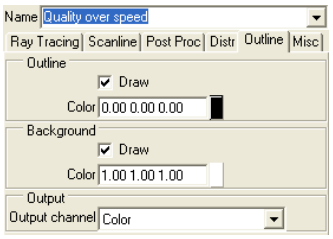
Remove/Add

These buttons allow you to add/remove hosts to the host list.

Disabled

Excludes the selected host from network rendering.

Outline Rendering



Outline rendering detects the edges (silhouettes) of objects and modifies the output accordingly. This feature can be used for technical representations, cartoon like rendering, etc. The following outline options are available:

Outline/Draw - Activates outline rendering. Edges become drawn with a given color.

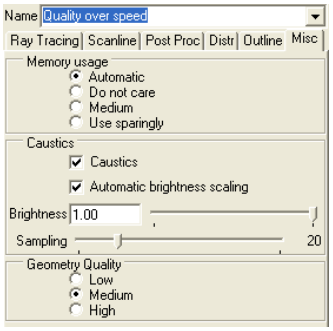
Outline/Color - Draw color of outline Edges.

Background/Draw - If set, interior of objects are filled with the given color. If cleared, the usual shaded surfaces fill the outline curves.

Background/Color - Background fill color.

Output channel - Defines the channel, to which the outline rendering is written. The default is color channel, which overwrites the usual shaded output. It is possible to direct the output to another (for example a user defined) channel and blend the outline and shaded output later using a suitable post effect.

Misc



Memory usage frame:

The rendering engine of Realsoft 3D uses RAM memory to store intermediate computation results and optimization data in order to speed up rendering. When running multiple applications, this kind of memory usage may be undesirable. Also, if the scene is so complex that the operating system must use virtual memory, using low memory rendering methods can actually increase rendering speed.

You can select one of the four memory usage alternatives:

- *Automatic* - Render engine examines the system and tries to select the optimal memory usage.
- *Do not care* - Render engine uses RAM memory without limitations.
- *Medium* - RAM memory is used in a reasonable way.
- *Use sparingly* - The render engine uses as little memory as possible.



Note

Automatic memory usage is the easy and recommended choice. However, it is impossible to predict the exact memory demand in advance. Therefore, it is possible that rendering speed can be increased

by forcing a particular memory mode instead of letting the render engine to estimate it. For example, if the automatic mode leaves lots of unused memory, 'Do not care' mode may be a faster solution. Or, if the automatic mode leads to intensive hard disk swapping, 'Use sparingly' mode may render faster. In a modern computer with several hundreds of megabytes of memory, this consideration is relevant only for complex scenes (50 000 surface items or more). The automatic mode is always optimal for simple scenes.

Caustics frame:

Caustics rendering system analyzes how light rays originating from light sources travel and scatter across the scene. Caustics rendering is automatically activated by geometric objects which have caustics map feature enabled. The settings of this frame control caustics rendering.

Caustics

Turns Caustics rendering on/off. It may be necessary to switch Caustics off temporarily for draft rendering purposes.

Automatic brightness scaling

If set, the brightness of caustics effect is calibrated so that the maximal point of brightness has the value defined by the 'Brightness' slider below.

Brightness

Controls the brightness of caustics effects.

Sampling

Defines how accurately caustics is computed. The higher the value, the longer caustics rendering takes, but the more accurate the result is.

photon map frame:

photon map stores indirect illumination by tracing photons and storing their surface hit positions with the carried illumination into a point cloud. Illumination in a surface point is then computed by finding and averaging the closest photons.

photon map illumination can be added to the scene in two different ways:

- By creating a photon map light source object. The object is placed into the scene hierarchy, and can be reused many times, or saved to a disk file. This approah is optimal for camera walktrough animations inside static scenes.
- By applying the dynamic Photon Mapping feature, which is an automatic preprocessing phase for ray tracing.

Photon mapping is controlled using the options listed below. Furthermore, Recursion Depth and Threshold of the Ray Tracing tab of render settings play an important role in photon mapping. Recursion depth should be set quite high, from 10 up to 50 bounces. Threshold should be 200 per mill units, because typical diffuse surfaces reflect less than 80% of the incoming light. Surface's diffuse color defines the main reflectivity, and the recursion threshold just adds a handy maximum limit. Thanks to it, you do not have to edit all colors, materials and textures of the scene to achieve a realistic diffuse reflectivity.

- *Auto photon map:* - Activates dynamic photon map computation, which is done automatically every time rendering starts.
- *Include Direct Photons:* - If set, direct photon hits from light sources will be stored into the photon map. Usually direct hits are not needed, because direct lighting can be evaluated more accurately using light ray tracing. Include

Direct Photons option is not sensible with dynamic photon mapping, because direct lighting will then be added twice. It may be useful for static photon mapping of area light sources.

- *KPhotons:* - The approximate number (counted as thousands) of photons that should be stored in the map. The higher the number, the more complicated and detailed illumination can be represented with the map. Complex scenes usually require several thousands of KPhotons. The amount of available RAM memory usually sets a limit. A rough rule of thumb is that 1 BG is enough for maximally 5 000 of KPhotons. The exact amount may be different, because other scene geometry and textures also need a highly varying amount of RAM.
- *Brightness:* - The brightness of the photon map illumination.
- *Sampling:* - The number of closest photons used for evaluating illumination in the examined point. Increasing the value reduces noise, but makes the illumination less detailed and also slower to render. A typical Sampling value for a high quality photon map is 100-1000.
- *Noise Reduction:* - A special preprocessing stage for filtering random noise of photon maps. High quality photon mapping requires fairly high noise reduction level, 50 or more. Very high levels, 100 - 1000, give sharp, low-noise photon maps, but preprocessing time can become very long. It should be noted that long preprocessing time often pays back later, because render time sampling rate can be set respectively lower, making rendering faster. Improved quality can be considered as an extra bonus. High Noise Reduction rate is especially recommended for static photon mapping in camera animated scenes, because the preprocessing will later save time in hundreds of rendered frames.

Geometry Quality frame:

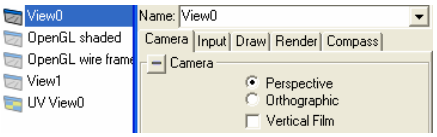
Many object types, such as SDS and NURBS, have adjustable render quality. The quality setting usually defines tessellation density for scanline drawing or accuracy of a numerical rayhit solving algorithm. The higher the quality, the smoother the surface appears.

The Geometry Quality option of render settings fine tunes object specific render qualities. For example, draft quality settings may use a lower accuracy to obtain faster rendering.

Medium quality renders with the actual quality value of each object. Low quality decreases object quality values and High quality increases the values. The actual amount of the relative change depends on the geometry type.

View Models

A View Model object contains all settings for a view window. You can assign a view model to a view by dragging and dropping it from the select window to the view. Each view window needs its own private view model. Therefore, if you assign a view model that is already in use to another view, the program automatically creates a duplicate and assigns it instead.



A view model object includes a large amount of options, which can be edited using the object property window and the view property window. For more information, see also Select Window/View Model Tab and View Window.

Animation Object

Animation object cannot be created directly. It is used by a number of other objects and provides a set of attributes for controlling timing. The attributes modify the local time and the effect propagates also to all the sub objects.

The object defines the following attributes:

Begin: Specifies the beginning of the object's time line.

End: Specifies the end of the object's time line.

Frequency: Scale factor for time. For example, setting Frequency to 2.0, will make the object run twice as fast.

Phase: Translation factor for time.

Periodic: Makes the time periodic, for creating loop animation effects.

Enable Time Curve: Enables the time deformation curve. If set, the time curve will be used for making objects local time non-linear.

Time Curve: The curve for deforming object's local time.



Note

If this feature is used, the key frame interface is no longer be able to show exact time values for the key frames!

Startup options

Realsoft 3D accepts the following startup options.

-tpaths [texture folders]

If a material refers to an image and the image is not found, these paths will be scanned to resolve the path. Useful if you have to render a project in another workstations. Multiple paths can be given by separating them with ; (semicolon).

Example: Realsoft 3D -tpaths "d:\mytextures;e:\moretextures"

-file [startup file]

Startup project.

-ccol [rrggbb]

Conflict color for the user interface controls representing conflicting attributes of multi selected objects.

-plugins [folder]

Specifies the folder from which the program reads plug-in modules. You don't usually have to specify this. The default values is 'Plugins'.

-lang [language]

This startup option allows you to select the language for the user interface. The value associcated with the switch refers to sub folders of locale folder.

For example:

Realsoft 3D -lang english

Attemps to read strings from the folder Realsoft 3D\locale\english.

-port [port number]

Distributed network rendering system uses the port 10050. If you have already used this port for something else, or if you can't use this port for whatever other reason (firewall), you can change it with this option (in which case remember to tell the port also for RendD!).

Example:

Realsoft 3D -port 10099

-btree

If given, the select window uses balanced tree layout.

-noguiload

If specified, user interface specific data sections are not loaded from project files.

-render If given, the program renders the given project and then exits. Only the current frame is rendered. This option is useful if you have to render several projects from a script file.

Example:

Realsoft 3D -render -file proj1.r3d Realsoft 3D -render -file proj2.r3d Realsoft 3D -render -file proj3.r3d

-play

This option renders the project as an animation.

Example:

Realsoft 3D -play myanim.r3d

-firstframe [frame]-lastframe [frame]

These options can be used with **-play** option to define the frame range to be rendered.

Example: Let's assume you have two workstations and you want to render the frames 0 ... 100 in the first workstation and the frames 101 ... 199 in the second workstation.

In workstation 1 enter: Realsoft 3D -play -firstframe 0 -lastframe 100

In workstation 2 enter: Realsoft 3D -play -firstframe 101 -lastframe 199

-forcetoscreen

If given, the program moves all windows to a visible screen so that the user can see them. You may need this option when reading projects that has been created on a multi monitor system.

-home [folder]

Specifies the home directory for the local user files. You don't usually have to touch this.

-nethome [folder]

If Network setup is used, executable files, manuals and other user independent files are installed on a server. This option specifies the home directory for these common program files. You don't usually have to touch this.

-debug

If specified, the program prints an error message whenever it finds a request it doesn't recognize. Can be useful for tracking old plug-in modules down, for example.

-prshost [hostname]-prsport [port]

These startup options control distributed network rendering. Servers to be used for rendering can be specified explicitely, by entering them to the Hosts list of rendering settings window. Or, you can ask the program to find them automatically from the host specified by these startup options. The host referred must run the PRS service.

-helpdir [folder]

This parameter can be used for defining custom path for the reference manual.

-html

If passed, the program attemps to use the html help manual rather than the native help manual.

-stcanvas

Enable single threaded rendering model. Helps to run the program with "beta" graphics drivers. help.

Scripting

Built-in JavaScript Classes

Axis Class

Realsoft 3D Axis.

Axis object consists of two attributes: position and direction. It can be used, for example, for representing vertices and associated vertex normals.



pos - vector defining origin (position) dir - vector defining direction For example:

```
o = new r3Vect(0.2, 0, 0);
d = new r3Vect(1, 0, 0);
a = new r3Axis(o, d);
```

Methods

Method: r3Axis - constructor

Syntax: a = new r3Axis()

Parameters: vPos, vDir - two vectors specifying origin and direction

Returns: new axis object

Description: Creates a new axis object. If no parameters are passed to the constructor, a null axis system is created (both attributes will be null vectors). If the number of parameters is two, then the constructor assumes that the parameters are vectors defining position and direction.

Example: // create an identity axis

Method: identity - identity

Syntax: a.identity();

Parameters: -

Returns: -

Description: Reset axis to identity state. This corresponds to the following code: a.pos.set(0, 0, 0); a.dir.set(0, 0, 0);

Example: cs.identity();

Method: normalize - normalize dir vector

Syntax: a.normalize();

Parameters: -

Returns: -

Description: Normalize dir vector to unit vector. This corresponds to the following code: a.dir.norm();

Example: a.normalize();

Method: print - dump values

Syntax: a.print();

Parameters: -

Returns: -

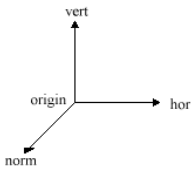
Description: Print current values of the axis object. Useful for debugging purposes.

Example: a.print();

Coordinate System Class

Realsoft 3D Coordinate System.

Coordinate system is analogous to Matrices in that it can be used for representing transformations and spaces.
origin - vector defining origin hor, vert, norm - three axes For example:




```
o = new r3Vect(0.2, 0, 0);
x = new r3Vect(0.1, 0, 0);
y = new r3Vect(0, 0.1, 0);
z = new r3Vect(0, 0, 0.1);
cs = new r3Coordsys(o, x, y, z);
```

Methods

Method: r3Coordsys - constructor

Syntax: v = new r3Coordsys()

Parameters: vOrigin, vHor, vVert, vNorm - four vectors specifying origin and axes for the coordinate system.

Returns: cs - new coordinate system object

Description: Creates a new coordinate system object. If you don't pass any parameters to the constructor, an indentity coordinate system is created. If the number of parameters is four, then the constructor assumes that the parameters are vectors defining oordinate system's origin, horizontal, vertical and normal axes.

Example: // create an identity cs ;

Method: identity - identity

Syntax: cs.identity();

Parameters: -

Returns: -

Description: Reset coordinate system to identity state. This corresponds to the following code: cs.origin.set(0, 0, 0); cs.hor.set(1, 0, 0); cs.vert.set(0, 1, 0); cs.norm.set(0, 0, 1);

Example: cs.identity();

Method: normalize - normalize coordinate system axes

Syntax: cs.normalize();

Parameters: -

Returns: -

Description: Normalize coordinate system axes to unit vectors.

Example: cs.normalize();

Method: print - dump current coordinate system values

Syntax: cs.print();

Parameters: -

Returns: -

Description: Print current values of the coordinate system object. Useful for debugging purposes.

Example: cs.print();

Low Level API

Scripting languages are linked to Realsoft 3D through a small set of low level functions corresponding to those used in the Realsoft 3D SDK 'C' API.



Note

The API described here is not indended for end users but for class implementators and other advanced users who need to get full control over Realsoft 3D.

Scrips can access Realsoft 3D through two variables:

- r3MainWindow - root of the user interface (main window)
- r3Model - root of the model (functionality).

These variables are actually 'pointers' which point to different type of objects depending on the application.

Functions

All the low level functions deal with Realsoft 3D objects rather than JavaScript objects. For sake of speed, there is no type checking. Passing wrong type of parameter to a function may crash the program. Type checking is only implemented by higher level JavaScript API.

Function: include - include a JavaScript file

Syntax: include(filename);

Parameters: filename - JavaScript file to be included

Returns: -

Description: Loads in given JavaScript file, if not already loaded. This function corresponds to #include directive of 'C/C++' and can be used for loading JavaScript files defining "proto type" objects (in Java or C++, these would be called 'classes').

Example:

```
include("oops/r3button.js");
```

Function: load - load a file

Syntax: load(filename);

Parameters: filename - the name of JavaScript file to be loaded

Returns: -

Description: Loads in given JavaScript file. Unlike include(), this function doesn't detect whether the file was already loaded. A file can be loaded any number of times.

Example:

```
load("myclasses/toolbars/mycreationtools.js");
```

Function: MakeID - Create an IFF identifier

Syntax: id = MakeID(a,b,c,d);

Parameters: a,b,c,d - four ascii values

Returns: 32 bit IFF identifier

Description: Creates a 32 bit IFF identifier from the given four 8 bit values.

Example:

```
var myid = MakeID('I','a','g','e');
```

Function: R3New - create a new Realsoft 3D object

Syntax: obj = new R3New(classid, taglist);

Parameters: classid - class identifier

Returns: obj - realsoft 3d object

Description: Creates a new Realsoft 3D object and returns a handle to it. The type of the object to be created is defined by 'classid'. Properties (attributes) for the object can be defined by one or more tag - value pairs.

Class id, methods and attributes for the object are defined in the object's class header file. All class header files can be found in 'scripts' folder.

Example:

```
// Create a sphere object
include("real/objects/r3sphere.js");

sphere = R3New(R3CLID_SPHERE,
               [R3RA_Name, "my sphere",
               R3SPHA_Center, new r3Vect(0.0, 0.1, 0.0),
               R3SPHA_Radius, 0.1]);
```

Function: R3Dispose - delete object

Syntax: R3Dispose(obj);

Parameters: obj - realsoft 3d object to be disposed

Returns: -

Description: Deletes the given Realsoft 3D object

Example:

```
// create a sphere
sphere = R3New(R3CLID_SPHERE, ...);
...
// delete the sphere
R3Dispose(sphere);
```

Function: R3Set - set object attribute

Syntax: R3Set(obj, id, value, type, flags);

Parameters: obj - address of Realsoft 3D object

Returns: -

Description: Set attribute. This function corresponds to 'R3SetAttrs()' function of 'C' API. The parameter 'object' refers to Realsoft 3D object. The 'id' is a unique identifier specifying the attribute to be set. The type of the 'value' parameter **must** match the type of the attribute to be set. There is no run time type conversion and the type of the attribute must be correct or program may crash.

Example:

```
// move the main window to x=10, y=20
R3Set(r3MainWindow, R3WGA_Left, 10, R3TID_INTEGER, 0);
R3Set(r3MainWindow, R3WGA_Top, 20, R3TID_INTEGER, 0);
```

Function: R3Get - fetch object attribute

Syntax: R3Get(obj, id, value, typeid, flags);

Parameters: obj - Realsoft 3D object

Returns: value - fetched attribute.

Description: Fetch object attribute. This function corresponds to R3GetAttrs() function of 'C' API. Typeid parameter specifies the type of the attribute to be fetched. Supported types are defined in 'oops/r3typids.js'. The 'byvalue' attribute indicates whether the attribute is fetched 'by value' or 'by reference'. This is needed to tell R3Get() whether it should allocate temporary buffer or not.

Example:

```
    name = R3Get(r3MainWindow, R3WA_Title, R3TID_STRING, R3TNS_COPYSTRING);
    print("Current window title is " + name);
```

Function: R3Do - call a method

Syntax: rc = R3Do(obj, mth, taglist) - call a method

Parameters: obj - address of Realsoft 3D object

Returns: rc - method specific return value

Description: Send a method to Realsoft 3D object. There are six different functions which corresponds to those used by SDK/'C' API. Because JavaScript keeps track of number of passed parameters, there is no need to terminate the tag list by 'R3TAG_END'.

Example:

```
// Ask the Realsoft 3D to find out its native size
R3DoA(r3MainWindow, R3WGM_FIT, R3WFP_BESTFIT);
```

Function: R3ProcessEvents - start event processing

Syntax: R3ProcessEvents(obj);

Parameters: obj - main window

Returns: -

Description: Start event processing for the given window. There can only be one event loop per application. If there is already an event processing loop running on the application in question, the function returns immediately. Otherwise the function returns when the event loop is terminated.

Example: R3ProcessEvents(window);

Function: R3Exit - stop event processing

Syntax: R3Exit();

Parameters: -

Returns: -

Description: Terminates event processing loop, which most likely causes the program to exit.

Example:

```
R3Exit();
```

Function: R3ClassFind - find class

Syntax: obj = R3ClassFind(classid);

Parameters: classid - class identifier

Returns: obj - class object

Description: Find class by given id. If the class is not registered, returns NULL.

Example: In Realsoft Graphics Oy's object oriented programming system, constructor is a class method. Creating a sphere means we call R3RM_CREATE method of the sphere class.

```
include("real/objects/r3sphere.js");

// find sphere class
class = R3ClassFind(R3CLID_SPHERE);
if(class) {
    // ask sphere class to create sphere instance
    sphere = R3Do(class, R3RM_CREATE,
                  [R3SPHA_Radius, 0.1]);
}
```

Function: Class - show class info

Syntax: Class(name);

Parameters: name - name of the object

Returns: -

Description: Show class specific info.

Example:

```
Class("CurrentProject.Geometrics.Root");
```

Function: Attrs - list public attributes

Syntax: Attrs(name);

Parameters: name - hierarchical name specifying the object

Returns: -

Description: This function attemps to find the given object and enumerate its attributes. The name is a hierarchical name which identifies an object in the application in question.

Example:

```
// Ask a sphere to enumerate its attributes
Attrs("CurrentProject.Geometrics.Root.mylevel.mysphere");
```

Function: Get - fetch attribute by name

Syntax: value = Get(name);

Parameters: name - hierarchical name specifying a Realsoft 3D attribute to be fetched

Returns: obj - attribute

Description: Fetches an attribute by symbolic name.

Example:

```
// fetch the addres of the project list object in Realsoft 3D
aProjectList = Get("");

// fetch the current project object
aCurrPrj = Get("CurrentProject");

// fetch the name of the current project
sName = Get("CurrentProject.Geometrics.Name");

// fetch the name of the Root level
sName = Get("CurrentProject.Geometrics.Root.Name");
```

Function: Set - set public attribute

Syntax: rc = Set(name, value);

Parameters: name = symbolic name identifying the object/attribute

Returns: -

Description: Set public attribute.

Example:

```
// rename the root level
Set("CurrentProject.Geometrics.Root.Name", "my new root");

// set the radius and center of 'mysphere'
Set("CurrentProject.Geometrics.Root.mysphere.Radius", 0.2);
Set("CurrentProject.Geometrics.Root.mysphere.Center", new r3Vect(0.2, 0.1, 0));

// set current animation frame
Set("CurrentProject.Animator.Frame", 30);
```

Function: Co - set Current Object

Syntax: ok = Co(name);

Parameters: name - symblic name

Returns: ok - true if succeeded

Description: Make the given object the current working object. Names passed to Get(), Set() and Attrs() functions are relative to the *current object*.

Example:

```
// make the 'root' level the current working object
Co("CurrentProject.Geometrics.Root");

// change the root's name
Set("Name", "my root level");
```

Function: Up - make the parent object the current object

Syntax: Up();

Parameters: -

Returns: -

Description: Make the parent object the current working object. Returns false if there is no parent object.

Example:

```
// make the 'root' level the current working object
Co("CurrentProject.Geometrics.Root");

// change the root's Translate attribute
Set("Translate", new r3Vect(0, 0, 0));

// pop up one level
Up();

// set root object's name
Set("Root.Name", "my root level");
```

Matrix Class

Realsoft 3D Matrix is a row dominated 4x4 matrix. Realsoft 3D uses matrices to represent spaces and transformations for geometric objects.

```
| a0  a1  a2  a3 |
|                |
| b0  b1  b2  b3 |
|                |
| c0  c1  c2  c3 |
|                |
| d0  d1  d2  d3 |
```

Matrix operations (such as concatenation, scale, translate etc. concatenate the new matrix on the right side. In other words, the latest operation is executed the first using 'first in last out' prinsiple.a0, a1, a2, a3, b0, ... d3 - 16 floating point numbers defining 4 x 4 matrix. For example:

```
m.a0 = m.b1 = m.c2 = 1.0; // reset scale transformation component to identity
```

Methods

Method: r3Matrix - constructor

Syntax: v = new r3Matrix()

Parameters: a0, ... d3 - sixteen floating point numbers

Returns: m - new matrix object

Description: Creates a new matrix object. If you don't pass any parameters to the constructor, an indentity matrix is created. If the number of parameters is four, then the constructor assumes that the parameters define a coordinate system's origin, horizontal, vertical and normal axes.

Example: // identity matrix;

Method: identity - identity

Syntax: v.identity();

Parameters: -

Returns: -

Description: Make the matrix identity matrix

Example: v.identity();

Method: set - set matrix attributes

Syntax: m.set(a0, a1, ... d3);

Parameters: a0, ... - sixteen floating point numbers

Returns: -

Description: Set matrix

Example: -

Method: translate - concatenate translation transformation to the matrix

Syntax: m.translate(x, y, z);

Parameters: x, y, z - floating point numbers specifying translation in x, y and z dimensions

Returns: -

Description: Concatenates given translation vector to the matrix.

Example: m = new r3Matrix();

Method: scale - concatenate scale transformation to the matrix

Syntax: m.scale(fX, fY, fZ);

Parameters: vX, vY, vZ - scale factors in x, y and z dimensions.

Returns: -

Description: Concatenates scale transformation to the matrix. vOrigin parameter may be given to specify scale origin. vOrigin defines a point which is translated to origin before the actual scale is applied. If no scale origin is given, [0, 0, 0] is assumed.

Example: m = new r3Matrix();

Method: rotate - concatenate rotation transformation

Syntax: v.rotate(fAngle, vAxis);

Parameters: fAngle - rotation angle in radians

Returns: -

Description: Concatenates rotation specified by angle, origin and rotation axis to the matrix. The rotation angle is defined in radians. If no origin is specified, [0, 0, 0] is assumed.

Example: // rotate about the 'y' axis

Method: rotate2v - concatenate rotation defined by two vectors

Syntax: v.rotate2v(vFrom, vTo);

Parameters: fOrigin - rotation origin

Returns: -

Description: Concatenates rotation specified by two vectors.

Example: // rotate 90 degrees about the 'y' axis

Method: skew - concatenate skew

Syntax: m.skew(fAngle, vFrom, vTo);

Parameters: fAngle - skewing angle in radians

Returns: -

Description: Concatenates skew transformation into the matrix. Skew transformation is defined in space defined by two vectors 'vFrom' and 'vTo'.



Note

The skewing angle must satisfy the following constraints:

```
-PI/2 < angle < PI/2
```

.

Example: // skewing space is x - y plane

Method: invert - inversion

Syntax: m.invert();

Parameters: -

Returns: -

Description: Invert the matrix. Assumes that the matrix is inversible.

Example: m.invert();

Method: mktran - make transformation matrix

Syntax: m3 = m.mktran(m2);

Parameters: m2 - matrix specifying target space

Returns: m3 - transformation matrix mapping points from 'm' to 'm2'

Description: Computes a matrix which maps a point from space defined by matrix 'm' to target space specified by 'm2'.

Example: mSrcSpace = new r3Matrix(o, h, v, n);

Method: transform - transform a vector

Syntax: m.transform(vTo, vFrom);

Parameters: vFrom - vector to be transformed

Returns: -

Description: Transform given vector. If two parameters are given, the method transforms the given 'vFrom' vector and assigns the result to 'vTo' (operation doesn't change 'vFrom'). If only one parameter is specified, the paramater acts as both source and target i.e. the result is assigned back to the given vector.

Example: // define a transformation matrix

Method: transforml - transform by 3 x 3 top-left sub-matrix

Syntax: m.transforml(vTo, vFrom);

Parameters: vFrom - vector to be transformed

Returns: -

Description: Transforms a vector by operating the parameter only with the 3*3 top-left submatrix, which is the linear part of the affine matrix operator. This means that you can use the transforml() method to transform vectors.

Example: // define a transformation matrix

Method: null - make the matrix null matrix

Syntax: m.null()

Parameters: -

Returns: -

Description: Sets all matrix components to null

Example: m.null();

Method: transpose - transpose matrix

Syntax: m.transpose()

Parameters: -

Returns: -

Description: Transpose matrix

Example: m.transpose()

Method: cmp - matrix comparison

Syntax: b = m.cmp(m2);

Parameters: m2 - matrix to be compared

Returns: b - boolean, true if matrices are identical.

Description: Compares the matrix 'm' against the given matrix 'm2' and returns true if the two matrices are identical.

Example: rc = m.cmp(m2);

Method: copy - copy a matrix

Syntax: m2 = m.copy();

Parameters: -

Returns: m2 - new matrix

Description: Creates an identical copy of matrix.

Example: m2 = m.copy();

Method: orthogonal - make orthogonal projection matrix

Syntax: m.orthogonal(fLeft, fRight, fTop, fBottom, fNear, fFar);

Parameters: fLeft ... - minimum and maximum clipping planes

Returns: -

Description: Concatenates orghogonal projection to the matrix. Transformation maps fNear to 0 and fFar to 1.0

Example: m.orthogonal(-0.1, 0.1, -0.1, 0.1, 0.1, 5.0)

Method: perspective - concatenate perspective projection

Syntax: m.perspective(fMinX, fMaxX, fMinY, fMaxY, fNear, fFar);

Parameters: fMinX ... - clipping planes specifying perspective projection

Returns: -

Description: Concatenate perspective projection to the matrix. Projection maps 'fNear ... fFar' to '0 ... 1'

Example: -

Realsoft 3D specific Classes

Realsoft 3D application consists of thousands of classes, which all can be accessed through JavaScript.

There is a JavaScript header file for each Realsoft 3D class. These so called JavaScript wrapper classes can be found in scripts/js folder. The folder structure corresponds to the structure of Realsoft 3D application.

There are three sub folders:

- oops - the kernel library (object oriented programming system)
- real - Realsoft 3D built-in libraries
- plugins - plug-in libraries

For more information about Realsoft 3D classes, see Realsoft 3D JavaScript Class Reference. You can find it from the Realsoft 3D CD. It is also online at realsoft.com [http://www.realsoft.com]

Vector Class

Realsoft 3D defines two separate vector classes for managing three and four dimensional vectors. Vector classes are built-in classes for sake of speed.Three dimensional vector class defines three attributes: x, y, z: Four dimensional vector defines attributes x, y, z and w.

Methods

Method: r3Vect - constructor for 3d vector

Syntax: v = new r3Vect()

Parameters: x, y, z, w - floating point numbers

Returns: v - new vector object

Description: Creates a new vector object. If you don't pass any parameters to the constructor, a null vector is created i.e. all the vector attributes are initialized to zero.

Example:

```
v = new r3Vect();
v = new r3Vect(0.1, 0,2, 0.3);
```

Method: add - addition

Syntax: v2 = v.add(v1);

Parameters: v1 - vector

Returns: v2 = result vector

Description: This adds the given vector to the object in question and returns the result as a new vector. The operation doesn't change the attributes of the object.

Example:

```
v1 = new r3Vect(0.1, 0, 0);
v2 = new r3Vect(0.2, 0, 0);
```

```
v3 = v1.add(v2);
```

Method: fadd - vector addition

Syntax: v.fadd(v2);

Parameters: v2 - vector to be added

Returns: -

Description: Vector operation v = v+v2.

Example:

```
v = new r3Vect(1, 0, 0);
v2 = new r3Vect(0.1, 0, 0);
v.fadd(v2);
```

Method: set - set value

Syntax: v.set(x, y, z); // r3Vect

Parameters: x, y, z, w - floating point numbers

Returns: -

Description: Set new value to the vector

Example:

```
v = new r3Vect();
v.set(0.1, 0.2, 1.0);
```

Method: sub - vector subtraction

Syntax: v3 = v.sub(v2);

Parameters: v2 - vector to be subracted from the given object

Returns: vector - the result of the subtraction

Description: Returns a new vector containing the result of vector operation 'v - v2'.

Example:

```
v1 = new r3Vect(1, 1, 1);
v2 = new r3Vect(0.5, 0.5, 0.5);
v3 = v2.sub(v1);
```

Method: fsub - vector subtraction

Syntax: v.fsub(v2);

Parameters: v2 - vector to be subracted from the given object

Returns: -

Description: Subtracts given vector 'v2' from the vector in question.

Example:

```
v = new r3Vect(1, 1, 1);
v2 = new r3Vect(1, 1, 1);
```

```
v.fsub(v2); // 0, 0, 0
```

Method: dot - dot product

Syntax: f = v.dot(v2);

Parameters: v2 - vector

Returns: f - floating point number

Description: Performs dot product between 'v' and given parameter 'v2' and returns the result. The operation has no impact to the object itself

Example:

```
v1 = new r3Vect(1, 0, 0);
v2 = new r3Vect(1, 1, 0);
f = v1.dot(v2);
```

Method: cross - cross product

Syntax: v3 = v.cross(v2);

Parameters: v2 - vector

Returns: v3 - result vector

Description: Computes cross product 'v x v2' and returns the resulting vector. The operation doesn't impact the object itself

Example:

```
v1 = new r3Vect(1, 0, 0);
v2 = new r3Vect(0, 1, 0);
v3 = v1.cross(v2);
```

Method: len - lenght of vector

Syntax: length = v.len()

Parameters: -

Returns: lenght - floating point number

Description: Compute and return the lenght of the vector.

Example:

```
v = new r3Vect(0, 2, 0);
l = v.len(); // l = 2;
```

Method: norm - normalize vector

Syntax: len = v.norm();

Parameters: -

Returns: len - original lenght of the vector

Description: Normalizes the vector and returns the original lenght.

Example:

```
v = new r3Vect(0.5, 0, 0);
v.norm();
```

Method: orth - orthogonal projection

Syntax: v3 = v.orth(v2);

Parameters: v2 - vector

Returns: v3 - result vector

Description: Orthogonal projection. The resulting 'v3' vector is perpendicular to 'v' and on the same plane as both 'v' and 'v2'.

Example:

```
v1 = new r3Vect(1, 0, 0);
v2 = new r3Vect(1, 1, 1);
v3 = v1.orth(v2);
```

Method: proj - projection

Syntax: v3 = v.proj(v2);

Parameters: v2 - vector to be projected on 'v'

Returns: v3 - vector

Description: Returns vectors 'v2' projection on 'v'.

Example:

```
v1 = new r3Vect(1, 0, 0);
v2 = new r3Vect(1, 1, 1);
v3 = v1.proj(v2);
```

Method: neg - negate

Syntax: v2 = v.neg();

Parameters: -

Returns: v2 - vector

Description: Returns a vector pointing to opposite direction than the object in question.

Example:

```
v = new r3Vect(1, 0, 0);
nv = v.neg();
```

Method: fneg - negative

Syntax: v.neg();

Parameters: -

Returns: -

Description: Makes the vector negative vector i.e. inverts the direction of the vector

Example:

```
v = new r3Vect(1, 0, 0);
v.fneg();
v.print(); // -1, 0, 0
```

Method: mul - scalar multiplication

Syntax: v2 = v.mul(f);

Parameters: f - scalar for multiplying the vector

Returns: v2 - result

Description: Multiplies the vector with the given scalar value and returns the resulting new vector.

Example:

```
v = new r3Vect(1, 2, 3);
v2 = v.mul(10); // v2 = 10, 20, 30
```

Method: fmul - scalar multiplication

Syntax: v.fmul(f);

Parameters: f - floating point number

Returns: -

Description: Multiplies the vector by given scalar.

Example:

```
v = new r3Vect(1, 2, 3);
v.fmul(10);
v.print(); // v = 10, 20, 30
```

Method: div - division by scalar

Syntax: v2 = v.div(f);

Parameters: f - floating point number

Returns: v2 - vector

Description: Divides the vector by given scalar and returns a new vector holding the result.

Example:

```
v = new r3Vect(1, 0, 0);
v2 = v.div(2.0);
```

Method: fdiv - division by scalar

Syntax: v.fdiv(f);

Parameters: f - floating point number

Returns: -

Description: Divides the vector by given scalar.

Example:

```
v = new r3Vect(1, 0, 0);
v.fdiv(2.0);
```

```
v.print(); // 0.5, 0, 0
```

Method: cmul

Syntax: v3 = v.cmul(v2);

Parameters: v2 - vector

Returns: v3 - vector

Description: Component wise multiplication. Multiplies each component in 'v' by corresponding component of 'v2' and returns the result as a new vector.

Example:

```
v1 = new r3Vect(1, 2, 3);
v2 = new r3Vect(1, 2, 3);
v3 = v1.cmul(v2); // v3 == 1, 4, 9
```

Method: fcmul - vector multiplication

Syntax: v.fcmul(v2);

Parameters: v2 - vector

Returns: -

Description: Multiplies the 'v' by given vector 'v2'.

Example:

```
v = new r3Vect(1, 2, 3);
v2 = new r3Vect(1, 2, 3);
v.fcmul(v2);
v.print(); // 1, 4, 9
```

Method: cmp - compare

Syntax: b = v.cmp(v2, tolerance);

Parameters: v2 - vector

tolerance - tolerance

Returns: b - boolean, true if vectors are identical within the given tolerance

Description: Compare the given vector 'v2' agains the vector in question and return true if the vectors are identical. Otherwise returns false.

Example:

```
v1 = new r3Vect(1, 1, 1);
v2 = new r3Vect(1.1, 1, 1);
v1.cmp(v2, 0.001);
```

Method: print - print vector

Syntax: v.print()

Parameters: -

Returns: -

Description: Print out the current value of the vector.

Example:

```
v.set(1, 2, 3);
v.print();
```

Method: angle - angle to given vector

Syntax: angle = v.angle(v2);

Parameters: v2 - vector

Returns: angle - angle in radians

Description: Compute the angle between the vector in question and the given parameter 'v2'.



Note

This method is not supported by 4d vector.

Example:

```
v = new r3Vect(1, 0, 0);
v2 = new r3Vect(0, 1, 0);
angle = v.angle(v2);
```

Method: dist - distance to given point

Syntax: dist = v.dist(v2);

Parameters: v2 - vector

Returns: dist - distance

Description: Measure the distance between two points specified by 'v' and 'v2'

Example:

```
v = new r3Vect(1, 0, 0);
v2 = new r3Vect(2, 0, 0);
distance = v.dist(v2); // distance == 1.0
```

Float Class

JavaScript does not support call-by-reference type of function parameters for basic types. Correspondingly, Realsoft 3D methods returning data back to the caller through the method parameters won't work. To overcome this limitation Realsoft 3D implements r3Float object type.

For example, the r3nurbs curve class implements method MEASURELENGTH(len), which returns the lenght of the curve through the function parameter 'len'. This would not be possible using JavaScript basic types:

```
// wont't work
var len;
curve.MEASURELENGHT(len);
```

because JavaScript basic types are passed to function as call-by-value parameters. To allow the MEASURE-LENGHT() method to actually change the value of the 'len' parameter, call-by-reference is required.

```
// this works
var len = new r3Float();
curve.MEASURELENGHT(len);
```

Methods

Method: r3Float - constructor for float

Syntax: v = new r3Float()

Parameters: value - floating point number

Returns: v - new float object

Description: Creates a new float object. If you don't pass any parameters to the constructor, the value is initialized to zero.

Example: v = new r3Float(10);

Method: add - addition

Syntax: v2 = v.add(v1);

Parameters: v1 - float

Returns: v2 = result float

Description: This adds the given float to the object in question and returns the result as a new float. The operation doesn't change the value of the object.

Example:

```
v = new r3Float(0.1);
r = v.add(0.1);
```

Method: fadd - float addition

Syntax: v.fadd(v2);

Parameters: v2 - float to be added

Returns: -

Description: Float operation v = v+v2.

Example:

```
v = new r3Float(1);
v.fadd(0.1); // add 0.1 to the current value
```

Method: set - set value

Syntax: v.set(value);

Parameters: value - floating point number

Returns: -

Description: Set new value to the float

Example:

```
v = new r3Float();
v.set(10.3);
```

Method: sub - float subtraction

Syntax: v3 = v.sub(v2);

Parameters: v2 - float to be subracted

Returns: float - the result of the subtraction

Description: Returns a new float containing the result of float operation 'v - v2'.

Example:

```
v = new r3Float(10);
v2 = v.sub(5);
v2.print(); // 5
```

Method: fsub - float subtraction

Syntax: v.fsub(v2);

Parameters: v2 - float to be subracted

Returns: -

Description: Subtracts given float 'v2' from the float in question.

Example:

```
v = new r3Float(1);
v.fsub(0.5);
v.print(); // 0.5
```

Method: neg - negate

Syntax: v2 = v.neg();

Parameters: -

Returns: v2 - float

Description: Returns negative value of the object.

Example:

```
v = new r3Float(1);
v2 = v.neg();
```

Method: fneg - negative

Syntax: v.neg();

Parameters: -

Returns: -

Description: Makes the float negative float.

Example:

```
v = new r3Float(1);
```

```
v.fneg();
v.print(); // -1
```

Method: mul - multiplication

Syntax: v2 = v.mul(f);

Parameters: f - float number

Returns: v2 - result

Description: Multiplies the float with the given value and returns the value.

Example:

```
v = new r3Float(2);
v2 = v.mul(10);
v2.print(); // 20
```

Method: fmul - multiplication

Syntax: v.fmul(f);

Parameters: f - floating point number

Returns: -

Description: Multiplies the float by given value

Example:

```
v = new r3Float(2);
v.fmul(10); // new value == 20
```

Method: div - division by scalar

Syntax: v2 = v.div(f);

Parameters: f - floating point number

Returns: v2 - float

Description: Divides the float by given value and returns a new float holding the result.

Example:

```
v = new r3Float(10);
v2 = v.div(2);
v2.print(); // 5
```

Method: fdiv - division by scalar

Syntax: v.fdiv(f);

Parameters: f - floating point number

Returns: -

Description: Divides the float by given value.

Example:

```
v = new r3Float(1);
```

```
v.fdiv(2); // 0.5
```

Method: cmul

Syntax: v3 = v.cmul(v2);

Parameters: v2 - float

Returns: v3 - float

Description: Component wise multiplication. Multiplies each component in 'v' by corresponding component of 'v2' and returns the result as a new float.

Example: v = new r3Vect(1, 2, 3);

Method: cmp - compare

Syntax: b = v.cmp(v2, tolerance);

Parameters: v2 - float

tolerance - tolerance

Returns: b - boolean, true if floats are identical within the given tolerance

Description: Compare the given float 'v2' agains the float in question and return true if the floats are within the given tolerance. Otherwise returns false.

Example:

```
v = new r3Vect(1.1);
if(v.cmp(1.2, 0.01)) {
    print("Not identical");
} else {
    print("Identical");
}
```

Method: print - print float

Syntax: v.print()

Parameters: -

Returns: -

Description: Print out the current value of the float.

Example:

```
v.set(1.5);
v.print();
```

Integer Class

JavaScript does not support call-by-reference type of function parameters for basic types. Correspondingly, Realsoft 3D methods returning data back to the caller through the method parameters won't work. To overcome this limitation Realsoft 3D implements r3Int object type.

Methods

Method: r3Int - constructor for int

Syntax: v = new r3Int()

Parameters: value - inting point number

Returns: v - new int object

Description: Creates a new int object. If you don't pass any parameters to the constructor, the value is initialized to zero.

Example: v = new r3Int(10);

Method: add - addition

Syntax: v2 = v.add(v1);

Parameters: v1 - int

Returns: v2 = result int

Description: This adds the given int to the object in question and returns the result as a new integer. The operation doesn't change the value of the object.

Example:

```
v = new r3Int(0.1);
r = v.add(0.1);
```

Method: fadd - int addition

Syntax: v.fadd(v2);

Parameters: v2 - int to be added

Returns: -

Description: Int operation v = v+v2.

Example:

```
v = new r3Int(1);
v.fadd(0.1); // add 0.1 to the current value
```

Method: set - set value

Syntax: v.set(value);

Parameters: value - inting point number

Returns: -

Description: Set new value to the int

Example:

```
v = new r3Int();
v.set(10.3);
```

Method: sub - int subtraction

Syntax: v3 = v.sub(v2);

Parameters: v2 - int to be subracted

Returns: int - the result of the subtraction

Description: Returns a new int containing the result of int operation 'v - v2'.

Example:

```
v = new r3Int(10);
v2 = v.sub(5);
v2.print(); // 5
```

Method: fsub - int subtraction

Syntax: v.fsub(v2);

Parameters: v2 - int to be subracted

Returns: -

Description: Subtracts given int 'v2' from the int in question.

Example:

```
v = new r3Int(1);
v.fsub(0.5);
v.print(); // 0.5
```

Method: neg - negate

Syntax: v2 = v.neg();

Parameters: -

Returns: v2 - int

Description: Returns negative value of the object.

Example:

```
v = new r3Int(1);
v2 = v.neg();
```

Method: fneg - negative

Syntax: v.neg();

Parameters: -

Returns: -

Description: Makes the int negative integer.

Example:

```
v = new r3Int(1);
v.fneg();
v.print(); // -1
```

Method: mul - multiplication

Syntax: v2 = v.mul(f);

Parameters: f - int number

Returns: v2 - result

Description: Multiplies the int with the given value and returns the value.

Example:

```
v = new r3Int(2);
v2 = v.mul(10);
v2.print(); // 20
```

Method: fmul - multiplication

Syntax: v.fmul(f);

Parameters: f - inting point number

Returns: -

Description: Multiplies the int by given value

Example:

```
v = new r3Int(2);
v.fmul(10); // new value == 20
```

Method: div - division by scalar

Syntax: v2 = v.div(f);

Parameters: f - inting point number

Returns: v2 - int

Description: Divides the int by given value and returns a new int holding the result.

Example:

```
v = new r3Int(10);
v2 = v.div(2);
v2.print(); // 5
```

Method: fdiv - division by scalar

Syntax: v.fdiv(f);

Parameters: f - inting point number

Returns: -

Description: Divides the int by given value.

Example:

```
v = new r3Int(1);
v.fdiv(2); // 0.5
```

Method: cmul

Syntax: v3 = v.cmul(v2);

Parameters: v2 - int

Returns: v3 - int

Description: Component wise multiplication. Multiplies each component in 'v' by corresponding component of 'v2' and returns the result as a new integer.

Example: v = new r3Vect(1, 2, 3);

Method: cmp - compare

Syntax: b = v.cmp(v2, tolerance);

Parameters: v2 - int

tolerance - tolerance

Returns: b - boolean, true if ints are identical within the given tolerance

Description: Compare the given int 'v2' agains the int in question and return true if the ints are within the given tolerance. Otherwise returns false.

Example:

```
v = new r3Vect(1.1);
if(v.cmp(1.2, 0.01)) {
    print("Not identical");
} else {
    print("Identical");
}
```

Method: print - print int

Syntax: v.print()

Parameters: -

Returns: -

Description: Print out the current value of the integer.

Example:

```
v.set(1.5);
v.print();
```

Tools

3D painting Tools

The 3D Painting tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the 3D Painting tools.



These tools can be drag&dropped to a tool bar.

Airbrush



Paint with an airbrush on UV mapped objects.

Set tool options through the Control Bar. Press down the left mouse button on a View window over a UV mapped object and move the mouse to paint with the air brush. You can release the left mouse button and move the mouse

to another place and start painting again by pressing down the left mouse button. To stop painting click Accept on the Control Bar or select Accept on the View window popup menu.

The airbrush tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Color - Color to paint.
- Continuous - If set, draws continuous line.

Circle



Paint outline or filled circles on UV mapped objects.

Set tool options through the Control Bar. Click two points on a View window over a UV mapped object to draw a circle. The first point defines the center of the circle and the second point defines a point on the circle's perimeter.

The Circle tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Filled - Outline or filled curve.
- Color - Color of the circle to paint.

Curve



Paint outline or filled freeform curves on UV mapped objects.

Set tool options through the Control Bar. Press down the left mouse button on a View window over a UV mapped object and draw the curve while holding the left mouse button down. Release the left mouse button to accept the curve.

The Curve tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Filled - Outline or filled curve.
- Color - Color of the curve to paint.
- Closed - Closed curve.

Line



Paint a polyline on UV mapped objects.

Set tool options through the Control Bar. Click points on a View window over a UV mapped object to draw a polyline. Click Accept on the Control Bar or select Accept on the View window popup menu to accept the polyline.

The Line tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Filled - Outline or filled line.
- Color - Color of the line to paint.

Paste Colors



Paste selected objects color values on other objects' texture maps. Projects the selected object's colors on the underlying UV mapped objects.

Select the objects whose colors to paste. Click Accept on the Control Bar or select Accept on the View window popup menu.

The Paste Colors tool has the following options:



- U resolution - Subdivision of object in U direction.
- V resolution - Subdivision of object in V direction.

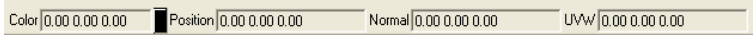
Picker



Color picker, read the color of the surface.

Click an object on a View window to pick its color.

The Picker tool has the following options:



- Color - Color of the picked surface point.
- Position - Surface hit point.
- Normal - Surface normal.
- UVW - UVW Coordinates.

Polygon



Paint symmetrical closed polygons on UV mapped objects.

Set tool options through the Control Bar. Click two points on a View window over a UV mapped object to draw a symmetrical polygon. The first point defines the center of the polygon and the second point defines one corner of the polygon.

The polygon tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Filled - Outline or filled polygon.
- Color - Color of the polygon to paint.
- Count - Number of points in the polygon.

Rectangle



Paint outline or filled rectangles on UV mapped objects.

Set tool options through the Control Bar. Click two points on a View window over a UV mapped object to draw a rectangle.

The Rectangle tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.
- Line Width - Custom line width.
- Filled - Outline or filled rectangle.
- Color - Color of the rectangle to paint.

Selected



Paint with the selected objects on UV mapped objects.

Select objects to be used as pens. Click Accept on the Control Bar or select Accept on the View window popup menu.

The Selected tool has the following options:



- Line Width - Select line width for 1 to 5 pixels or custom line width.

- Line Width - Custom line width.
- Filled - Outline or filled curve.
- Color - Color of the curve to paint.

UVImage



This tool creates and attaches a new image or a material with a texture map to the selected objects.

The tool can evaluate various shading components. It does so called 'baking' - procedural materials, complex illumination etc. can be evaluated and stored as a texture map. Using a texture map instead of a multi-layered procedural shader set can speed up rendering significantly. Baking is also a useful feature for scene export purposes.

To use the tool:

- Select objects to make the UV images for. For Illumination maps and surface shading, select also a light source because that helps to select the correct surface side.
- Set tool options through the Control Bar.
- Click Accept on the Control Bar or select Accept on the View window popup menu.

The UV Image tool has the following options:







- X - Pixels in X direction.
- Y - Pixels in Y direction.
- Type - Type of the image to be created.
- Constant - Initializes a blank image and assigns it to the selected channel
- Evaluate - Evaluates all materials and stores the result (in a selected channel) to a texture map
- Direct Lighting - Direct shadows and illumination from light sources
- Indirect Lighting - Global diffuse illumination bouncing from objects. Direct lighting is not included.
- Caustics - Light caustics casted by reflective surfaces
- Surface Shading - Evaluates all surface shaders from surface geometry to surface finishing. Illumination, reflections etc. are included. This option can be used e.g. for reflection map creation. The viewing direction in shading (which is relevant for reflection computation and some illumination effects) is perpendicular to the surface.
- Channel - Channel to be evaluated.

- UV - Parameter channel.
- Color - Constant fill color for the image to be created.
- Attach - Attach generated image to the selected object as texture map.
- Per Face - Attach the map as a SDS face material (for subdivision surfaces only).
- File - Name of file to write image to.
- Samples - Amount of supersamples per pixel.
- Backside - Selects the side of the surface for which illumination is evaluated.
- Settings - Defines recursion depth, threshold and traced rays.

Vertex Painter



Paint vertex properties. Select the object whose pointwise attribute you want to paint. Select the property you want to paint and the value to paint with.

Press down the left mouse button on a View window over the selected object and move the mouse to paint. You can release the left mouse button and move the mouse to another place and start painting again by pressing down the left mouse button. To stop painting click Accept on the Control Bar or select Accept on the View window popup menu.

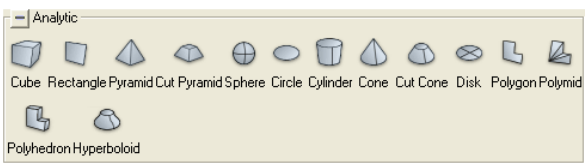
The Vertex Painter tool has the following options:



- Pen Size - Size of the circular pen.
- Pen Type
  - Solid - All parts of the pen have the same effect.
  - Gradient - The center of the pen has the maximum effect. The effect diminishes toward the perimeter of the circular pen.
- Value - Value to be painted with.
- Target - Property to be painted.

Analytic Tools

The Analytic tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains tools for creating analytic objects.



These tools can be drag&dropped to a tool bar.

Circle



Creates an analytic circle object.

Specify options in the Control Bar. Define the circle by clicking two or three points on a View window depending on the Define option. The Circle tool has the following options:



- Define - How the circle is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If checked, creates a disk, otherwise creates a circle.
- Arc - If checked creates an arc.

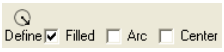
Cone



Creates an analytic cone object.

Specify options in the Control Bar. Define the bottom circle of the cone by clicking two or three points on a View window depending on the Define option. Define the apex point by clicking on a View window.

The Cone tool has the following options:



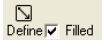
- Define - How the cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If not checked the top and bottom lids of the cone are not created.
- Arc - If checked creates a cone sector.
- Center - If checked, the apex will be centered over the bottom.

Cube



Creates an analytic cube object.

Specify options in the Control Bar. Define the cube by clicking two points on a View window. The Cube tool has the following options:



- Define - How the cube is defined, either by two corners or center and corner.
- Filled - If not checked the top and bottom face of the cube are not created.

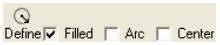
Cut Cone



Creates an analytic cut cone object.

Specify options in the Control Bar. Define the bottom circle of the cut cone by clicking two or three points on a View window depending on the Define option. Define the top circle by clicking two points on a View window.

The Cut Cone tool has the following options:



- Define - How the cut cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If not checked the top and bottom lids of the cut cone are not created.
- Arc - If checked creates a cut cone sector.
- Center - If checked, the top will be centered over the bottom.

Cut Pyramid



Creates an analytic cut pyramid object.

Define the bottom rectangle of the cut pyramid by clicking two points on a View window. Define the top rectangle point by clicking two more points on a View window. The tool has the following options:



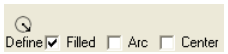
- Define - How the cut pyramid is defined, either by two corners or center and corner.
- Filled - If not checked the top and bottom face of the cut pyramid are not created.
- Center - If checked, the top will be centered over the bottom.

Cylinder



Creates an analytic cylinder object.

Specify options in the Control Bar. Define the cylinder by clicking two or three points on a View window depending on the Define option. The Cylinder tool has the following options:



- Define - How the cylinder is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If not checked the top and bottom lids of the cylinder are not created.
- Arc - If checked creates a cylinder sector
- Center - If checked, the top will be centered over the bottom.

Disk



Creates an analytic disk object.

Specify options in the Control Bar. Define the disk by clicking two or three points on a View window depending on the Define option. The Disk tool has the following options:



- Define - How the Disk is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If checked, creates a disk, otherwise creates a circle.
- Arc - If checked creates an arc.

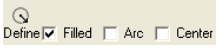
Hyperboloid



Creates a hyperboloid or ellipsoid segment object.

Specify options in the Control Bar. Define the bottom circle of the hyperboloid or ellipsoid segment by clicking two or three points on a View window depending on the Define option. Define the top circle by clicking two points on a View window.

The Hyperboloid tool has the following options:



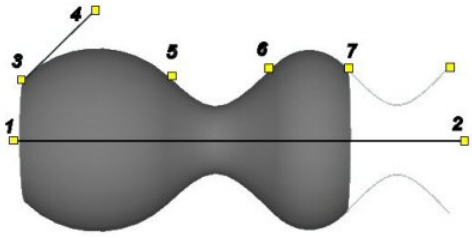
- Define - How the hyperboloid is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Filled - If not checked the top and bottom lids of the hyperboloid are not created.
- Arc - If checked creates a hyperboloid sector.
- Center - If checked, the top will be centered over the bottom.

Lathe



Creates a composite object consisting of sharply or smootly connected ellipsoid, hyperboloid, cone and cylinder objects.

The tool uses the first two entered points to define the symmetry axis. The remaining points define the surface of revolution. Every successive point creates a new quadric object that is joined with the previous object. The Ctrl key enables/disables smooth joining of consecutive sections. Correspondingly, the created object consists of cylinders, cones, hyperboloids and ellipsoid objects.



Surface angle, section thickness and radius can be set numerically from the measuring window.



Note

The tool does not automatically create concave sections, if the drawing direction is reversed against previous sections. The user can perform the required Boolean operations manually after the tool has created the quadric components.

Polygon



Creates an analytic polygon object.

Specify options in the Control Bar. If the Freehand option is set, then press down the left mouse button on a View window and draw the outline of the polygon while holding the left mouse button down. Otherwise click a number of points on a View window and accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Polygon tool has the following option:

- Freehand - Define polygon by moving the mouse while holding down the left mouse button.

Polyhedron



Creates an analytic polyhedron object.

Specify options in the Control Bar. If the Freehand option is set, then press down the left mouse button on a View window and draw the outline of the polyhedron while holding the left mouse button down. Otherwise click a number of points on a View window and accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Polyhedron tool has the following options:



- Depth - Distance between of the top and the base polygon.
- Freehand - Define polygon by moving the mouse while holding down the left mouse button.

Polymid



Creates an analytic polymid object.

Specify options in the Control Bar. If the Freehand option is set, then press down the left mouse button on a View window and draw the outline of the bottom polygon while holding the left mouse button down. Otherwise click a number of points on a View window and accept the bottom polygon by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar. Define the apex point by clicking on a View window.

The Polymid tool has the following options:



- Depth - Distance between the apex and base polygon.
- Freehand - Define polygon by moving the mouse while holding down the left mouse button.

Pyramid



Creates an analytic pyramid object.

Specify options in the Control Bar. Define the bottom rectangle of the pyramid by clicking two points on a View window. Define the apex point by clicking on a View window. The Pyramid tool has the following options:



- Define - How the pyramid is defined, either by two corners or center and corner.
- Filled - If not checked the bottom face of the pyramid is not created.
- Center - If checked, the apex will be centered over the bottom.

Rectangle



Creates an analytic rectangular plane.

Specify options in the Control Bar. Define the rectangle by clicking two points on a View window. The Rectangle tool has the following options:



- Define - How the rectangle is defined, either by two corners or center and corner.
- Filled - Whether to create a surface or a border line.

Sphere



Creates an analytic sphere object.

Specify options in the Control Bar. Define the sphere by clicking two points on a View window. The Sphere tool has the following option:

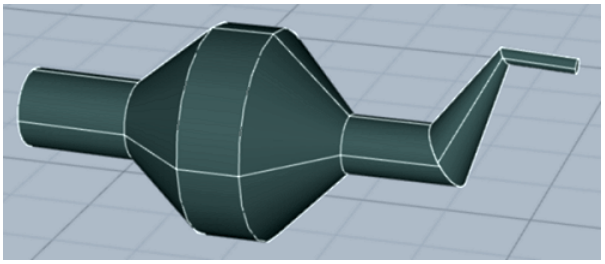


- Input - How the sphere is defined, by center and radius, two corners, or center and corner.

Tube



Creates a tube composite object from sharply or smootly connected cylinders and cones.



The tool uses the first two entered points to define the starting point and the radius for the tube. Every successive point defines a new cylinder or cone object connected to the previous object with the specified joint type (sharp, rounded).

Instead of the second mouse click, the radius of the tube can be defined numerically from the tool bar or using the mouse wheel. The radius can be adjusted any time during drawing with the wheel or numerically - this adds a cone surface, which changes the radius.

The tool sets up a parametric construction system. The position of joits and their radius can be changed later by editing the parameter spheres placed into the sub hierarchy.



Note

It is possible to connect two conical surfaces with common intersection only in some special cases.

Boolean Tools

The Boolean tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the Boolean operation tools.



These tools can be drag&dropped to a tool bar.

Boolean



The Boolean tool creates a Boolean operation object or a boundary representation solid.

Select the objects to be operated. Select the boolean operation. Select Accept from the View window's popup menu or click the Accept button on the control bar.

The Boolean tool has the following options:



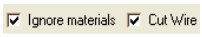
- Operation
  - Union - collect one or more objects inside a new level.
  - Intersection - AND operation between the selected objects.
  - Remove - first selected object AND NOT the remaining selected objects.
  - Solid - Boundary Solid. Create new solid primitives from one or more freeform surfaces that define closed volume.
- Ignore Materials - If set, the material of the first selected object is not affected by the tool object.
- Cut Wire - If set, wireframes of boolean operands are trimmed to match the result volume.

Intersect



Apply Boolean AND operation between the selected objects. The resulting object contains those parts of space that are inside all of the operated objects.

The Intersect tool has the following options:



- Ignore Materials - If set, the material of the first selected object is not affected by the tool object.
- Cut Wire - If set, wireframes of boolean operands are trimmed to match the result volume.

Invert



Invert the inside/outside property of the selected objects. For example: a cube with a drilled hole can be defined as AND boolean operation, where the volume of the cylinder (drill) is inverted.

Remove



Apply Boolean AND NOT operation between the selected object, i.e. the first selected object AND NOT the remaining selected objects. The resulting object contains those parts of space that are inside the first selected object but not inside any of the other selected objects. The Remove tool has the following options:

- Ignore Materials - If set, the material of the first selected object is not affected by the tool object.



- Cut Wire - If set, wireframes of boolean operands are trimmed to match the result volume.

Solid



Create a Boundary Solid. Create new solid primitives from one or more freeform surfaces that define a closed volume. The tool has no options.

Split



Split the first selected object into two parts using the second selected object. The Split tool has the following options:

- Ignore Materials - If set, the material of the first selected object is not affected by the tool object.



- Cut Wire - If set, wireframes of boolean operands are trimmed to match the result volume.

Union



Collect one or more objects (the currently selected objects) inside a new level. The resulting object contains those parts of space that are inside any of the operated objects. The Union tool has no options.

Building Construction Tools

Building



This is the main tool for building construction. Other building tools described below add sub components to the manager object created using this tool. To use the tool:

- Take a top view and click the building tool button
- Define some building options (see the option list below) if necessary
- Click a sequence of points in a view window to define the outer wall for the building
- In polyline mode, click Accept to finish an open wall sequence, or close the wall loop by clicking on top of the first point
- Go to the material library of the select window. Using the property window, select some texture maps for the material templates, which the tool created.

The tool has the following options:

- Type: rectangle or polyline. Rectangle mode switches automatically to polyline drawing if the two first points define a vertical or horizontal line (a collapsed rectangle would be useless). In polyline mode, the first line can travel in any angle.
- Closing: Automatic mode draws an open line which closes automatically when the user clicks on top of the first point. Open mode keeps the line open and does not finish at overlapping points. Closed mode forces a closed

wall loop, so that the tool can be finished by selecting Accept or by hitting enter key, and there is no need to aim at the first point to finish the tool.

- Auto Texture: when set, the tool adds material templates for texturing the building parts (walls, base, floor and ceiling) to the material library. The tool also adds required material maps to the geometric hierarchy of the building.
- Wall Thickness: can be defined here or changed later using the property window.

The defined wall point sequence can be found from the sub hierarchy of the building object. It is the first wall object in the 'wall definitions' sub folder. The actual point sequence is stored into a polygonal NURBS curve in the mentioned wall object hierarchy. Editing the NURBS curve updates the walls.



Note

Although the building tool accepts open wall sequences, real houses should include a closed outer wall loop in the first wall object. The same data defines directly the base and the roof edges. Splitting walls to several open sections disables automatic base and roof construction.

Hole

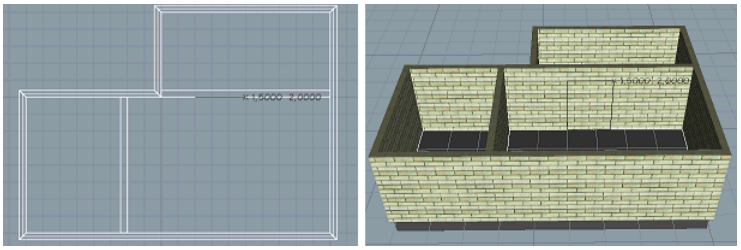


Cuts a hole to a wall. The tool has two operation modes:

- Wall surface projection
- Wall base line projection

The hole tool chooses the most appropriate projection automatically. If the hole is defined from a top view (or almost top view), drawing a rectangular hole to a vertical wall is practically impossible. Therefore, the tool lets the user to draw a line segment along the wall base lines. The mouse clicks define only the width of the hole; elevation from the floor level and height can be defined numerically or afterwards by editing the hole object.

If the modeling view shows the walls from a sufficiently perpendicular angle, the tool lets the user to draw the hole directly to the wall under the mouse pointer. All hole dimensions and the position is defined by the two mouse clicks.



Drawing a hole in orthogonal wireframe view and in perspective shaded view

To use the tool:

- Select a building. The hole tool appears to the tool bar. Click it.
- navigate a view to find a suitable modeling angle. Click two times to define the hole.

The tool adds a hole object to the sub level called 'holes' of the building object. A hole object is just a tagged rectangle object. You can move, stretch or duplicate it to get more holes. Rotated holes are not supported; hole edges must be parallel with the wall edges. Moving a hole away from the target wall turns hole-cutting feature off.

The tool does not have any toolbar options, but the dimensions and the elevation can be controlled numerically using the measuring window. The controls are useful when the hole is created from top view, because height and base elevation cannot be set with the mouse.

Roof



Adds a roof constructor object to the building. The roof tool does not have any tool options. To use it, select a building and click the roof tool. The tool creates a default roof shape immediately. The roof can be customized by editing its properties and using the Roof Line tool.

Roof Line

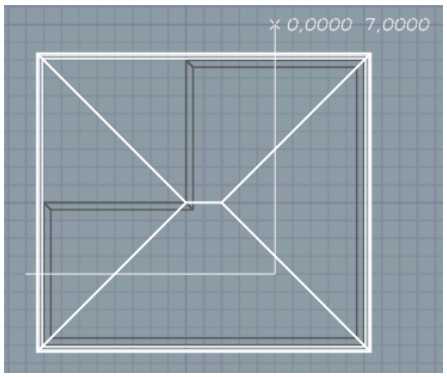


Defines the structure of the roof. The rectangular default roof can be changed to L-shaped, cross-like or to other custom shapes.

The tool is used as follows:

- Select a roof. The roof line tool appears to the tool bar. Click it.
- From a top view, click points to define a poly line. The ends of the line do not have to match exactly with the building.
- Click accept to finish the tool.

The tool can be applied multiple times to achieve complex branching roof structures. Defined roof lines are stored as NURBS objects into the 'roof lines' sub folder of the roof object. Editing the NURBS poly lines updates the roof.



Drawing a roof line



Note

The current version does not allow variable elevation of roof lines. All lines must be at the same height.

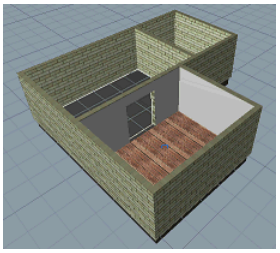
Room



Adds a local room decoration into the building. Use the tool as follows:

- Select a building. The Room tool appears, click it.
- Set the options for the room in the tool bar. Especially the name is important, because the created material templates will refer to it. A proper name will help you to identify the respective materials.
- Click inside a room you wish to decorate
- Go to the Material library of the select window and define properties for the room materials.
- The hierarchy of the room object contains sub folders, which define mouldings. You can add an example piece of moulding to the folders.

The tool has the following options:



- Texture Walls: covers the walls with a new surface (wall paper)
- Texture Floor: covers the floor with a new surface
- Texture Ceiling: covers the ceiling with a new surface
- Add Material Templates: adds the material templates for texture mapping the created surfaces.

Note, that you have an option to add just plain white surfaces, or the surfaces and material templates for texturing them.

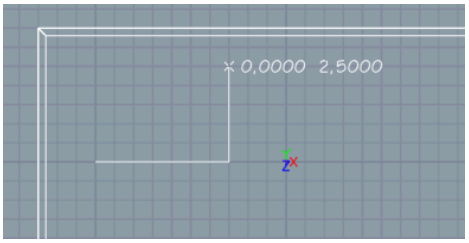
Sub Wall



Adds a new wall sequence to the building. The building can be split into rooms using this tool. Use the tool as follows:

- Select a building object. The sub wall tool appears to the toolbar. Click it to activate.
- Draw a wall point sequence. Note that when start and end snapping are active, you do not need to aim exactly at the outer wall. Just click close to the wall to which you wish to snap.
- Finish the tool by clicking Accept (or hit Enter)

The tool options are the same as for the building tool above with some exceptions. Auto texture option is not available; inner walls can be textured using the room tool. Snap Start/End options will search for the nearest walls and match the ends of the new wall respectively.



Sub wall will auto snap to outer walls

Construction Tools

Introduction

All tools in the construction tool group create an object, which applies a certain effect to its target objects. Any number of construction objects can be created to control an object.

For example, a *point translator* object can deform some of the points in a SDS object. Bend deformer can then be used for bending the deformed object. In this case, there would be two deformations associated with the SDS object.



A subdivision object controlled by point translator and bend objects



Bend deforms a lattice which deforms a SDS object

Construction objects can also be applied to other construction objects. For example, a *lattice deformer* can be used for bending a SDS sphere. A *bend deformer* can be used for bending the lattice deformer. This leads to a hierarchical construction system, where any object can have any number of constructors.

A certain construction effect can be modified by editing the attributes of the corresponding constructor object. To remove an effect of a constructor, delete the constructor object.

Construction based tools can be found from the Construction tool group.

Construction Tools

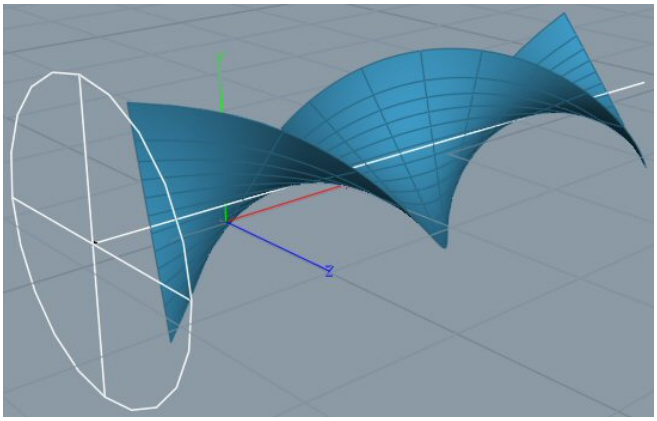
Axial



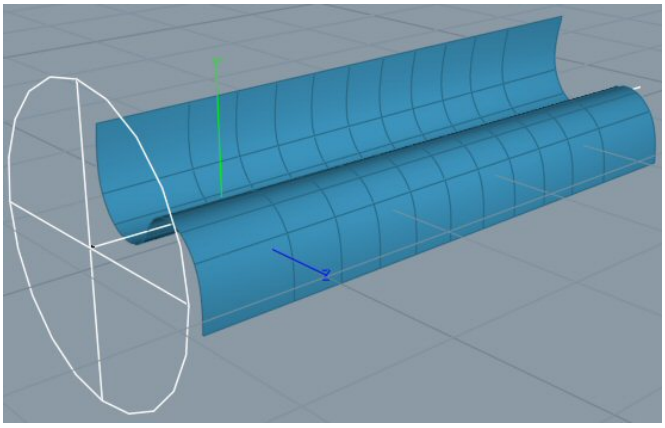
Axial deformer is a modifier that translates target points based on their distance or projection to the defined axis. The level of deformation can be controlled with three parameters: Angle, that controls the rotation about the axis (tangential translation), Scale controlling how much point is scaled about the axis and Translation that defines how much points are moved along the axis. Control curve option defines the level of effect.

To use the tool, select one or more target objects (particles, freeform surfaces), activate the tool and enter two points through the view window to define the deformation axis.

Rotation with distance mode. The distance mode can be used, for example, by creating tornados: the shorter the distance to the axis, the higher the rotation angle (that is, the angular velocity).



Rotation with projection mode. In this mode the rotation depends on the points position on the axis. The rotation angle in the beginning of the axis is zero. Full rotation angle is achieved at the end point of the axis.



The Axial deformer defines the following options:

Type	Conrols whether target points are deformed in function of projection-to-axis or distance-to-axis.
Curve	The control curve to be used for defining the geometry of the deformation.
Tangential, Radial, Axial	These values define now much the target point in question is rotated about the axis, scaled towards the axis and translated along the axis.
Animate	Controls whether the deformation is axis, or defined as a function of time.
Weight	Strenght of the deformation.

To create a tornado:

1. Create a particle cloud

2. Select the particle object, activate the tool and enter two points in the view window to define the symmetry axis for the tornado.

3. Use Hyperbolic control curve with Animation option and appropriate rotation angle.

4. Play

Bend Deformator

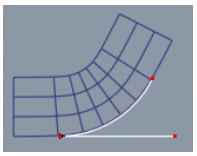


Bend deformator is a constructor based deformator. It can be used for generating circular bend effects to free form objects.

To bend a nurbs mesh:

1. Select the target object of bending and activate the Bend Constructor tool. Then define bending axis by clicking two times in a view window.

2. Switch the created bend object to point edit mode and drag the end point handle to bend the axis. The target object bends with the axis.



A bend deformator applied to a nurbs mesh

Carriage Tool

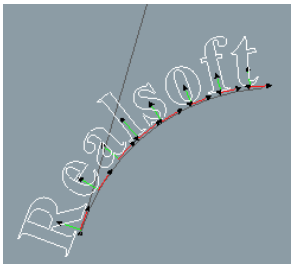


This tool creates so called 'carriage' object for the currently selected target objects. The object consists of a single coordinate system handle, through which the target objects can be transformed.

To use the tool:

- Select desired target object(s).
- Activate the tool and enter two points through the view window to define the coordinate system for the carriage object.
- Modify the carriage object and the target objects are modified accordingly.

The following image demonstrates the effect of the object.



Bend applied to a font curves through a set of carriage objects. Thanks to carriage objects, each font object is only translated and rotated, not bent

Construction Bind Tool



This tool can be used for binding constructor objects, such as Skeleton, Radial Deformator, Point Displacer and 3D lattice to their target objects. The tool can be used when you have to bind only some of the objects in a certain level.

To use the tool, select one or more target objects and then the constructor object to be bound, in this order. Activate the tool and click 'Accept'.

The tool shows you also one option 'Recursive'. If this option is set, the tool will bind also the sub objects of the selected target objects. Otherwise only the selected objects will be bound.

Displacer



Displacer is a very powerful constructor based deformator object. It allows the user to deform geometric objects using material mapping objects. It can be used, for example, to create waving flags and landscapes, just to name a few examples.



A displaced mesh consists of a mesh and a displacer object

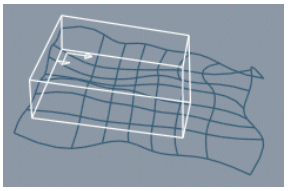
The displacement effect is defined by a material mapping object, which is given as a sub object for the displacer.

To create a waving flag:

1. Create a nurbs mesh consisting of 10 x 6 control points and representing a flag.

2. In the material selection tab, select the desired material and create a parallel mapping object.

3. Multi select the nurbs mesh and the material mapping object, and click the displacer tool.



A waving flag generated by moving a displacement deformator over a mesh

4. In the property window, set the 'Space' option to 'UVW'.

5. Create a key frame animation where the parallel mapping object moves over the mesh.

Drag Tool



To use the tool:

- Create an animated object (say, a spinning wheel or a walking character)
- Create an object representing the ground, say an analytic rectangle.
- Multi select both objects: first animated object and then the ground object.
- Click the tool and enter two points in the view window to define direction and strength of gravity. Use the property window to set desired rebound energy and surface friction for the drag object.

Inverse Kinematics Tool



This tool creates so called Inverse Kinematics object for the currently selected skeleton object. If any of the joints in the selected skeleton is inside the created I.K object, the I.K object is automatically bound to the joint. Because binding is based on inside/outside test rather than to selected joints, is easy to save animated I.K objects to a file and use them later to animate other skeletons.

Moving the I.K object will apply Inverse Kinematics to the attached joint.

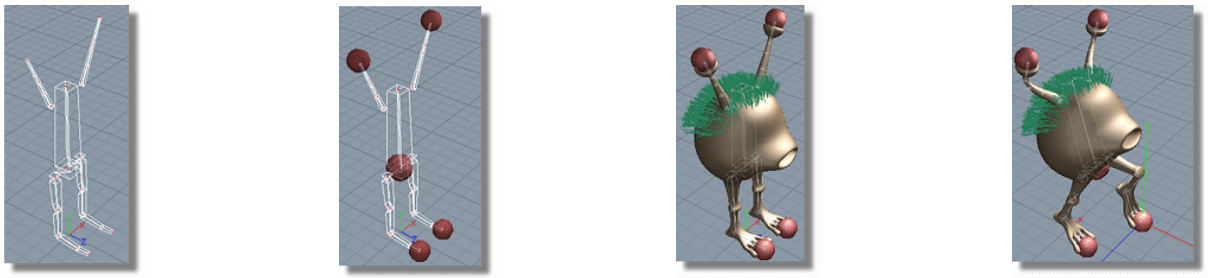
Example 1:

- Create several I.K objects and use motion capture system to define their translations.
- Load in desired skeleton and match desired skeleton joints to desired I.K objects.
- Multi select all I.K objects and click 'Bind'.
- Bind desired body to the skeleton.

Example 2:

- Create a skeleton and use move tool to animate its position (transform it in the object mode so that you have a rigid moving skeleton).

- Set Anchor for the root joints of the both legs.
- Use the tool to create I.K objects for the feet. Modify the positions and radiuses of the I.K objects so that they match the tail joints of the legs.
- Multi select all the I.K objects and click 'Bind'.
- Animate the I.K objects according to the movement of the skeleton to create a walking object.

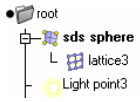


Attach I.K objects to a skeleton. Attach a body to a skeleton. Animate I.K objects.

Lattice



Lattice object is a global deformator object. Create it into an object and all the geometric points in the object are influenced by the lattice object.



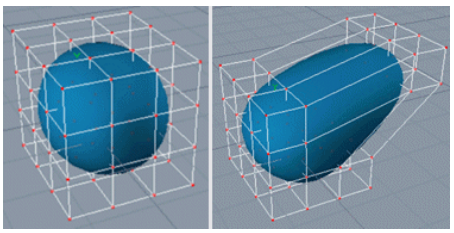
Lattice applied to a SDS sphere

The lattice object can be used as a deformation tool for modeling purposes, or as an animation tool. When activated, the tool provides three integer fields which define the point counts in three dimension.



Number of points in x, y and z dimensions

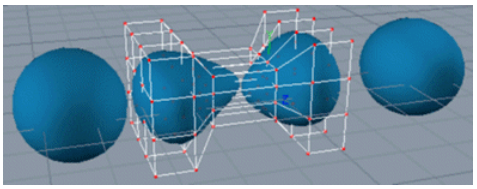
One can create a lattice over any object and deform it to deform the target object.



A SDS sphere deformed by a lattice object

You can also move an object through a deformed lattice object to get a deformation effect applied.

Lattice object properties are explained in models/geometrics/lattice documentation.



A SDS sphere moving through a lattice

Look At Tool



This tool creates a 'lookat' object. The object rotates the target objects about their origins so that their relative orientations with respect to the lookat object is maintained. For example, you can create a large number of planar trees, and use this object to control the orientation of the trees so that they will always face towards the camera.

The lookat object also defines a direction vector, which controls the banking of the target objects.

To use the tool.

- Create an object representing a gun.
- Select the gun object and create a lookat object so that the gun will point directly towards the lookat object.
- Move the lookat object sideways. The gun will be rotated accordingly.

The lookat object defines two options for controlling the rotation:

- Pitching On/Off.
- Banking On/Offp

These can be set in the property window.

Morphing



This tool creates a morphing object. The object interpolates its sub objects and assigns the result to its target objects.

To create a morphing object:

1. Multi select desired key objects. The selected objects must have identical hierarchical structure and vertice count.

2. Apply the Morphing tool

Morphing is set up. Play the animation to see how the target object interpolates through the specified key objects.

See also: Morphing.

Move



Move constructor translates the target objects based on the specified translation vector.

To create a move constructor:

1. Select the desired target objects and activate the Move constructor tool.

2. Enter two points through the view window to define translation vector.

Noise



Noise is a modifier that uses harmonic fractal noise to deform the target points. The weight attribute can be used for controlling the strenght of the noise, and the cube geometry defines the size of the noise.

To use the tool, multi select one or more target objects (particles, freeform surfaces) and enter two points through the view window to define the bounding box for the noise object.

The noise object defines the following options for controlling its effect:

2D Disable noise in the 'z' direction

Weight Strenght of the noise.

To create a flock of wriggling particles:

1. Create a flock of 1D particles and animate the translate attribute on object's 'X' axis. So you have a flock of particles moving horizontally in some rather boring fashion.

2. Select the particle object and activate the Noise tool. Enter two points in the view window to define bounding box for the deformer.

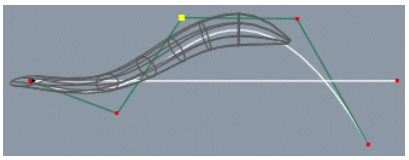
4. Play the animation.

Path Level



Path Level object is similar to bend deformator in that it can be used for bending geometric objects. A path level defines a deformation using a nurbs curve.

To create a wriggling snake:



1. Create a horizontal tube representing a snake.

2. Animate the snake to move horizontally.

3. Activate the Path Level tool and define an initial bending axis along the tube. Then single point edit the nurbs curve found inside the Path Level object to get desired deformation effect. Play the animation to see a wriggling snake.

A suitable banking option can be selected through the property window.

Point Translator



Point Translator can refer to one or more geometric points and define point wise translations for the points. In this respect, it is similar to the *Sub Group* object. However, the point translator is based on the construction system whereas the sub group is based on the lattice mapping system.

Point translators are typically used with skeletons to define facial expressions and other muscle deformations.

For example, one can create a point translator for the points representing a nose and animate the point translator to get facial expressions defined for the nose.

To animate a nose of a human head with the point translator:

1. In edit mode, select the points representing the nose of a head

2. Click the Point Translator tool. A point translator object is created.

3. Activate the 'Construct' option for the point translator.

4. Modify the point translator object in animation recording mode to animate the nose points.

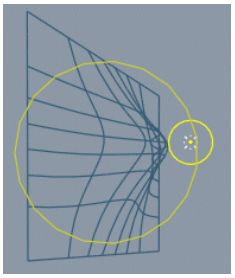
Properties defined by the point translator object are explained in models/geometrics/pointtranslator page.

Radial Deformator



Radial deformator is a constructor based deformator. The shorter the distance to the center point of the deformator the stronger the deformation effect.

The initial position and radius of the deformator represents the state of no deformation. The desired deformation effect can then be achieved by editing the center and the radius of the sphere.



A radial deformator applied to a nurbs mesh

All points inside the initial radius will be influenced by the deformator. All points outside the sphere will not be affected by the deformator.

By moving the center point, one can achieve a magnet like effect. All points are moved with the sphere. The higher the distance of the point and the center of the initial sphere the smaller the translation.

To deform a nurbs mesh:

1. Create a nurbs mesh consisting of say 10 x 6 control points

2. Activate the radial deformer tool and specify two points defining the center point and radius attributes for the deformer.

3. Switch to edit mode and drag the center point or the radius handle of the deformer. The target points are deformed accordingly.

Setting the radius attribute to a smaller value generates a gravity like effect. In other words, target points are scaled towards the center point rather than translated with it.

The strength factor can also be controlled by defining a deformation curve through the property window. The default curve is linear. The beginning of the curve represents the maximum distance of influence and the end point corresponds to the zero distance.

The deformator also supports fractal noise deformation effect. The strength and density of the noise effect can be defined through the property window. This option is useful for creating turbulence effects. For example, a star ship can be moved through a noise field.

Rotate



Rotate constructor rotates the target objects about the specified axis.

To rotate the selected objects by three full circles:

1. Click the 'Rotate' constructor tool to activate it:

2. Enter two points in the view window to define rotation axis.

3. Specify the desired rotation angle through the property window. To make the target object to rotate three full cycles, enter 3 * 360.

Rotator



Create a rotator object for rotating copies of selected objects about a given point or aligning them along a given line. Select the object(s), activate the Rotator tool and set options. Define a point (Rotate mode) or a line segment (Align mode) on a View window.

The rotator tool has the following options:



- Mode - Defines whether to rotate or align.
- Orientation - Defines how rotation axis is given. This control is available only when Rotate mode is selected.
  - Front
  - Side
- Copy Method - Defines whether to duplicate or instance the selected object
  - Instance
  - Duplicate
- Count - Number of copies

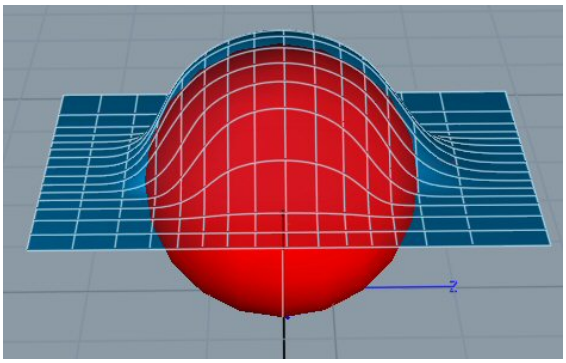
See also: Rotator Object.

Shrink



Shrink Wrap is a modifier that projects its targets point towards the sub object of the deformer. Parallel, spherical or cylinder wise projection can be defined. Collision detection system is used to constraint the shrink effect.

To use the tool, multi select one or more target objects (particles, freeform surfaces) and then one 'tool' object'. Then enter two points through the view window to define the geometry of the projection. The interpretation of the two defined points depends on the selected shrink projection as follows:



- Parallel: The line between the two points defines a fixed shrinking direction. Only direction matters, line position is irrelevant.
- Cylinder: Points are projected perpendicularly towards the defined axis line. Set the axis carefully inside the cylindrically shrunk geometry.
- Spherical: Points are projected towards the first defined point. Direction of the line is irrelevant in this case.

In all three projections, the length of the defined line sets the maximum limit for the shrink effect. Furthermore, spherical shrink does no shrink points beyond the center point and cylindrical shrink does not shrink points to the opposite side of the shrink axis.

The shrink object defines the following options:

Type Projection type: parallel, cylindrical or spherical.

Weight Strenght of the deformation.

To wrap a nurbs sphere around an analytic cube:

1. Create a point dense nurbs sphere and a smaller analytic cube inside the sphere.

2. Select the nurbs sphere and the analytic cube, in this order. Activate Modifier/Shrink tool.

3. Set Projection=Spherical from the tool bar.

4. Click in the middle of the sphere. Points will be projected towards the first defined point. Move the mouse pointer to define the amount of shrinking, then click second time. In spherical shrink, the direction of the second point from the first point does not matter, only the distance between the two points.

Skeleton



Skeleton is a special purpose deformation object typically used for controlling bodies in character animations. When the skeleton is modified, the attached objects are modified accordingly.

To create a skeleton representing a leg:

1. Activate the Skeleton tool from the 'Construction' tool tab.

2. Take a side view. Click the first point at the hip bone. Now, move the mouse towards the knee but don't fix the position for the knee yet. We will have to define proper constraints for the thighbone first. Rotate the mouse upwards

so that it represents the knee in its uppermost position. Then move the mouse to the opposite direction so that it shows the position of the knee in its back position. Then move the mouse to a position between the two extreme angles and to a suitable distance (the length of a thighbone) and fix the second point (the knee) by clicking the mouse.

3. Then move the mouse towards the ankle and repeat the steps above in order to define constraints for the shinbone as well.

4. Next define a bone from the ankle to the tip of the toes (we do not create toe bones this time) with appropriate constraints. Then select 'Accept' from the pop-up menu to terminate the skeleton creation.

Then you can bind the skeleton to a target object (such as a SDS mesh representing a human body) using appropriate binding tools.

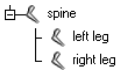
Skeletons can also be grouped to hierarchical structures to represent, for example, the bone structure of a human body.

To create a hierarchical skeleton, say a spine with two legs:

1. Switch to the side view and create a skeleton representing the spine of a human body.

2. Create a skeleton representing a leg.

3. Drag & drop the leg inside the spine. The leg is now a sub object of the spine.



A hierarchical skeleton system

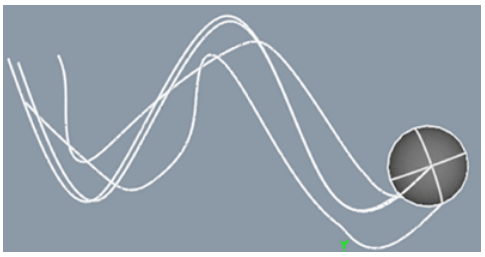
4. Repeat the steps 3 and 4 to create another leg. Then select the legs and click the 'Attach' tool. This attaches the legs to the spine so that if you drag one of the leg joints far enough, the effect will propagate to all three skeletons.

Traject



Traject tool can be applied to animated objects in order capture their motions into a trajectory curves. The tool supports three kind of curves: continuous, point and dash.

Trajectories created by moving and rotating analytic sphere. Continuous line type used.



The Traject object defines the following options:

Mode	In 'Object' mode only one trajectory is created for the object. In 'Point' mode trajectories are created per geometry point (vertex).
Line Type	Continuous, Dot or Dash. Nurbs curves, 1D particles or 2D particles are used to represent trajectories accordingly.
Max Count	Maximum number of points allowed per trajectory. Setting this value to zero means that the maximum point count is infinite.
Line Width	Diameter of the generated trajectory (in millimeters).
Step	Minimun distance between two subsequent points. No new points get inserted before the distance to the previous point is higher than this value.

To use the tool:

1. Create a moving object.

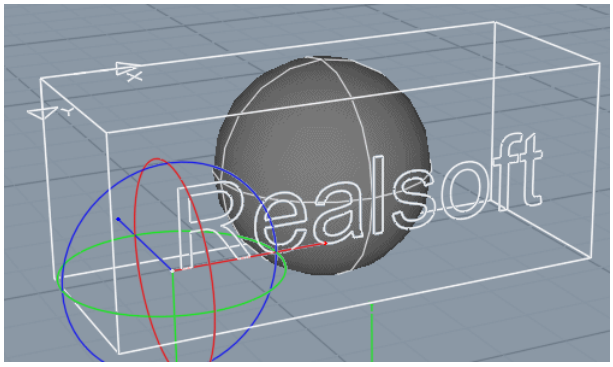
2. Select the animated object and apply the tool.

3. Play the animation

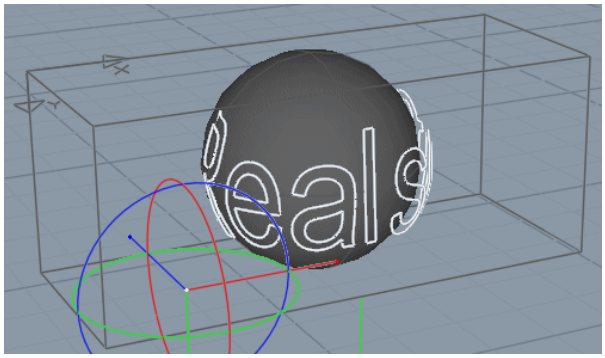
Warp



Warp is a modifier that uses its sub object to deform its targets points. For example, using a sphere as a sub object one can deform a planar mesh to spherical mesh.



To use the tool, multi select one or more target objects (nurbs curves, particles, freeform surfaces etc.) and one deformation geometry object (say an analytic sphere) and enter two points through the view window to define the bounding box for the warp object, in this order.



The warp object defines the following options for controlling the warp deformation:

Volume — If set only the points inside the objects bounding box will be deformed.

2D — Set to project all points onto the surface of the deformation geometry.

Periodic — Disable extrapolation of the deformation geometry for the points that don't fit inside the bounding box of the warp object.

Weight — Strenght of the warp effect.

To create a waterfall of particles:

1. Create 1D particles and animate the translate attribute on object's 'X' axis. So you have a flock of particles moving horizontally in some very boring way.

2. Create a bended NURBS mesh representing the shape of the waterfall.

3. Multi-select the particle object and nurbs mesh, activate the Warp tool and and enter two points in the view window to define bounding box for the warp deformer.

4. Play the animation and the particle object will bend along the surface of the NURBS object.

Binding Tools

When a skeleton and one of its target objects are selected the control bar shows the following skeleton map specific tools:

Bind

In edit mode, this tool binds the currently selected target points to their nearest currently selected joints. In Object mode all points in the selected target objects are bound to their nearest bones of the selected skeleton.

To use the tool, multi select one or more target objects and a skeleton object. Then switch to point edit mode and select the desired bones and the target points. Click the tool and the selected points are bound to the nearest selected joints.



Note
In case more than one joint is selected, each point gets mapped only to the nearest one! To bind a point to the nearest bone, just select the point and all the joints in the skeleton.

Unbind

This tool unbinds the selected points from the selected joints. If the target object is hierarchical, its sub objects are not bound.

To use the tool, multi select a target object and a skeleton object. Then switch to edit mode and select some joints and target points. Click the tool and the selected points are unbound from the selected joints.

Let's assume some of the points representing a nose got bound to a wrong skeleton. To unbind the points from the skeleton:

- Multi select the points to be unmapped and all the joints in the skeleton from which the points should be unbound. The tool is now shown in the control bar.
- Click the tool.

To unbind a point that got bound to a wrong joint:

- Select all points in the target object
- Select the joint in question
- Apply the tool

Points associated with the selected joint become unbound.

Select

The tool selects all target points bound to the currently selected joints.

The tool is only shown when both the target object and the skeleton object is selected.

The tool is useful, for example, when you want to view the points bound to a certain bone. Or, when some of the points got bound to a wrong bone and you want to unbind them. Just select the bone and click the 'Select' tool to select the target points as well.

Deselect

Deselect all the target points bound to the currently selected joints.

The tool is only shown when two objects are multi selected: a skeleton object and its target object. The tool is useful when binding points manually to complex hierarchical skeletons. It allows the user to exclude points already mapped to other skeletons.

Fidelity

This tool can be used for setting point wise fidelity.

To use the tool, select the desired target points and the associated joint. Then specify desired fidelity value through the fidelity field.

Fidelity is a special form of multi binding, based on the fact that in most cases multi binding operates over subsequent joints so that points near a joint behave like an elastic skin.

If the fidelity for a point is 0, then the point is bound only to one joint. If the fidelity is 0.5, then the point is bound to its closest joint with a weight 0.5 and to the next joint with a weight 0.5. In other words, such a point is influenced by two joints.

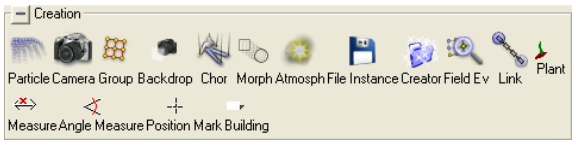
If the fidelity is 1, then only the next joint controls the point.

Weight

Sets a multi binding weight for the selected points. Select the target points and the associated joints. Then specify a weight value through the weight field.

Creation Tools

The Creation tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains some creation tools.



These tools can be drag&dropped to a tool bar.

Atmosphere



Creates an Atmosphere object, which defines volumetric properties for the entire space.

The atmosphere tool has the following options:



- Material - Material to be used for defining the atmospheric effects. Only VSL materials that contain the Volume phase can be used.

See also: Atmosphere Object.

Backdrop



Creates a backdrop object for the currently selected camera. If no camera is selected then a camera is created as well.

The tool has the following options:



- Distance - distance between the image and the camera.
- Material - name of the material to be mapped to the backdrop image.
- New Image - create new texture map.
- Color - create constant color backdrop.

See also: Backdrop Object.

Camera



You create a camera object by defining three points on a View window. The first point determines the origin of the camera. The second point determines the aiming direction. The third point determines the viewing angle (and the focal length) of the camera lens.

See also: Camera Object.

Choreography



This tool creates a choreography object for hierachical time lines.

Let's imagine you have a large number of animated objects. Correspondingly, the choreographer window shows you a large number of time lines. In order to control the timing of all the choreographs through a single time line:

1. Click the tool to create a choreographer object.

2. Move all the animated objects inside the newly created choreography object.

3. Make sure the choreography object is selected and open the Choreographer window. Now you can see only one time line through which you can control all the sub time lines.

See also: Choreography.

Constraint



Create a constraint object for the selected objects. When any of these target objects are modified, the constructor object's script, as defined in the Spec tab of the property window, will be executed. Constraint objects can be used to set up relations between geometric objects, e.g. to move an object to a position that is defined by the sub object of the constraint.

The constraint tool has the following options:

- Recursive - if set, the constraint object is bound also to the sub objects of the selected target objects.

The constraint tool creates a new choreography to the selected objects. The position in the choreography stack determines when the constructor object's script is executed.

The constructor object defines the following variables:

- Target - target object
- Self - the constructor object the script is associated with
- Time - current time
- P - the current target attribute (associated with the choreography). The type of the variable depends on the the type of the target attribute (vector, vector4, float, integer, axis, coordinate system).
- Type - type of the current target attribute
- Tag - attribute identifier of the current target attribute
- Index - attribute index of the current target attribute, or -1 for regular attributes

Example:

- select a set of objects to be controlled
- create a constraint object
- open the property window and enter desired script through the Spect tab.

See also: Constraint Object.

File Instance



This tool creates an object which instances specified Realsoft 3D project file (.r3d).

The object is visualized as a bounding box, representing the actual bounding box of the referred project.

The created file instance object can be transformed similar to any geometric object.

Because the actual data resides on a disk, File Instance objects can be used for constructing complex scenes.

Only geometric objects, materials and particle effects are loaded from the specified project.

To use the tool to instance a project:

1. Click the tool to activate it.

2. Select the desired .r3d file through the shown tool control bar.

3. Click the Accept button.

Instead of accepting the tool by clicking the Accept button, you can enter two points through the view window to define bounding box for the project.

The property window allows the user to disable any of the three chunks.

Group



The Group tool is used to create objects that contain "selection sets" of NURBS mesh, NURBS curve, or Subdivision object points. First select the points, then apply the Group tool.

See also: Group object.

Instance



Creates instances of the selected objects. An instance duplicates the geometry of the target object with a private matrix transformation.

Link



The link tool creates link objects that point to the currently selected geometric objects. A link object can be used where a geometric object would be evaluated.

See also: Link Object.

Metaball



Create a Metaball object.

Select one or more analytic spheres and then apply the tool to create a Metaball object.

Particle Tools

Distribute



Create a particle distributor from the selected geometric objects. The particles will be distributed over the surfaces of the selected objects.

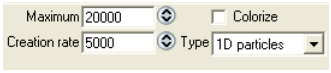
When actived the tool shows the following options:

- Type - Particle type, possible values are:
  - 1D Particle - is defined by a single point
  - 2D Particle - consists of two points. So, it has properties such as 'direction' and 'length'.
  - 3D Particle - is represented as a coordinate system
- Maximum - Maximum number of particles this object can have
- Creation rate - Particle creation rate
- Colorize - If set the color of the created particle will be determined by the color of the surface from which the particle was emitted

Emitter



Create a particle emitter from the selected geometric objects.

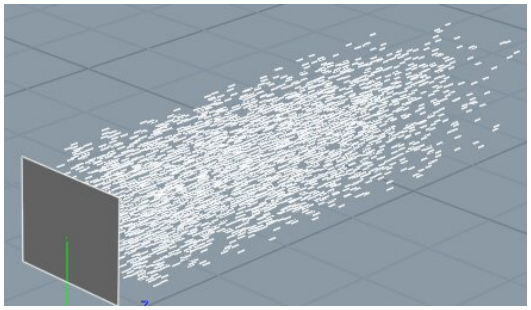


When actived the tool shows the following options:

- Type - Particle type, possible values are
  - 1D Particle - is defined by a single point
  - 2D Particle - consists of two points. So, it has properties such as 'direction' and 'length'.
  - 3D Particle - is represented as a coordinate system
- Maximum - Maximum number of particles this object can have
- Creation rate - Particle creation rate
- Colorize - If set the color of the created particle will be determined by the color of the surface from which the particle was emitted

Analytic rectangle used as an emitter.

More options can be set in the property window.



Filler

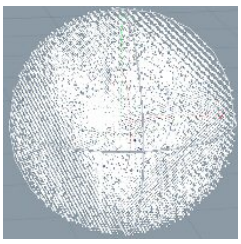


Create a particle filler from the selected geometric objects.

When actived the tool shows the following options:

- Type - Particle type, possible values are
  - 1D Particle - is defined by a single point
  - 2D Particle - consists of two points. So, it has properties such as 'direction' and 'length'.
  - 3D Particle - is represented as a coordinate system
- Maximum - Maximum number of particles this object can have

- Creation rate - Particle creation rate
- Colorize - If set the color of the created particle will be determined by the color of the surface from which the particle was emitted



Analytic sphere filled with 200 000 point particles.



Note

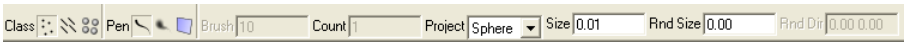
The feature works with any solid object, such as an analytic sphere, or with level of freeform objects that has their 'Boolean' type set to 'Boundary Surface Solid' and that are spatially closed (when it is not possible to get inside it without passing through a surface).

Particle



Create clouds of 1D, 2D, or 3D particles by painting with the mouse.

The particle tool has the following options:



- Class - Particle class, possible values are
  - 1D Particle - is defined by a single point
  - 2D Particle - consists of two points. So, it has properties such as 'direction' and 'length'.
  - 3D Particle - is represented as a coordinate system
- Pen - Pen type, possible values are
  - Pencil - creates one particle for each mouse movement
  - Brush - The number of particles created for each mouse movement can be defined through the Count field.
  - Object - Emit particles from a selected curve. This creates particles along the selected NURBS curve. The number of particles generated for each mouse move event can be defined using the Count control.
- Brush - controls the radius of the airbrush particle tool. The size is measured in pixels.
- Count - Defines the number of particles created per each mouse movement when using the airbrush and selected object pens.
- Project - controls the depth coordinate of particles added by the airbrush pen. Three choices are available:
  - Disc: Particles are created inside a circular area defined by the brush in the input plane.
  - Sphere: Each mouse movement adds a new spherical cloud of particles around the mouse position defined by the input plane. The diameter of the particle cloud is defined by the Brush size control.

  - Surface: Particles are added to the surface of the selected objects. The surface also defines the orientation of the particles.

- Size - controls the size of created particles. With 2D and 3D particles, the size controls the distance of the actual 3D points defining each particle. With 1D particles, the size is stored as a pointwise rendering attribute.
- Rnd Size - randomize the particle size. Value 0 makes all particles equally big (as defined by Size above). Value 1 picks a totally random value between 0 and the value of the Size field.
- Rnd Dir - randomize particle direction. Not available for 1D particles, which do not have any direction. Randomization is controlled with 2 values. The first value defines how much the axis of each particle changes randomly. In case of 3D particles, the controlled axis is the z axis. The second value (unused for 2d particles which are symmetric) controls the random rotation around the axis.

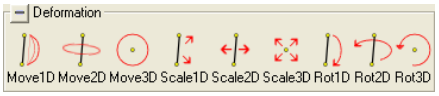
  The greater the values, the more randomized direction each particle gets. For the first component, value 1.0 already randomizes the direction 180 degrees, but the original direction still dominates the average of directions. For the second random component, value 1.0 gives a totally random rotation around the axis.

  For example, when brushing tree leaves onto a ground surface, the first random component can be zero, because the flat leaves lie along the ground. The second component should be one, so that leaves have a totally random direction on the ground. In other words, a suitable random value pair is "0 1".

See also: Particle Objects.

Deformation Tools

The Deformation tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the deformation tools.



These tools can be drag&dropped to a tool bar.

Move1D



Move with deformation as a function of projection to an axis.

Define the axis by clicking two points on a View window. Each point is projected to the axis to get a deformation parameter. Points that project to near the first end of the axis get smaller deformation parameters than points that project near the second end. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click on a View window to start the deformation. Move the mouse to deform and click to accept the tool.

Move2D



Move with deformation as a function of distance to an axis.

Define the axis by clicking two points on a View window. Distance to the axis is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click on a View window to start the deformation. Move the mouse to deform and click to accept the tool.

The Move2D tool has the following options:



- Target - Transformation target.

- Native
- Geometry
- LatticeCoords
- Handles
- Texture

- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.

Move3D



Move with deformation as a function of distance to a point.

Click two points on a View window to define a circle. The radius of the circle defines how far the deformation is effective. Distance to the center of the circle is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click on a View window to start the deformation. Move the mouse to deform and click to accept the tool.

The Move3D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random

- Linear

- Collision Deformation - Activate collision detection system.

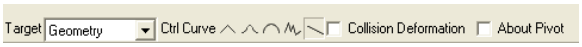
Rotate1D



Rotate with deformation as a function of projection to an axis.

Define the axis by clicking two points on a View window. Each point is projected to the axis to get a deformation parameter. Points that project to near the first end of the axis get smaller deformation parameters than points that project near the second end. The control curve is then evaluated by the deformation parameter to get the amount of deformation. click on a View Window to define the center of rotation. Click on a View Window to define the point you want to rotate about the rotation center. Move the mouse until desired rotation is achieved and click to accept the tool.

The Rotate1D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.
- About Pivot - Rotate object about the pivot point.

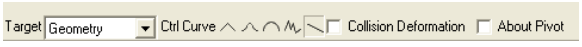
Rotate2D



Rotate with deformation as a function of distance to an axis.

Define the axis by clicking two points on a View window. Distance to the axis is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click on a View Window to define the point you want to rotate about the rotation center. Move the mouse until desired rotation is achieved and click to accept the tool.

The Rotate2D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.
- About Pivot - Rotate object about the pivot point.

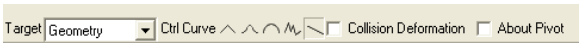
Rotate3D



Rotate with deformation as a function of distance to a point.

Click two points on a View window to define a circle. The radius of the circle defines how far the deformation is effective. Distance to the center of the circle is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click on a View Window to define the point you want to rotate about the rotation center. Move the mouse until desired rotation is achieved and click to accept the tool.

The Rotate3D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian

- Circle
- Random
- Linear

- Collision Deformation - Activate collision detection system.
- About Pivot - Rotate object about the pivot point.

Scale1D



Scale with deformation as a function of projection to an axis.

Define the axis by clicking two points on a View window. Each point is projected to the axis to get a deformation parameter. Points that project to near the first end of the axis get smaller deformation parameters than points that project near the second end. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click two points on a View window to define the scaling center and scale. Move the mouse to deform and click to accept the tool.

The Scale1D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.
- About Pivot - Scale object about the pivot point.

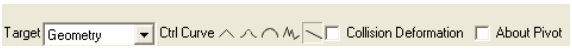
Scale2D



Scale with deformation as a function of distance to an axis.

Define the axis by clicking two points on a View window. Distance to the axis is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click two points on a View window to define the scaling center and scale. Move the mouse to deform and click to accept the tool.

The Scale2D tool has the following options:



- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles
  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.
- About Pivot - Scale object about the pivot point.

Scale3D



Scale with deformation as a function of distance to a point.

Click two points on a View window to define a circle. The radius of the circle defines how far the deformation is effective. Distance to the center of the circle is calculated for each point to get a deformation parameter. The control curve is then evaluated by the deformation parameter to get the amount of deformation. Click two points on a View window to define the scaling center and scale. Move the mouse to deform and click to accept the tool.

The Scale3D tool has the following options:

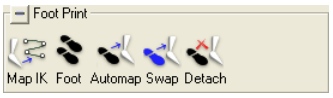


- Target - Transformation target.
  - Native
  - Geometry
  - LatticeCoords
  - Handles

  - Texture
- Ctrl Curve - Shape of deformation parametrization
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- Collision Deformation - Activate collision detection system.
- About Pivot - Scale object about the pivot point.

Foot Print Tools

The Foot Print tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the foot print tools.



These tools can be drag&dropped to a tool bar.

Automap



Map hierarchical skeleton to foot print object.

Select one or more skeleton joints and one I.K. object and then accept to tool.

Detach



Detach skeleton from a foot print object.

Select the skeleton and the foot print object and accept the tool.

Foot



Create foot prints defining an I.K. path.

Enter points for the foot prints and accept when done.

Map IK



Map skeleton to I.K. Path.

Select one or more skeleton joints and one I.K. object. Select Accept to map skeletons to the I.K. object.

Swap



Attach skeletons to a new foot print object.

Select the skeleton and the new foot print object and then accept the tool.

Geometry Manager Tools

3D Measure



Creates a 3D measure object, which measures surface area and volume of its sub objects.

To use the tool:

- Select objects to be measured
- Click the tool button

You can also create the measure object first without object selection and later drop objects into its hierarchy.

See also: 3D Measure Object.

Angle Measure



Creates an Angle Measure object, which displays the angle between its two control lines.

To use the tool:

- Click the tool button
- Click in a view window to define the origin of angle
- Click second time to define the first control line
- Click third time to define the second control line. This finishes the tool.

See also: Angle Measure Object.

Measure



Creates a Measure object, which displays the distance between two points.

To use the tool:

- Click the tool button
- Click in a view window to define the start point
- Click second time to define the end point. This finishes the tool.

See also: Measure Object.

Packer



Packer Tool creates a geometry manager object for packing and aligning objects. The tool has one control option: Orientation. When Orientation is set to Detect, the tool finds out the axis along which the pivot points of the selected objects have been aligned, and packs the selected objects to that axis. The user can also explicitly select one of the packing directions X, Y and Z.

After the tool is started, the user can define the direction and size of the packer box with two mouse clicks. The first click position defines the corner, which does not move when packer dimensions are later changed. If the tool is accepted before the box is drawn, the packing directions are taken from the first selected object.

Position Mark

Creates a Position Mark object, which displays the 3d coordinates of the control point.

To use the tool:

- Click the tool button
- Click in a view window to define the position. This finishes the tool.

See also: Position Mark Object.

Replicator



The tool creates a replicator object, which automatically repeats its sub objects to fill its control box. An example: select a kitchen cabinet and create a replicator. Stretch the replicator to get an entire row of kitchen cabinets.

The Orientation option of the tool works the same way as in the Packer tool above.

View Control Tools

This tool set provides the tools for controlling various View Window specific functions. The tools can be drag&dropped to a tool bar.

Assign



Assign view's orientation to the selected objects. Usually this tool is used for recording a view to a camera object, but you can also set spot light orientation this way, for example.

Boxren



Define Render Box and enable box rendering. When enabled, the user can LMB drag a rectangle whose contents will be rendered. All later renderings will be limited to the defined are until the render box is deactivated.

Fetch



Fetches the view camera orientation from the selected object. The view will be moved to its position and rotated to match its object space. For example, you can take the view from a camera object or from a spot light this way.

Fit



The autofocus tool. The view camera is aimed at the center of the selected objects and zoomed/moved so that the selected objects fill about 75% of the view window. Also the clipping range is adjusted so that all selected objects become visible.

Shift-clicking the tool focuses on the selected edit handles, if the selected objects are set to editing state.

Front



This tool puts the view camera to the front view orientation. The camera will look against z axis (heading 180 degrees, pitching zero). The camera is rotated to the front view about its aimpoint. Also the input plane is rotated so that its normal matches the z axis. Shift clicking the icon gives the opposite view (the back view).

Grid



Enables and disables grid drawing for the currently selected grid. To select a grid, drag it from the select window's grid tab and drop it into the view window.

IP by 3P



Reads three points and fits the input plane to match the defined plane. The line between the two first points will become the input plane's x axis. For example, you can fit the input plane to any triangle or rectangle by applying this tool: activate the tool and then LMB drag three corner points with the depth modifier key ',' (comma) held down.

IP by N



Fits the input plane to surface tangent. Activate the tool and LMB click over a surface. You may also LMB drag (preferably grid drawing on) to see how the input plane rotates; releasing the mouse button sets the input plane.

Persp



Sets perspective projection for the view on/off. The parallel camera projection is often more suitable for accurate modeling purposes.

Postproc



Post process the the current view window. The tool loads the previously rendered image and applies the post effects to it.

This tool works only if the view window has 'Backup Ray Trace' option set.

Quad



The default user interface in Realsoft 3D supports single and quadview modeling environments. This tool can be used for toggling between the two.

Render



Render the current view using the photo realistic renderer. The View is rendered according to the current rendering settings. You can set rendering settings for a view window by dragging and dropping the desired rendering settings frmo the select window into the view window, or by using the view property window.

Reset



Resets the view back to the 'Native state' that has been stored into it. This usually changes camera and input plane positions and orientation, as well as the zoom scale.

Shaded



A tool for managing OpenGL rendering. Allows the user to toggle between shaded and wireframe opengl modes. If the view window uses some other than OpenGL rendering, then the tool appears in conflict state and clicking it will change the View's rendering mode to OpenGL.

A plain click toggles Wireframe and Wire+Shaded modes. Shift-click selects the two remaining modes, plain Shaded and Hidden line drawing.

Side



The side view tool. Side view is the view from the left side of the object (along positive x axis). The side menu is similar to the front view above. Shift clicking the icon gives the opposite view (the right side view).

Snap



Enables and disables grid snapping. A grid must be also assigned for the view, otherwise snapping does not work. You can assign the grid by drag and dropping it from the select window into the view window, or using the view property window.

Top



The top view tool. View camera and input plane are rotated so that the camera looks down against y axis. See the front view tool above for the side menu explanations. Shift clicking the icon gives the opposite view (the bottom view).

Track



Set view window to camera tracking mode. The view window will track the current camera during animation playback.

Walk Thru



Walk Thru animation tool.

The tool allows you to navigate in a scene. The navigation can be recorded and used in the following ways:

- Create a camera out of the recorded data
- To attach the recorded motion to the selected objects

The tool uses the mouse and keyboard input to control the orientation of the camera similar to many games: moving the mouse controls the direction and the four keys a, w, d and s allow you to move forward, bacward and sideways. Collision detection system is used for automatic floor and wall detection.

To use the tool, press down the mouse in a view window. This will activate the navigation for the window and you can then control the orientation of the camera by moving the mouse.

The keys 'w', 'a', 'd' and 's' can be used for walking forward, left, right and backwards. The mouse wheel controls the focal lenght (zoom) of the camera.

The tool supports large number of options, well known from real world cameras.

Options

Action	Controls whether the tool creates an animated camera or animates the selected objects.
Walk	Set the maximum walking speed (m/s). The Shift key can be used for slowing the velocity down. The actual speed can also be slower when approaching walls: the tool automatically slows the user down near walls to give the user more time to react.
Turn	Controls the rotation speed.
Height	The height of the Observer (altitude). The camera will be elevated this much above the ground.
Threshold	Controls the key generation in motion recording. This value is relative to the defined turning and walking speed. The value of 0 means the motion is always recorded, no matter how much the camera moves or turns. This will result lots of key frames. The value of 1.0 means the motion is practically recorded only when the camera rotates full 360 degrees, or moves more than the given walking speed.
Gravity	Gravity on/off. If gravity is off, the user will be able to climb on the walls and fly.
Collision	Enable or disable Collision Detection. If the option is set, then the tool will detect floor and walls and it is not possible to pass through any surfaces.
Stabilizer	Image stabilizer. Smooths out shaking and sudden jumps.
Auto Focus	Auto Focus. Sets focusing distance automatically by detecting the distance to the nearest surface in front. When enabled, the tool renders a focusing frame representing the users dimensions on the scene. Autofocus option is useful when rendering animations with Dept-of-Field effect.
Input Plane	Manage Input Plane orientation. The orientation for the input plane is determined automatically from the surface in front of the camera. If the Input Grid is not shown, then the tool renders a small rectangle to visualize the input plane. This option is useful when modeling new objects to the scene: just walk thru the scene and create objects while you go.
Record	Enable camera motion recording. When set all navigation actions are recorded.
Play	Preview the recorded animation data.

Zoom



This is a combined pan and zoom tool. To use it: Activate the tool. Then LMB drag a rectangle on a view window. Note how the dragged rectangle has automatically the proportions of the view window. After left-to-right dragging, the defined rectangle area is magnified to fill the entire view. Reversed dragging from right to left will shrink the current view area to fit the defined rectangle.

To just pan, without zooming, just click the mouse button once, without dragging.

Lattice Mapping Tools

Lattice mapping system is based on the idea that all geometric objects define three dimensional parameter space (also called uvw space). For example, the 'u' coordinate for the nurbs curve runs along the curve whereas the two other dimensions are perpendicular to the curve.

The parameter space for the nurbs mesh is such that the first two dimensions corresponds to the parameter space of the nurbs mesh whereas the third dimension corresponds to the surface normal of the mesh.

Any object can be mapped into this parameter space, in which case the coordinates of the mapped object are defined in the parameter space of the lattice object rather than in the world space. When the lattice object is then modified, the mapped geometry will get modified accordingly.

Lattice mapped objects can be animated. For example, a sphere moving in 'x' direction in the parameter space of the curve means that the sphere moves along the curve.

Available GUI Objects

The Lattice Mapping tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the lattice mapping tools.



These tools can be drag&dropped to a tool bar.

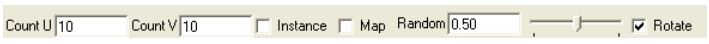
Distribute



Distribute over surface.

Select the target object and the lattice object. Define the Distribute tool options through the Control Bar and accept the tool via the View's pop-up menu or by clicking the Accept button on the Control Bar. This distributes copies or instances of the target object over the surface of the skeleton object.

The Distribute tool has the following options:



- Count U - Number of points is 'u' direction.
- Count V - Number of points is 'v' direction.
- Instance - If set, instances rather than copies.
- Map - If set, both maps and distributes, otherwise just distributes.
- Random - Randomize.

- Rotate - If set, rotates objects according to the surface normal.

Rotate



Make the first selected objects rotate about the last selected 'axis' object.

Select two objects: the object to be rotated and the rotation axis. Define rotation options through the Control Bar and accept the tool via the View's pop-up menu or by clicking the Accept button on the Control Bar. This creates a rotation animation.

The Rotate tool has the following option:



- Angle - Define rotation angle.

Map



Set up lattice mapping.

Select two or more objects. All but the last selected objects are mapped to the parameter space of the last selected object. Set mapping options through the control bar and accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Map tool has the following options:



- Mapping mode - Defines how the target object should be mapped.
  - Object
  - Pointwise
  - Selected Points
  - Look At point
- Weight - Defines weight for mapping. This is needed when mapping a point to multiple skeletons.

Multi Map



Map selected objects to lattice objects utilizing multi mapping.

Select two or more objects. The first selected objects are mapped to the parameter space of the last selected object. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

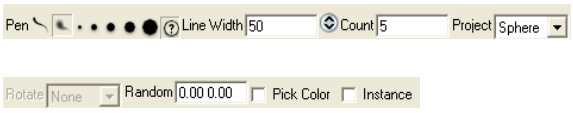
Paste

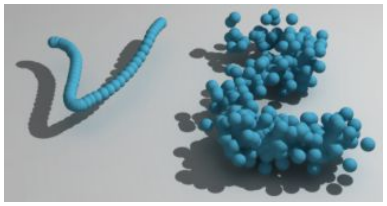


Copy and paste selected objects to a surface or into the 3D space.

Select the object(s) to copy. Drag over the desired positions with the left mouse button held down to create copies or instances.

The Paste tool has the following options:

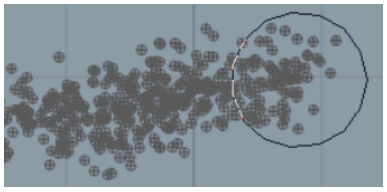




Pasting with a sharp pen and with an airbrush

- Pen - paste single copies accurately or airbrush randomized object sets.

- Line Width - the radius of the airbrush pen measured as pixels. The brush is drawn as a circle when the mouse moves over a view window.
- Count - the number of object copies for the airbrush pen. The higher the value, the denser object cloud is pasted into the scene.



Brush width is shown as a circle

- Project - defines how the mouse pointer position and created object copies are projected into 3D space:
  - Sphere - objects are placed into a spherical volume around the mouse pointer coordinates on the input plane.
  - Disc - objects are placed into a planar disc on the input plane.
  - Surface - objects are projected into the underlying surfaces.



Comparison of spherical, flat disc and surface projected brushes

- Rotate - defines how surface-projected object copies are oriented:
  - None - the original orientation is kept. Only the position changes.
  - Normal - the z axis of the object space (the blue axis) is rotated to match the surface normal.
  - Reflect - the z axis of the object space is rotated to match reflected directions.



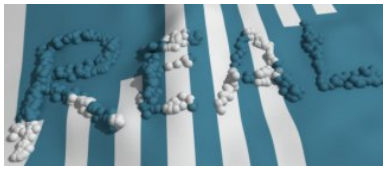
Hair brushed to three spheres from above using a fixed direction, surface normal's direction and reflected direction

- Random - randomizes the orientation of pasted copies. The first random value component controls how much the z axis direction of pasted copies is randomized. The second component randomizes the rotation around the z axis. For example, when pasting a tree object, the trunk should always point up, but the tree can rotate freely around its trunk. This can be achieved with a value pair "0 1".



A set of objects brushed spherically using a strongly randomized direction

- Pick Color - if checked, the color of the projection surface defines the color for created objects.
- Instance - paste instances instead of real object copies.



Underlying surface controls the color of pasted objects



Note

By holding down the Ctrl key while dragging, you can paste objects directly to the scene so that they start interacting with the tool. The tool updates the surface hit system every time you release the left mouse button. This means that by holding down the Ctrl and by Surface mode brushing objects with short LMB clicks, you can build cumulative piles of pasted matter. It is an interesting modeling approach and fun to play with.

Path



Make the first selected objects follow last selected 'animation path' object.

Select two or more objects. The first selected objects will be moved along the last 'path' object. Define path options through the Control Bar and accept the tool via the View's pop-up menu or by clicking the Accept button on the Control Bar. This creates a path keyframe animation.

The Path tool has the following options:



- Target - Defines how the target object should be mapped.
  - Objects - The target is mapped as a rigid object
  - Points - Points of the target are mapped to the path. Path will deform the target.
  - Selected Points - same as above, but only for the selected points of the target object.
- Weight - Defines weight for mapping. This is needed when mapping a point to multiple skeletons.
- Translate - If set, the lattice object translates the target objects.
- Rotate - If set, the lattice object rotates the target objects.
- Scale - If set, the lattice object scales the target objects.
- Skew - If set, the lattice object skews the target objects.
- Method - controls how the path rotates the target object.
  - Torsion from the path - The target is rotated to follow the curve using the shortest rotation route between each two consecutive positions. This usually banks the target i.e. rotates it around the curve. Banking cumulates towards the end of the path.
  - No banking - A horizon axis is estimated from the path curve geometry, and all banking with respect to the horizon is eliminated.
  - Natural banking - The target is rotated to match the normal of the curve in non-cumulative manner.
  - Cumulative banking - Same as Torsion from the path, but with a path level construct.
  - Aeroplane Banking - A horizon axis is estimated from the curve. The banking behavior simulates aeroplane. The length of the horizon axis under the path level controls the amount of banking. A very long axis will eliminate banking.
- Constant Speed - If set, creates a constant speed path. Otherwise the target will accelerate and decelerate according to the point density of the curve.



Note

The path tool estimates the horizon plane from the curve geometry. If the curve is not planar, the estimate may be not what was intended. In that happens, just unmap the target, edit the horizon axis and map the target again to the path level, but this time with the default path options (torsion from the path, no constant speed) to avoid nested path levels.

Reconnect



Detach target object from existing lattices (skeletons) and attach it to newly selected objects.

Select the target object and new lattice objects for the target. Select Accept to detach the target objects from their original lattices and attach them to the selected lattice objects.

Unmap



Break mapping binding between the selected objects.

Select one or more target objects and the lattice object. Set Unmap tool options through the control bar. Select Accept to unmap objects from the lattice object.

The Unmap tool has the following options:

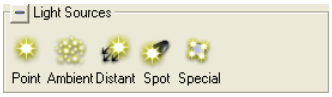


- Unmap - Define whether the tools should unmap only selected points or the entire object.
- Sub lattices - If set, also sub lattices are unmapped.

Light Source Creation Tools

Available GUI Objects

The Light Sources tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the light source creation tools.



These tools can be drag&dropped to a tool bar.

Ambient Light



Create an ambient light source.

Select Accept to create the light source to the camera position, or click the left mouse button in the View window to define the position for the light source. Move the mouse to define the light fall-off radius and click to accept.

See also: Models/Geometrics/Light Sources/Ambient Light.

Distant Light



Create a distant light source.

Click in a View window to define the first point for the light direction. Move the mouse to define the light direction and click to accept.

See also: Models/Geometrics/Light Sources/Distant Light.

photon map Tool

Create a photon map light source. The tool evaluates the illumination in the current scene and stores it to a photon map object. The scene must include ordinary light sources which cast photons around.

The tool has the following options:

- Settings - render settings to define how accurately photon bounces are traced. Recursion depth and threshold play an important role here. Rendering settings have a dedicated section in the misc tab for controlling photon mapping. For example, the desired photon count is defined there.
- Include Direct Photons - A special option to capture direct illuminaion from light sources. Usually this is not desirable as direct illumination is faster and more accurate to compute using ray tracing. When using this option, original lights should probably be erased; otherwise the photon map will duplicate their illumination.



Photon mapping applied to an architectural scene. Courtesy of Pixel Perfect.



Note

Usually photon mapping works best in closed interior scenes, where all traced photons contribute into the overall illumination. For other kind of scenes, use of spot lights is recommended; spot lights emit the photons efficiently along the defined light cone. This can speed up photon map rendering significantly. An example of a challenging photon map setup is a scene, where small objects are placed around a point light. In such a scene, 99.99% of photons can end up into empty space, creating nothing visible but just slowing down the rendering.

See also: Models/Geometrics/Light Sources/photon map and Tutorials/Rendering/photon map Illumination.

Point Light



Create a point light source.

Select Accept to create the light source to the camera position, or click the left mouse button in the View window to define the position for the light source. Move the mouse to define the light fall-off radius and click to accept.

See also: Models/Geometrics/Light Sources/Point Light.

Special Light



Create a special light source. Any geometry inside the special light source object radiates light.

Select the objects to be converted into light sources. Then click this tool.

See also: Models/Geometrics/Light Sources/Special Light.

Spot Light



Create a spot light source.

Click the left mouse button in a View window to define the position for the light source. Move the mouse to define the light direction and click. Move the mouse and click to define the inner angle for the light cone. This is the angle in which the light sources casts rays of maximum brightness. Specify the outer angle for the light source. The brightness falls off with the angle and corresponds to zero with this angle.

The Spot Light tool has the following options:

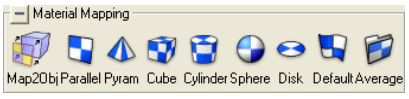


- View - The Side view option defines the conical shape of the spotlight from the side. The Front view option places the spotlight exactly over the view camera. The user can define the inner and outer light cones by viewing along the spotlight axis.
- Material - Material defining volumetric properties

See also: Models/Geometrics/Light Sources/Spot Light.

Material Mapping Tools

The Material Mapping tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the material mapping tools.



These tools can be drag&dropped to a tool bar.

Average



Create 'Average map' by blending multiple materials with Scope channel.

Activate the tool. Select two or more mapping object to be averaged. Click Accept on the Control Bar or select Accept on the View window popup menu. The selected mapping objects are moved inside an average mapping level.

Cube



Create a cube mapping. Allows you to map six materials, one for each side.

Activate the tool. Specify options in the Control Bar. Define the mapping geometry (cube) by clicking two points on a View window. The materials for individual faces of the cube mapping can be set through the Property window.

The Cube Mapping tool has the following options:



- Define - How the mapping geometry is defined, either by two corners or center and corner.
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.
- Environment Map - If set, creates an environment map.

See also: Models/Geometrics/Material Mapping Objects/Cube Mapping.

Cylinder



Map a material to an object using cylindrical mapping.

Activate the tool. Define the mapping geometry (cylinder) by clicking two or three points on a View window depending on the Define option.

The Cylinder Mapping tool has the following options:



- Define - How the mapping geometry is defined.
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.

See also: Models/Geometrics/Material Mapping Objects/Cylinder Mapping.

Default



Map a material to an object using default mapping.

Activate the tool. Specify options in the Control Bar. Click Accept on the Control Bar or select Accept on the View window popup menu.

The Default Mapping tool has the following options:



- Material - Define material to be mapped.
- Mapping Type - Mapping method to use:
  - Surface - the material effect is mapped to the surface space of the actual geometry to be textured.
  - Identity - this method defines the material effects in identity space.
  - Null - this mapping method uses the current material space, as defined by the previous material mapping objects.
  - Camera - uses the current camera orientation to project material properties to geometric objects.
  - Custom - moves a selected channel (a vertex variable etc.) to the material mapping coordinate channel.

See also: Models/Geometrics/Material Mapping Objects/Default Mapping.

Disk

 Map a material to an object using disk mapping.

Activate the tool. Specify options in the Control Bar. Define the mapping geometry (cylinder) by clicking two or three points on a View window depending on the Define option.

The Disk Mapping tool has the following options:



- Define - How the mapping geometry is defined.
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.

See also: Models/Geometrics/Material Mapping Objects/Disk Mapping.

Map2Obj

 Convert mapped material properties to object properties.

Select one or more objects and one material mapping object. Activate the tool. Specify options in the Control Bar. Click Accept on the Control Bar or select Accept on the View window popup menu.



The Map2Obj tool has the following options:

- Target - Define properties of objects, points or selected points.
- Points - Define pointwise property for all geometry points.
- Selected Points - Define pointwise property for the selected geometry points.
- Objects - Define property for the object.
- Source - Source channel (material property) to be evaluated.
- Destination - Destination channel to be defined.
- Destination Material - Destination face material.

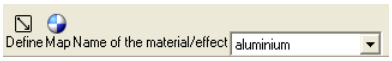
Parallel



Map a material to an object using parallel mapping.

Activate the tool. Specify options in the Control Bar. Define the mapping geometry (cube) by clicking two points on a View window.

The Parallel Mapping tool has the following options:



- Define - How the mapping geometry is defined, either by two corners or center and corner.
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.

See also: Models/Geometrics/Material Mapping Objects/Parallel Mapping.

Pyramid



Map a material to an object using pyramid mapping.

Activate the tool. Specify options in the Control Bar. Define the mapping geometry (pyramid) by clicking two points on a View window.

The Pyramid Mapping tool has the following options:



- Define - How the mapping geometry is defined, either by two corners or center and corner.
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.

See also: Models/Geometrics/Material Mapping Objects/Pyramid Mapping.

Sphere



Map a material to an object using spherical mapping.

Activate the tool. Specify options in the Control Bar. Define the mapping geometry (sphere) by clicking two points on a View window.

The Sphere Mapping tool has the following options:

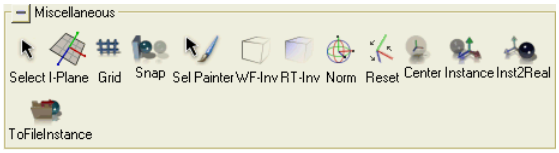


- Input - How the mapping geometry is defined.
  - Center & Radius
  - Corner & Corner
  - Center & Corner
- Map - Type of class to map. Either Material of Effect.
- Name of the material/effect - Define material or effect to be mapped.
- Environment Map - If set, creates an environment map.

See also: Models/Geometrics/Material Mapping Objects/Sphere Mapping.

Miscellaneous Tools

The Miscellaneous tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains some miscellaneous tools.



These tools can be drag&dropped to a tool bar.

Accept



Accept the active tool.

Cancel

Cancel the active tool.

Center



Moves the pivot point of selected object to the center of the bounding box.

Edit



Toggle edit mode of the selected objects.

In edit mode the selected objects reveal their object specific handles allowing the user to activate object specific tools by dragging them. If edit mode is disabled, general transformation handles common to all geometric objects are shown.

Delete



Delete the selected points.



Note

Not all the geometric objects support this tool. Many geometric objects define a fixed set of vertices.

Grid



Mofify the current grid or create new grids.

Select a grid and then enter two points on a View window to define the grid size and position. If you only want to change the grid position, click one point on a View window and then select Accept.

The Grid tool has the following option:

- Grid - List of available grids.

I-Plane



Define the input plane.

Define a rectangle representing the input plane by clicking two points in a View window.

The I-Plane tool has the following option:

- Define - Method to be used for defining geometric data for the rectangle.
  - Corner-Corner - Define by two opposite corners
  - Center-Corner - Define by center and corner

Inst2Real



Converts all instances and file instances to their real counterparts. The tool works recursively; you can select the root level of the hierachy and apply the tool to convert all instances in the scene. It is naturally possible to convert a sub level or a single instance as well.



Note

The tool loads in all materials of the file instances, which it converts, and indexes the loaded materials to achieve unique names. The tool does this to avoid material name conflicts. For example, a converted file instance of a window object can include a material called 'glass' with regular glass properties, and another converted file instance of a wine glass may include another kind of glass material, with the same name but with a higher refraction index (crystal glass). Both object use the correct glass after conversion.

Invert



Invert point selection: make selected points unselected and unselected points selected.

Normalize



Normalize Object Space. The tool transforms the object space matrix (or desired components of it) to identity state. The geometric points are transformed with the inversion matrix so that their positions in the world space remains unchanged.

This tool is useful when one needs to animate transformed objects. Scaling and skewing is often introduced as a result of a normal modeling work flow. However, in animations, objects are easier to manage if their object spaces are world oriented.

The tool can be applied selectively to any of the following single transformation components:

- Translate
- Scale
- Rotate
- Skew

Orthogonal



Makes x,y and z dimensions of selected 3D particles perpendicular to each other.



Particles before and after Orthogonalization

Reset



Reset object handles to default.

Roll



Reorders XYZ directions of 3D particles. X becomes Y, Y becomes Z and Z becomes X axis. This has a visual effect when rendering 3D particle quads, because the quad surface is defined by the XY plane of each particle.

RT-Invisible



Toggle Ray Tracing invisibility of the selected objects.

Select

Select objects and their handles through the View windows.

You can make a selection by clicking the object or handle, by the drag box, or by the lasso selector. To use the drag box press the left mouse button down outside the object or handle and move the mouse so that the object or handle falls inside the drag box and then release the left mouse button. To use the lasso selector press down and hold the '4' key above the letter keys, press down the left mouse button and draw a freeform curve around the object or handle to select. The drag box and the lasso selector can be used to select multiple objects or handles. The object and handle selection modifiers are:

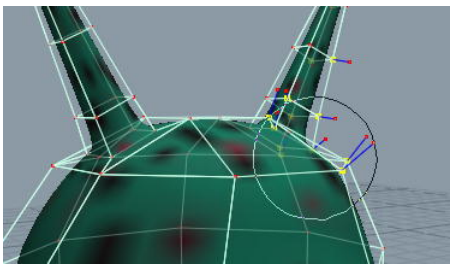
- Shift - Add to selection (OR)
- Ctrl - Intersection with previous selection (AND)
- Shift Ctrl - Invert selection (XOR)
- Alt - Subtract from selection
- 4 - Lasso selector in dragging

Selection Painter

Selection Painter is a tool which selects object editing handles (points, edges and faces) in a painting like manner. Selection happens simply by LMB-dragging a circular brush over the targets.

The tool can be used also for selecting entire objects. Object/handle painting is detected automatically from the already selected objects. If all selected objects are in editing mode, handle selection is painted. Otherwise (also when there is no selected objects yet), object selection is painted. Object selection painting is a useful feature when dealing with a large number of objects, such as curve modeled hairs.

The tool obeys the standard selection modifiers: Alt removes from the current selection and Shift adds points to the current selection.



Selecting point handles by painting

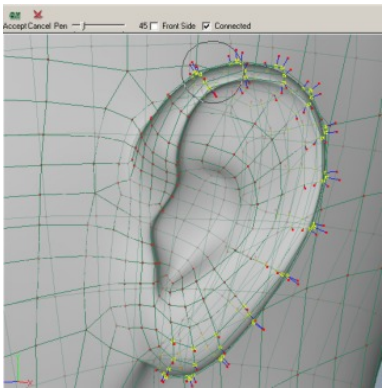
The size of the circular selection brush can be adjusted from the toolbar or with the mouse wheel.

The default shortcut for selection painter is Alt-Spacebar.

The Selection painter tool has an option for front side selections. The feature uses true geometric visibility to ensure that flipped SDS normals or open NURBS surfaces do not cause any problems.

The tool also supports connected selections. The connectivity feature limits the selection only to the neighborhood of already selected handles. Selection does not jump to disconnected surfaces that happen to be behind the desired target area. For example: select a tip of a finger and spread selection down the finger - other fingers will not interfere.

The Gradient option allows painting of soft selections. When enabled, the associated curve defines the distribution of the selection weight along the radius of the circular brush.



Connected selection selects the back side of the ear but not the skull behind the ear

Simulation



When a level object or an object derived from the level class is selected, the Simulation tool button appears on the Control Bar. You can activate and turn off simulation animation for the level.

Snap



The Snap tool snaps the selected objects to other objects. Snapping is based on ray/surface hit testing. Hit invisible objects, material maps and objects without 2D surface are not considered in snapping.

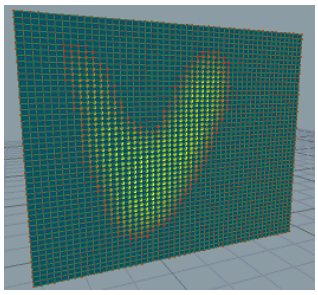
The snap tool options are the same as in the property gadget of the packer object (see Models/Geometrics section).

Soften



This tools softens handle selection values at the edges of the selection.

The tool is based on topological connections and therefore works only on geometric surfaces and curves where connectivity can be examined (NURBS, SDS, triset). By applying the tool multiple times, a wider softening can be achieved.



A hard edged lasso selection softened by applying the Soften tool multiple times

To File Instance



This tool converts the selected object(s) to a file instance object. Use the tool as follows:

- Select the desired object and start the tool.
- In the tool control bar, define the file name for saving the objects and geometry display method (bounding box/ accurate wireframe).
- Click Accept.



Note

The tool also collects all referred materials and saves the respective material chunck.

Volume Selector

Select points or objects by volume.

WF-Invisible

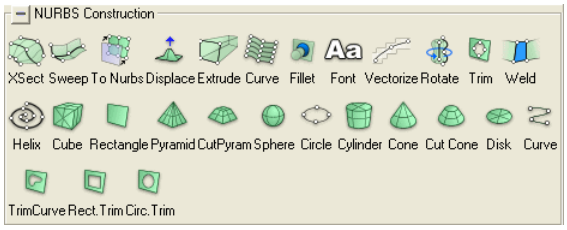


Make the object invisible in real time drawing (shaded OpenGL or wireframe).

NURBS Curves and Surfaces

NURBS Construction Tools

Available GUI Objects



The NURBS Construction tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the NURBS construction tools.

These tools can be drag&dropped to a tool bar.

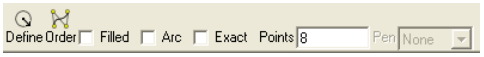
Circle



Create Nurbs circle.

Specify options in the Control Bar. Define the circle by clicking two or three points on a View window depending on the Define option.

The Circle tool has the following options:



- Define - How the circle is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the NURBS curve.
- Filled - If checked creates a disk, otherwise created a closed circular curve.
- Arc - If checked creates an arc.
- Exact - If checked creates an exact circle.
- Points - number of points along the circle.
- Density - number of extra curves.

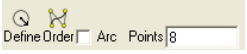
Circular Trim Curve



Create circular trim curve.

Specify options in the Control Bar. Define the circular trim curve by clicking two or three points on a View window depending on the Define option.

The Circular Trim Curve tool has the following options:



- Define - How the circle is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the trim curve.
- Arc - If checked creates an arc.
- Points - number of points along the circle.

Cone



Create Nurbs cone.

Specify options in the Control Bar. Define the bottom circle of the cone by clicking two or three points on a View window depending on the Define option. Define the apex point by clicking on a View window.

The Cone tool has the following options:



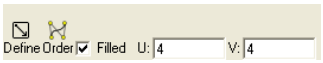
- Define - How the cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the NURBS surface.
- Filled - If not checked the top and bottom lids of the cone are not created.
- Arc - If checked creates a cone sector.
- Center - If checked, the apex will be centered over the bottom.
- Points - number of points along the bottom circle.
- Density - number of extra curves.

Cube



Creates a Nurbs cube. Specify options in the Control Bar. Define the cube by clicking two points on a View window.

The Cube tool has the following options:



- Define - How the cube is defined, either by two corners or center and corner.
- Order - Order of the NURBS surface.
- Filled - If not checked the top and bottom face of the cube are not created.
- U - number of points in the width direction.
- V - number of points in the height direction.

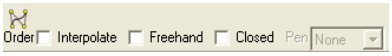
Curve



Create Nurbs curve.

Specify options in the Control Bar. If the Freehand option is set, then press down the left mouse button on a View window and draw the curve while holding the left mouse button down. Otherwise click a number of points on a View window and accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Curve tool has the following options:



- Filled - If checked creates a surface whose boundary is the defined freeform curve.
- Order - Order of the Nurbs curve or surface.
- Interpolate - If set the curve passes through the defined points.
- Freehand - Define curve by moving the mouse while holding down the left mouse button.
- Closed - Define closed curve or surface.
- Pen - When a Nurbs curve is selected define how a surface is created.
  - None - Creates a curve.
  - Sweep - The selected pen curve will be swept along the curve drawn. The pen object is rotated according to the curvature of the curve drawn.
  - Rail - The selected pen curve will be swept along the curve drawn. The pen object is not rotated while sweeping.
  - Rotate - Uses the pen object as an axis and rotates the new curve around it.

Cut Cone



Create Nurbs cut cone.

Specify options in the Control Bar. Define the bottom circle of the cut cone by clicking two or three points on a View window depending on the Define option. Define the top circle by clicking two points on a View window.

The Cut Cone tool has the following options:



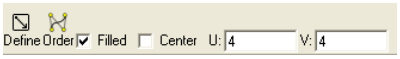
- Define - How the cut cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the NURBS surface.
- Filled - If not checked the top and bottom lids of the cut cone are not created.
- Arc - If checked creates a cut cone sector.
- Center - If checked, the top will be centered over the bottom.
- Points - number of points along the bottom circle.
- Density - number of extra curves.

Cut Pyramid



Create Nurbs cut pyramid. Define the bottom rectangle of the cut pyramid by clicking two points on a View window. Define the top rectangle point by clicking two more points on a View window.

The Cut Pyramid tool has the following options:



- Define - How the cut pyramid is defined, either by two corners or center and corner.
- Order - Order of the NURBS surface.
- Filled - If not checked the top and bottom face of the cut pyramid are not created.
- Center - If checked, the top will be centered over the bottom.
- U - number of points in the width direction.
- V - number of points in the height direction.

Cylinder



Create Nurbs cylinder.

Specify options in the Control Bar. Define the cylinder by clicking two or three points on a View window depending on the Define option.

The Cylinder tool has the following options:



- Define - How the cylinder is defined. The choices are:

  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the NURBS surface.
- Filled - If not checked the top and bottom lids of the cylinder are not created.
- Arc - If checked creates a cylinder sector.
- Center - If checked, the top will be centered over the bottom.
- Points - number of points along the bottom circle.
- Density - number of extra curves.

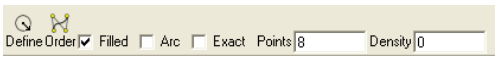
Disk



Create Nurbs disk.

Specify options in the Control Bar. Define the disk by clicking two or three points on a View window depending on the Define option.

The Disk tool has the following options:



- Define - How the Disk is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Order - Order of the NURBS curve.
- Arc - If checked creates an arc.
- Points - number of points along the disk perimeter.
- Density - number of extra curves.

Displace



Displace the selected Nurbs meshes with a bit map image.

Select the NURBS mesh you want to displace. Specify the displacement height and the image file in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Displace tool has the following options:



- Displacement - Defines the displacement height for the selected meshes.
- Image - Select bitmap file to be used for defining displacement.

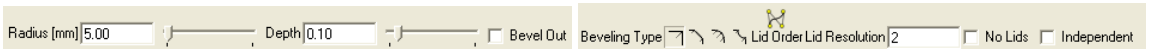
Extrude



Extrude selected curves with given bevelling. Creates an Extrusion Manager object.

Select the curve or curves to be extruded. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Extrude tool has the following options:



- Radius - Radius (size) of the beveling in millimeters.
- Depth - Extrusion depth.
- Bevel Out - Bevel in/out.
- Beveling Type - Defines how the top and bottom lids and the side mesh are welded with each other.
- Lid Order - NURBS order of the lid.
- Lid Resolution - Number of points in the top and bottom lids.
- No Lids - No top or bottom lids.
- Independent - Treat each selected object independently.

Fillet



Construct a fillet between surfaces, Nurbs curves, or trim curves.

Select two NURBS surfaces, or two or more NURBS curves. To build a fillet from the an open ended cylindrical mesh pointing to another mesh, select the target mesh first, then the cylindrical mesh. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Fillet tool has the following options:



- Trim - If set, created trim curves cut their parent surfaces.
- Radius - Fillet radius.
- Order - Order of the fillet.

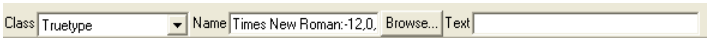
- Resolution - Number of points in the 'U' direction of the fillet.

Font

Create Nurbs curves from outline fonts.

Select the font class. Set the font class specific options like the font name or font file. Type in the text to create font curves from. Click two points on a View window to define the baseline and size of the curves.

The Font tool has the following options:



- Class - Class of the font.
- Text - Enter text to be created.
- Render - Create rendering visible curves.

Note that each font class can have its own options.

Helix

Create helix curves.

Set the helix type and the number of points on the Control Bar. Click the first point to define the center of the helix. Click the second point to define the starting direction of the helix curve. Move the mouse around the center point to define the shape of the helix curve and click to accept the tool.

The Helix tool has the following options:



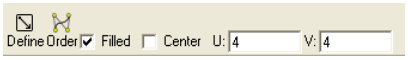
- Type - Type of the curve to be created: Disk, Cone, or Cylinder.
- Count - Number of points in the curve.

Pyramid

Create Nurbs pyramid.

Specify options in the Control Bar. Define the bottom rectangle of the pyramid by clicking two points on a View window. Define the apex point by clicking on a View window.

The Pyramid tool has the following options:



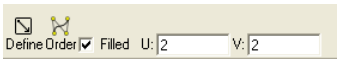
- Define - How the pyramid is defined, either by two corners or center and corner.
- Order - Order of the NURBS surface.
- Filled - If not checked the bottom face of the pyramid is not created.
- Center - If checked, the apex will be centered over the bottom.
- U - number of points in the width direction.
- V - number of points in the height direction.

Rectangle



Create Nurbs rectangle.

Specify options in the Control Bar. Define the rectangle by clicking two points on a View window. The tool has the following options:



- Define - How the rectangle is defined, either by two corners or center and corner.
- Order - Order of the NURBS surface.
- Filled - Whether to create a surface or a border line.
- U - number of points in the width direction.
- V - number of points in the height direction.

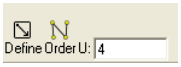
Rectangular Trim Curve



Create rectangular trim curve.

Specify options in the Control Bar. Define the rectangular trim curve by clicking two points on a View window.

The Rectangular Trim Curve tool has the following options:



- Define - How the trim rectangle is defined, either by two corners or center and corner.
- Order - Order of the trim curve.
- U - Number of vertices in the width (u) direction.

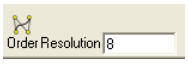
Rotate



Construct rotational nurbs surface by rotating the second curve about the first curve.

Select the NURBS curve to be used as an axis. Select the NURBS curve to be used as a profile curve. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Rotate tool has the following options:



- Order - Order of the rotation NURBS surface.
- Resolution - Number of points in the 'U' direction of the rotation surface.

Sphere



Create Nurbs sphere.

Specify options in the Control Bar. Define the sphere by clicking two points on a View window.

The Sphere tool has the following options:



- Input - How the sphere is defined, by center and radius, two corners, or center and corner.
- rder - Order of the NURBS surface.
- W - number of points in the width direction (along the equator).
- H - number of points in the height direction (from pole to pole).

Sweep



The Sweep tool constructs a surface by moving a cross section curve along one or two rail curves.

Select the curve to be used as a cross section. Select one or two more curves to be used as rails. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Sweep tool has the following options:

- Orthogonal - Rotate the profile curve with the rail curve.

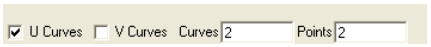
ToCurves



Create curves from surfaces of the selected objects.

Select the object to create curves from. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar. This creates a curve manager object whose attributes can be changed though the Property window.

The Curve tool has the following options:



- U Curves - Create curves in the 'U' direction of the parameter space.
- V Curves - Create curves in the 'V' direction of the parameter space.
- Curves - Number of curves to be created.
- Points - Number of points per curve.

ToNurbs



Convert selected objects to Nurbs surfaces of Bezier form.

Select the objects to be converted. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The To Nurbs tool has the following options:



- Density U - Number of Bezier patches per complexity unit in the 'U' direction.
- Density V - Number of Bezier patches per complexity unit in the 'V' direction.

Trim



Map the control points of a NURBS curve to the parameter space of a mesh.

Select one or more NURBS curves and one or more NURBS meshes. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar. The selected NURBS curves are projected onto the parent Nurbs mesh and corresponding trim curves are created on the NURBS meshes.

Trim Curve



Create Nurbs trim curve.

Select a NURBS mesh to trim. Specify options in the Control Bar. If the Freehand option is set, then press down the left mouse button on a View window and draw the trim curve over the NURBS mesh while holding the left mouse button down. Otherwise click a number of points on a View window and accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Trim Curve tool has the following options:



- Order - Order of the trim curve.
- Interpolate - If set the trim curve passes through the defined points.
- Freehand - Define the trim curve by moving the mouse while holding down the left mouse button.
- Closed - Define closed curve. Should be set.

Vectorize



The Vectorize tool reads an image file, traces out contours from the image and generates a set of NURBS curves which represent the traced contours.

Select the bitmap image file to be vectorized. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The Vectorize tool has the following options:



- Image - Bitmap file to be vectorized.
- Edge Filtering - This should be checked unless the bitmap is a line drawing.
- Threshold - Edge thershold value. The smaller the value, the more edges are detected.
- Fit to nurbs - Cubic nurbs curves are fitted to edge polylines.
- Density - Point density of created curves.

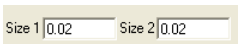
Weld



Weld single edges of two Nurbs meshes.

Select two NURBS meshes. Mark an edge from each mesh by dragging along the surface with the mouse. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar. A mesh is created to weld the selected edges together.

The Weld tool has the following options:



- Size 1 - Size of the weld at the first mesh.
- Size 2 - Size of the weld at the second mesh.

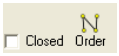
XSect



The Cross Section tool constructs a mesh through the selected curves.

Select two or more NURBS curves in the order you want them to appear in the NURBS mesh. Specify options in the Control Bar. Accept the tool by selecting Accept on a View window's popup menu or by clicking the Accept button on the Control Bar.

The XSect tool has the following options:



- Closed - Close the mesh in the 'V' direction.
- Order - Order of the NURBS surface.

NURBS Mesh Tools

Add



Insert new curve to the mesh.

Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position both U and V directed curves. Release the mouse button to accept.

Add U directed curve Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position a U directed curve. Release the mouse button to accept.

Add V directed curve Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position a V directed curve. Release the mouse button to accept.

Break



Break the NURBS mesh from the given point.

Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to break the mesh by both U and V directed curves. Release the mouse button to accept.

Break by U directed curve Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to break the mesh by U directed curve. Release the mouse button to accept.

Break by V directed curve Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to break the mesh by V directed curve. Release the mouse button to accept.

CloseU

 Toggle open/close the mesh in U direction.

Open in U direction Open the mesh in U direction.

Close in U direction Close the mesh in U direction.

CloseV

 Toggle open/close the mesh in V direction.

Open in V direction Open the mesh in V direction.

Close in V direction Close the mesh in V direction.

Concatenate

 Concatenate the selected two meshes.

Curves

 Convert mesh to NURBS curves.

Creates curves that have the same control points as the mesh.

U and V curves Creates both U and V curves from the control points of the mesh.

V curves Creates V curves from the control points of the mesh.

DefineUV

 Allocate and define texture UV coordinates for the mesh.

Uniform Allocate and define texture UV coordinates with uniform parametrization for the mesh.

Chord Length Allocate and define texture UV coordinates with chord length parametrization for the mesh.

Centripetal Allocate and define texture UV coordinates with centripetal parametrization for the mesh.

Delete

 Delete selected points.

Edit

 Choose how to select, either by points, curves of objects.

ElevU



Degree elevation in U direction.

Elevate with fixed shape The degree of the mesh is elevated by one. Elevation is done so that the shape does not change. For example, a linear mesh remains linear by its shape. However, you can later bend it smoothly because the higher degree allows it.

Elevate by smoothing the shape Degree elevation uses the control points to form as smooth shape as the higher degree allows.

ElevV



Degree elevation in V direction.

Elevate with fixed shape See U directed elevation above.

Elevate by smoothing the shape See U directed elevation above.

Fillet



Create fillet between two meshes.

Activates the Fillet tool.

Invert



Make the surface interpolate through the control points.

ReduceU



Reduce degree in U direction. Opposite to Degree elevation operation described above. Degree reduction usually changes the curve shape.

ReduceV



Reduce degree in V direction. Opposite operation of degree elevation.

Set Start



Set starting point for a closed mesh.

Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to the starting point of the mesh in both U and V directions. Release the mouse button to accept.

Set start in 'u' direction Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to the starting point of the mesh in U direction. Release the mouse button to accept.

Set start in 'v' direction Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to the starting point of the mesh in V direction. Release the mouse button to accept.

Sharpen



Insert sharp edge (crease) to the mesh.

Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position both U and V directed creases. Release the mouse button to accept.

Add U directed crease Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position a U directed crease. Release the mouse button to accept.

Add V directed crease Select a NURBS mesh object. Click and drag over the NURBS mesh object on a View window to add and position a V directed crease. Release the mouse button to accept.

Swap



Swap direction of the mesh.

Swap direction of the mesh in both U and V direction.

Swap U direction Swap direction of the mesh in the U direction.

Swap V direction Swap direction of the mesh in the U direction.

SwapUV



Swap direction of the mesh. The U direction becomes the V direction and vice versa.

Weight



Modify the weight of the selected control points.

Select the points to change the weight for. Activate the tool. Click on a view window. Move the mouse cursor up to increase weight or down to decrease weight. Click again to accept.

Weld



Weld two meshes to each other.

NURBS Curve Tools

Break



Break the curve at a specified point. A closed curve the curve is made open and the starting point is set to the specified point. An open curve is separated into two curves at the specified point.

Select a NURBS curve object. Click and drag over the NURBS curve object on a View window to set the breaking point. Release the mouse button to accept.

Close

Make closed curves open and open curves closed.

Open - Open the curve.

Open Exact - Open the curve so that its shape does not change.

Close - Close the curve.

Concatenate



Concatenate two or more Nurbs curves.

Delete



Delete selected points.

Distribute



Distributes curve points evenly (at an equal distance from each other) along the curve. The shape of the curve usually changes in this operation. If the curve is not smooth everywhere, changes can be remarkable.

If the curve is cubic, the number of points increases by 2 and parametrization changes to uniform.

Elevate



Degree elevation.

Elevate with Fixed Shape

The degree of the curve is elevated by one. Elevation is done so that the shape does not change. For example, a linear curve remains linear by its shape. However, you can later bend it smoothly because the higher degree allows it.

Elevate by Smoothing the Shape

Degree elevation uses the control points to form as smooth shape as the higher degree allows.

Extrude



Extrude selected curves with given bevelling.

Activate the Extrude tool with the selected curves.

Invert



Make the curve pass through its knot points.

InvKin



Move the selected point with Inverse Kinematics.

Select a point from a Nurbs curve. Activate the InvKin tool. Set options on the Control Bar. Click on a View window. Move mouse to move the selected point. Click again to accept.

Measure



Measures the length of the curve (in meter scale). The result is printed into the status window and to the info window.

New Point



Insert new point to the curve. The shape of the curve is not changed.

Select a NURBS curve object. Click and drag over the NURBS curve object on a View window to add and position the new point. Release the mouse button to accept.

Parallel



Creates a parallel curve from the selected curve. The created curve travels through a point, which user defines by a mouse click.

The offset distance is added in the view plane of the window where the user clicked the offset point. This means that it is usually best to apply the tool from the same viewing direction from which the curve was created.



Parallel curves created for some curve shapes

The created parallel curve has an equal number of points with the original curve. Second order parallel curves (polylines) follow the original curve exactly at the defined distance. For higher degree curves, such as cubic curves, this is generally not possible. The control polygons are parallel but the actual curves may get closer or further apart at strongly bent curve sections.

Param



Set curve parametrization.

Uniform - Set uniform parametrization for the curve.

Chordlength - Set chord length parametrization for the curve.

Centripetal - Set centripetal parametrization for the curve.

Reduce



Reduce degree in U direction. Opposite to Degree elevation operation described above. Degree reduction usually changes the curve shape.

Rotate



Construct rotational nurbs surface by rotating the second curve about the first curve.

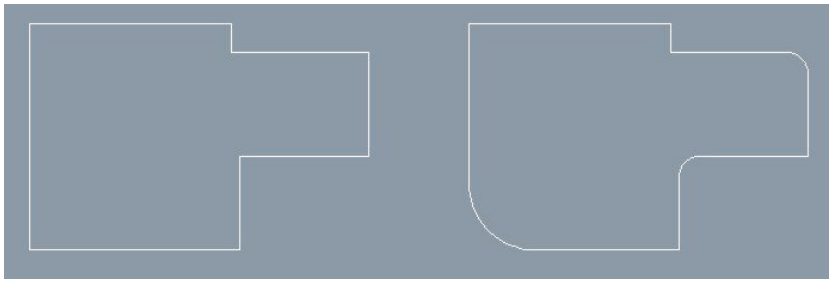
Activate the Rotate tool with the selected curves.

Round



Round the control polygon of selected curves at selected points.

Number of points used for rounding and rounding radius can be defined.



Original control polygon at left, rounding applied to some vertices at the right side

Sharpen



Add a sharp point on the selected curve.

Select a NURBS curve object. Click and drag over the NURBS curve object on a View window to add and position the new sharp point. Release the mouse button to accept.

Start



Set curve start point.

Select a closed NURBS curve object. Click and drag over the NURBS curve object on a View window to set the start point. Release the mouse button to accept.

Swap



Swap the direction of the curve.

Sweep



Construct a surface by moving a cross section curve along one or two rail curves.

Activate the Sweep tool with the selected curves.

ToBezier

 Convert to Bezier curve.

Trim

 Trim the parent mesh.

Select a Nurbs curve whose parent is a Nurbs mesh. Activate the Trim tool. The selected Nurbs curve is projected onto the Nurbs mesh and corresponding trim curve is created on the Nurbs mesh.

Weight

 Modify the weight of the selected points.

Select the points to change the weight for. Activate the tool. Click on a view window. Move the mouse cursor up to increase weight or down to decrease weight. Click again to accept.

XSect

 Construct a mesh through the selected curves.

Activate the XSect tool with the selected curves.

TrimNurbs Tools

Invert

 Set to trim out inside, reset to trim out outside.

Toggle the Invert attribute of the trim curve, whether the surface points inside or outside the curve are trimmed out.

To3D

 Map the selected trim curves to regular 3D nurbs curves.

Trim

 Trim the surface.

Toggle the Cut attribute of the trim curve, whether to trim the parent mesh or not.

Weld



Create a weld surface through the two selected trim curves.

Activates the Weld tool with the selected trim curves.

Plant Tools

Branch



Adds a new sub branch object to each selected branch or plant root object. You can use hierarchal branch structures to define growth behaviors, which change a lot between generations: trunk, main branches, small branches etc.

Leaf



Adds a new leaf object into each selected branch or plant root object.

Plant



Creates a fractal plant's seed object with a sub branch object. To use the tool:

- Activate the Plant Root tool from the tool bar or from the pull down menu.
- Click on a view window to define the point where the plant grows.
- Move the mouse upwards to define the direction and speed of the growth. Click second time to finish the tool.

Randomize



Sets a new random seed value to selected plant root objects. If plants have randomized properties, each plant becomes unique.



These small trees have the same growth properties but differend random seeds

Root



Adds roots to the selected plant. To use the tool, select a plant (or several plants) and click the button. The tool does not need any mouse clicks.

The tool adds a root sub object to each selected plant. A root object is similar to the branch object, except that it grows downwards. The geometry type of the root is initialized to match the first branch object.

Variate



Adds a random change to all growth properties of the selected plant components and their sub objects.

Applying the tool a couple of times usually changes the look of the plants quite much.



A plant and some variations created from it

Reset



Resets attributes of the selected plant component(s) back to the default state. This tool does not work hierarchically. Applying it to a plant seed level resets only the seed attributes - sub branches and leaves will not change.

Subdivision Surface Tools

Subdivision Creation Tools

The SDS tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains many SDS creation tools.



These tools can be drag&dropped to a tool bar.

Circle



Create SDS circle.

Specify options in the Control Bar. Define the circle by clicking two or three points on a View window depending on the Define option.

The SDS Circle tool has the following options:



- Define - How the circle is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Arc - If checked creates an arc

- Points - number of vertices along the circle
- Density - number of extra curves
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Cone

 Create SDS cone.

Specify options in the Control Bar. Define the bottom circle of the cone by clicking two ot three points on a View window depending on the Define option. Define the apex point by clicking on a View window.

The SDS Cone tool has the following options:



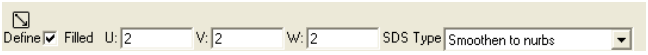
- Define - How the cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Arc - If checked creates a cone sector
- Points - number of vertices along the bottom circle
- Density - number of extra curves
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Cube

 Create SDS cube.

Specify options in the Control Bar. Define the cube by clicking two points on a View window.

The SDS Cube tool has the following options:



- Define - How the cube is defined, either by two corners or center and corner.
- Filled - If not checked the top and bottom face of the cube are not created.
- U - number of vertices in the width direction
- V - number of vertices in the height direction
- W - number of vertices in the depth direction
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Cut Cone



Create SDS cut cone.

Specify options in the Control Bar. Define the bottom circle of the cut cone by clicking two ot three points on a View window depending on the Define option. Define the top circle by clicking two points on a View window.

The SDS Cut Cone tool has the following options:



- Define - How the cut cone is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Arc - If checked creates a cut cone sector
- Points - number of vertices along the bottom circle
- Density - number of extra curves
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Cut Pyramid



Create SDS cut pyramid.

Define the bottom rectangle of the cut pyramid by clicking two points on a View window. Define the top rectangle point by clicking two more points on a View window.

The SDS Cut Pyramid tool has the following options:



- Define - How the cut pyramid is defined, either by two corners or center and corner.
- Filled - If not checked the top and bottom face of the cut pyramid are not created.
- Center - If checked, the apex will be centered over the bottom.
- U - number of vertices in the width direction
- V - number of vertices in the height direction
- W - number of vertices in the depth direction
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Cylinder



Create SDS cylinder.

Specify options in the Control Bar. Define the cylinder by clicking two or three points on a View window depending on the Define option.

The SDS Cylinder tool has the following options:



- Define - How the cylinder is defined. The choices are:
  - Center & Radius
  - Center & Axes
  - Corner & Corner
  - Three points in arc
  - Center & Corner
- Arc - If checked creates a cylinder sector
- Points - number of vertices along the bottom circle

- Density - number of extra curves
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Pyramid



Create SDS pyramid.

Specify options in the Control Bar. Define the bottom rectangle of the pyramid by clicking two points on a View window. Define the apex point by clicking on a View window.

The SDS Pyramid tool has the following options:



- Define - How the pyramid is defined, either by two corners or center and corner.
- Filled - If not checked the bottom face of the pyramid is not created.
- Center - If checked, the apex will be centered over the bottom.
- U - number of vertices in the width direction
- V - number of vertices in the height direction
- W - number of vertices in the depth direction
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Rectangle



Create SDS rectangle.

Specify options in the Control Bar. Define the rectangle by clicking two points on a View window.

The SDS Rectangle tool has the following options:



- Define - How the rectangle is defined, either by two corners or center and corner.
- U - number of vertices in the width direction
- V - number of vertices in the height direction

- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Sphere



Create SDS sphere.

Specify options in the Control Bar. Define the sphere by clicking two points on a View window.

The SDS Sphere tool has the following options:

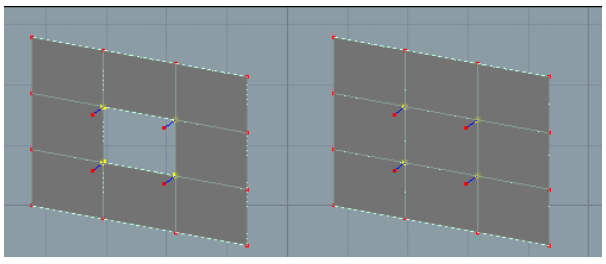


- Input - How the sphere is defined, by center and radius, two corners, or center and corner.
- W - number of vertices in the width direction (along the equator)
- H - number of vertices in the height direction (from pole to pole)
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

Subdivision Editing Tools

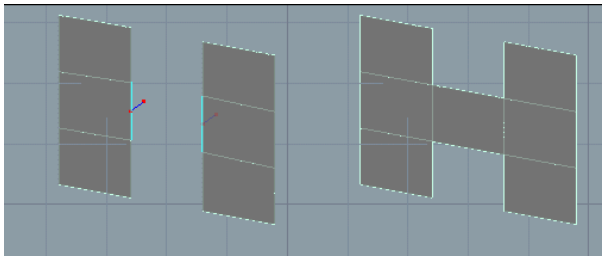
Add





Add Face

Creates a new face that connects the selected vertices. The order in which the vertices are selected determine their order in the face.

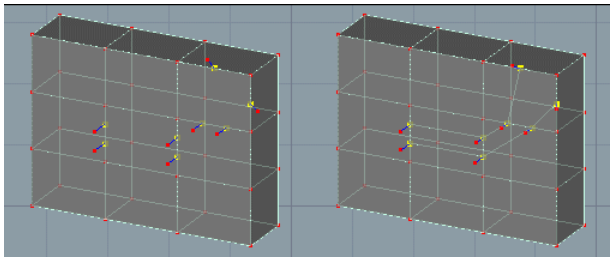


Add Face by Connecting Edges

Creates a face between two selected edges.

Add Edge

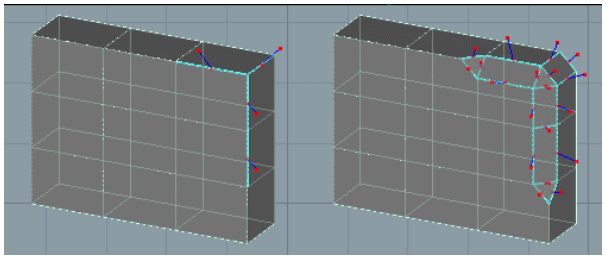
Creates new edges between the selected vertices of a face thus dividing the face. The number of selected vertices in a face must be even and the longest contiguous selection must contain exactly half of the selected vertices in that face.



Bevel



Bevel edges or vertices

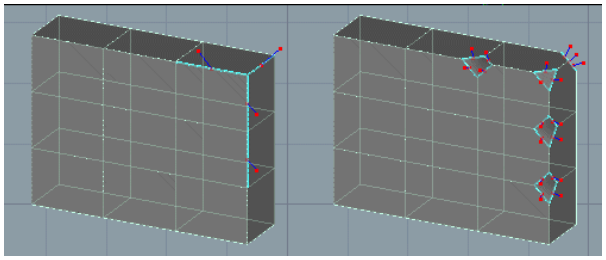


Bevel Edge

The first click on a view window starts the beveling. The distance of the cursor from the first click position defines the amount of beveling. The second click finishes the tool.

Bevel Vertex

The first click on a view window starts the beveling. The distance of the cursor from the first click position defines the amount of beveling. The second click finishes the tool.



Crease

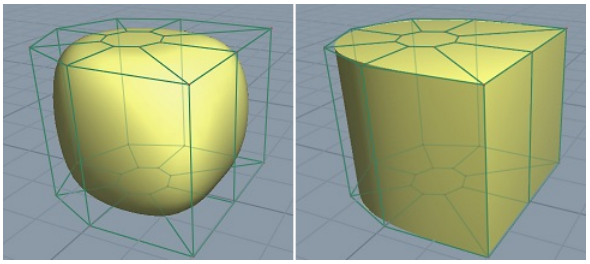
This tool examines SDS geometry and sets sharp creases to all points/edges, where the angle between connected faces exceeds the given limit.

In point/edge/face mode, the tool changes only selected areas. In object mode, the whole object is considered.



Note

The tool only adds creases. It does not remove user defined creases. Semi sharp or free creases are not changed to sharp.



Set edge and point creases automatically by a threshold angle

The tool has the following controls:

- Angle - the threshold angle. In edge crease creation, surrounding faces must exceed the limit angle at both ends of the connecting edge. A sharp point crease is set when all faces connect to the point's average normal with an angle that is greater than the limit. For example, a cone like surface usually gets a point crease in the apex, although surrounding cone side faces join to each other smoothly. That happens because each face has a sharp angle with the apex direction.
- Target - selects if point, edge or both crease types should be considered. Often the threshold angle shoudl be different for edge and point creases.

Create Subdivision Object



The interactive SDS (SubDivision Surface) creation tool can be used to create SDS vertices and faces or to add them to an existing SDS object. The SDS creation tool is in the 'SDS' section of the default toolbar.

The creation tool has the following options:



- Edit Selected - If checked adds new vertices and faces to the selected SDS object, otherwise creates a new SDS object
- Create Face - If checked connects the new vertices into faces, otherwise creates separate vertices
- Auto Accept - If checked automatically accepts a face when Points per face vertices have been entered.
- Points per face - How many vertices go into a face when the Auto Accept option is selected
- Type - SDS Object type, e.g. polygonal

New vertices can be added by clicking on a view window. Holding down the 'Alt' modifier key while clicking adds the nearest existing vertex to the new face.

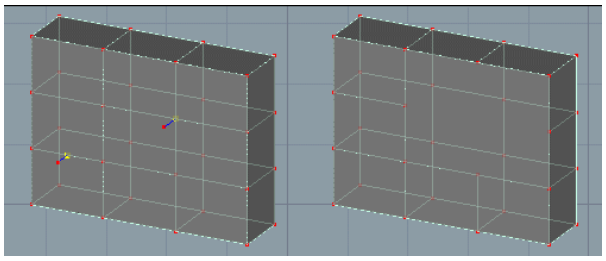
The Accept Face button can be used to accept a face when the Auto Accept option is not used or when accepting a face with fewer vertices than in the Points per face control.

Delete

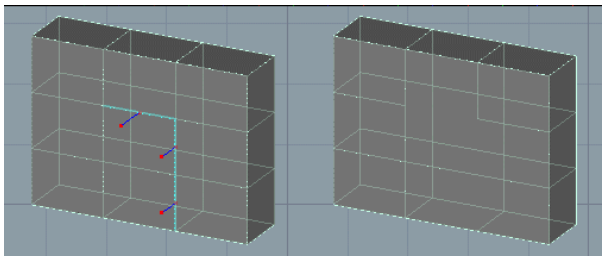


Selected:

In the vertex selection mode removes selected vertices. If a selected vertex is a shared corner of multiple faces then these faces are merged.

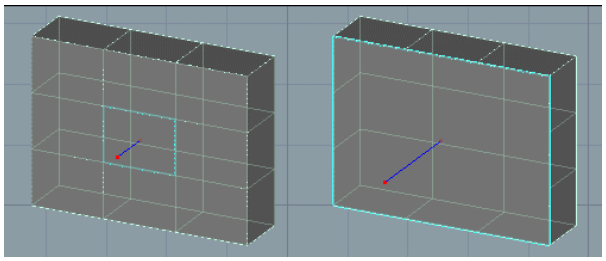


In the edge selection mode merges the faces that share a selected edge.



In the face selection mode the selected faces are merged with their neighbors.

Orphan Vertices: Deletes all vertices that are not part of any face.

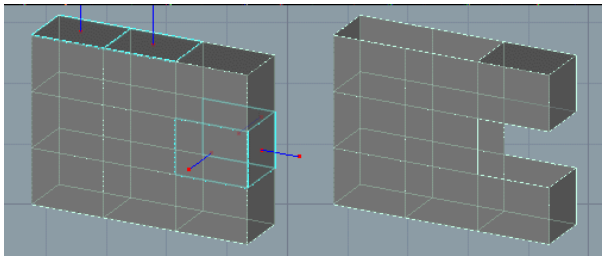


Destroy Faces



Destroy Faces - Destroys selected faces and vertices that were connected only to the destroyed faces.

Destroy Faces, keep orphan vertices - Destroys selected faces but does not delete any vertices.



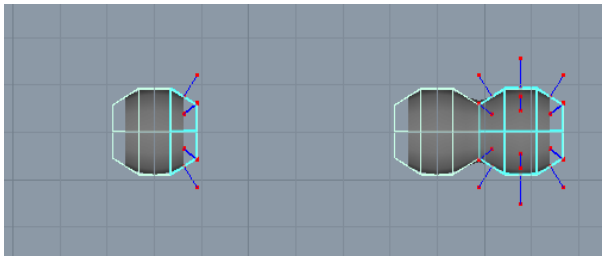
Duplicate and Mirror



Duplicates those parts of the SDS geometry that do not form a closed surface. The duplicated geometry is mirrored about the open edges loops (holes or outlines).

If there are more than one open edge loop, then selecting an edge or face at the open edge loop selects the edge loop about which to mirror.

The mirrored geometry is connected to the edge loop. The vertices at the edge loops are not duplicated.



Duplicate and Mirror with a selected edge loop

Duplicate Faces



Copies all the faces. If the object has open edges then the open edges of the original faces and those of the copied faces are connected with new faces. The copied faces become selected so that they can be readily transformed, e.g. moved apart.

Duplicate Selected Edges

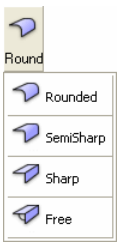


Duplicates selected edges and creates new faces to connects the duplicates with the original edges. The duplicates are selected so that they can be readily modified, e.g. moved to a new position.

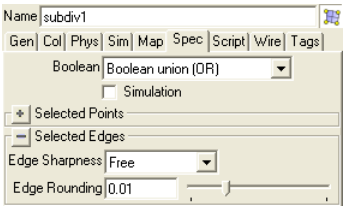
Edge Rounding

Sets rounding of the selected edges. Possible values are:

- Rounded
- SemiSharp
- Sharp
- Free



Set rounding of the selected edges



When edge rounding is set to Free, the rounding radius can be adjusted through the Spec tab of the property window

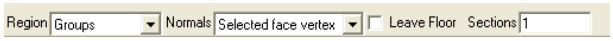
Extrude



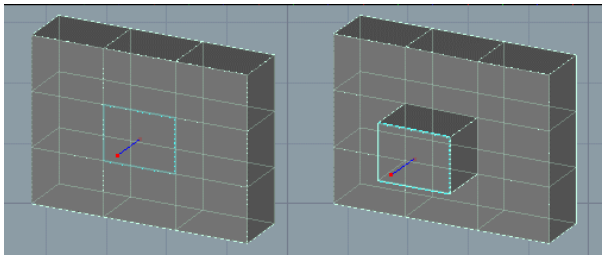
Extrude selected faces or vertices

Extrude Face

This tool has the following options:

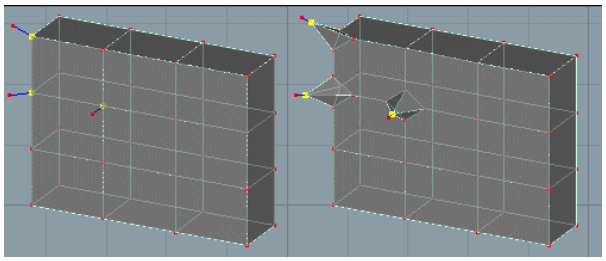


- Region - Groups: adjacent faces are extruded as a single region, Separately: each face is extruded separately
- Normals - Group: all vertices in a face group are moved in the direction defined by all the faces in the group, Vertex: each vertex is moved in the direction of its normal, Selected face vertex: the direction of each vertex is the average of the normals of the selected faces that the vertex belongs to.
- Leave Floor - If checked faces are added where the selected faces were originally.
- Sections - The extrusion is divided into this many sections.



The first click on a view window starts the extrusion. Moving the mouse in the vertical direction adjusts the amount of extrusion. The second click finishes the tool.

Extrude Vertex



The first click on a view window starts the extrusion. Moving the mouse in the vertical direction adjusts the amount of extrusion and horizontal movement adjusts the size of the base of the extrusion. The second click finishes the tool.

Faces



Activates the interactive Subdivision Creation Tool described in the beginning of this chapter.

Fill Holes

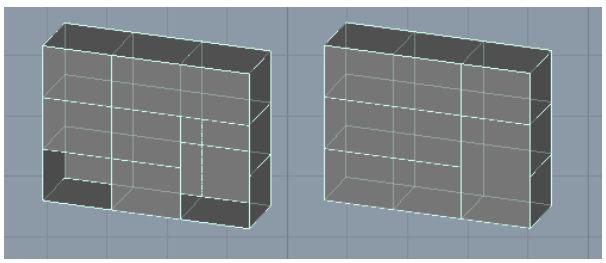


Adds new faces where there are loops of edges that are connected to one face only. These edges define the holes that can be filled.

This tool has the following options:



- Max Edges - the maximum number of edges around a hole to be filled. Only holes that have don't have more than this many bordering edges are filled.
- Selected - If checked, only holes in selected regions are filled.

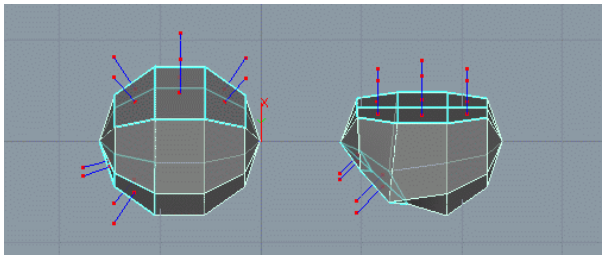


Flatten



In face mode flattens selected face regions separately.

In vertex or edge mode flattens all selected vertices or edges as one group regardless of topology.



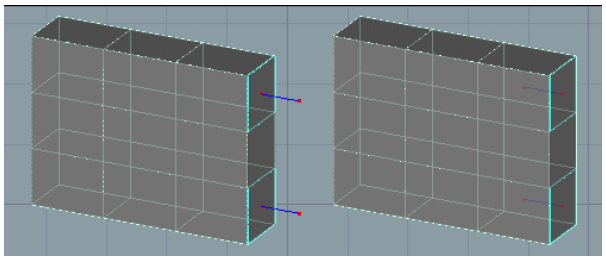
Flatten with two selected face regions

Flip Faces



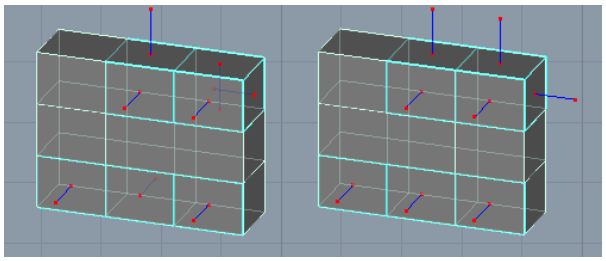
Separately

Flips the face normals of the selected faces to the opposite direction.



Regions

Flips the opposing face normals in each selection region so that all faces normals in that region will point to the same side. The majority of faces in the region defines to which side the normals will point. If none of the faces is selected then the whole object is considered as selection regions defined by one or more groups of adjacent faces.



Freehand Knife



Lets the user draw a freeform curve or a line segment that is used to cut edges of the underlying subdivision object. The edges are cut by adding new vertices where the knife hits the edges.

The knife tool has the following options:



- Selected - If checked only selected edges can be cut, otherwise all edges can be cut.
- Add Edges - If checked connects the new vertices with new edges to divide the faces into smaller faces, otherwise creates only the vertices where the knife cuts the edges.
- Select - Select which faces will be selected after cutting.
  - No selection - No faces will be selected
  - Inside - The faces inside the freehand curve will be selected
  - Outside - The faces outside the freehand curve will be selected

  The selection is based on the center point of the face (whether it is inside or outside the curve).

- Freehand - If checked a freehand curve can be drawn, otherwise a single line segment can be drawn.

Hide/Show



Hide selected faces or show all faces

Hide Selected Faces - Hides selected faces in wireframe display.

Show All Faces - Displays all faces.

Init UV

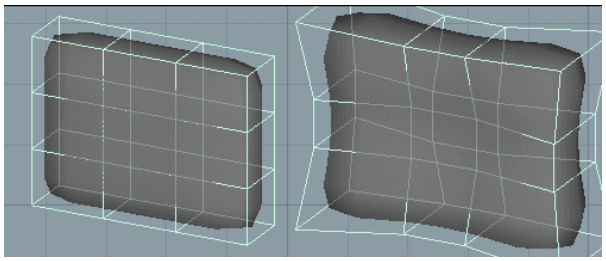


Calculates default values for the currently selected UV coordinate channel. The direction of each vertex from the center (average) of all vertices defines the UV value for that vertex.

Invert



Makes the surface go through the original control polygon. This is true only for non-polygonal SDS objects.



SDS cube inverted

Join Faces

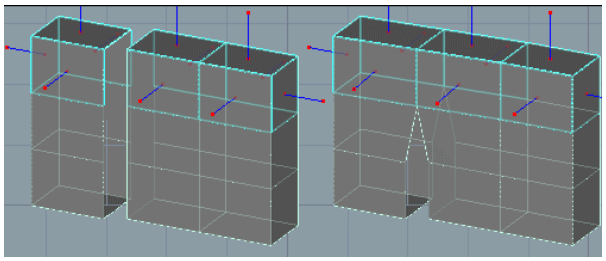


Merges near-enough pairs of open edges thus joining the faces. If there are selected faces or edges then only these edges will be considered. If no face or edge is selected then all edges will be considered.

This tool has the following option:



- Distance - the maximum distance between edges to be connected. Edge pairs that are further apart will not be considered.

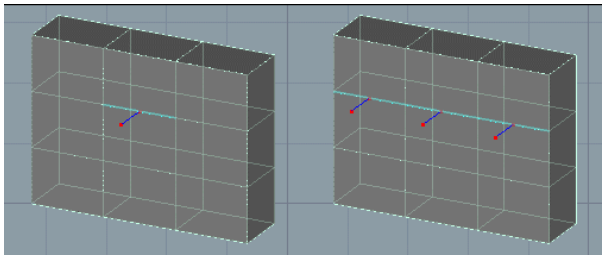


Loop

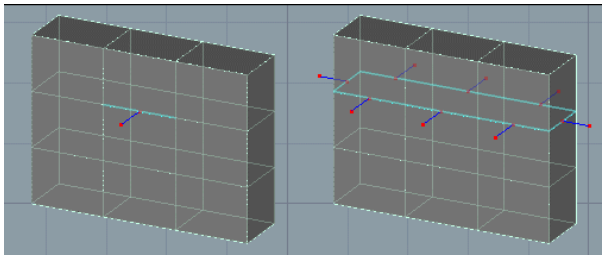


Expands the current edge selection either to contiguous edges or parallel edges. This tool uses the following modifiers keys:

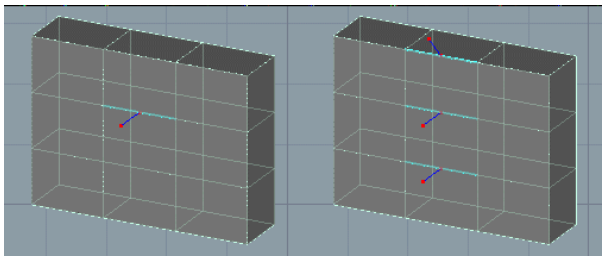
- Shift - Expands the selection as far as there are edges that meet the selection criteria.
- Ctrl - Selects edges that are parallel to the currently selected edges.



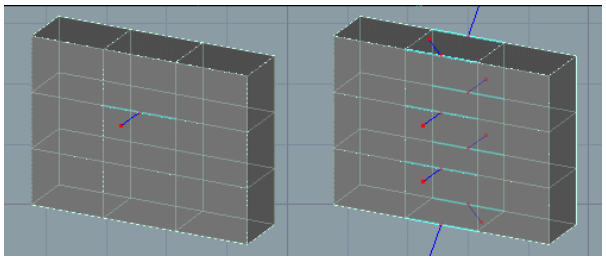
Edge selector tool used without modifier keys



Edge loop selector tool used with the <Shift> modifier key



Edge loop selector tool used with the <Ctrl> modifier key

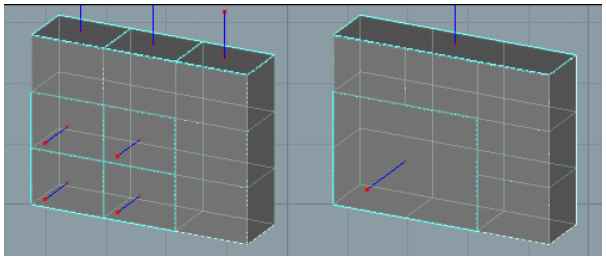


Edge loop selector tool used with the <Ctrl> and <Shift> modifier keys

Merge



Merge Faces, Edges or Vertices

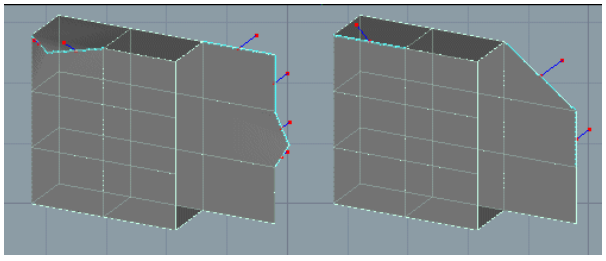


Merge Faces

Merges adjacent selected faces into a single face.

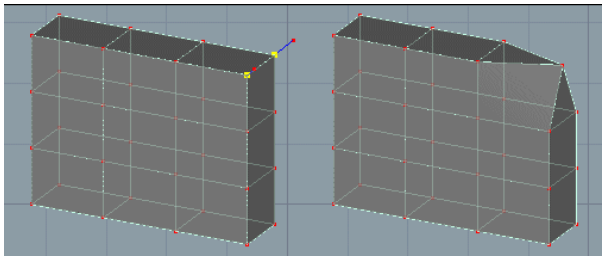
Merge Edges

Merges two or more contiguous edges into one. All these edges must be either connected only to a single face or go along a border between two faces.



Merge Vertices

Merges two or more selected vertices into one.



Merge Objects



Merge Objects - Merges all selected SDS objects into a single SDS object.

Merge Objects, Selected Faces - Merges the first selected SDS object and the selected faces from the other selected SDS objects into a single SDS object.

Object Knife



Lets the user cut SDS objects with the surface of a tool object. The edges are cut by adding new vertices where the tool object's surface hits the edges.

Select one or more SDS objects and one tool object (selected last). Set the tool options on the Control Bar and accept the tool.

The object knife tool has the following options:



- Selected - If checked only selected edges can be cut, otherwise all edges can be cut.
- Add Edges - If checked connects the new vertices with new edges to divide the faces into smaller faces, otherwise creates only the vertices where the knife object cuts the edges.
- Select - Select which faces will be selected after cutting.
  - No selection - No faces will be selected
  - Inside - The faces inside the tool object will be selected
  - Outside - The faces outside the tool object will be selected

  The selection is based on the center point of the face (whether it is inside or outside the tool object).

Separate



In face mode disconnects the selected faces from the rest of the SDS object. The separated faces remain selected so that they can be e.g. moved to a new place.

In edge mode separates the model along selected edges.

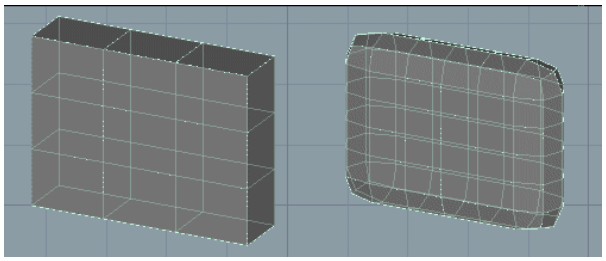
In vertex mode each connected face gets its own copy of the selected vertices.

Smooth



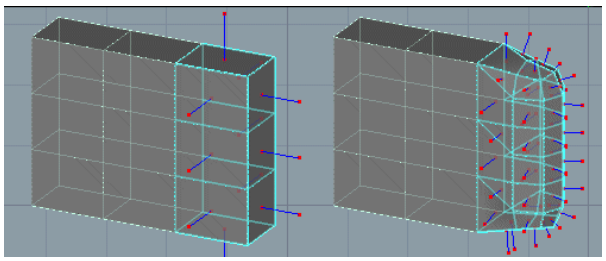
Smooth Whole Object

The subdivision algorithm is used to create a smoother SDS object with more vertices and faces.



Smooth Locally

The subdivision algorithm is used to create a smoother SDS object from the selected faces. This smoothed object is then joined with the unselected faces.



Smoothen/Flatten

Smoothen of flatten selected shading normals 

This tool can be applied to polygonal SDS objects, which have the Phong shading option set. To use it, set the target object to the edit mode, select some faces/edges/points and apply the tool. Clicking the tool icon toggles between smooth and sharp shading normals. From the side menu, the desired alternative can be chosen directly as follows:

Smoothen selected shading normals

Sets the phong shading normal of every face passing trough a selected point to the common average normal. Phong shading interpolates smoothly over the selected points.

Flatten selected vertex normals

Sets the phong shading normal of every face passing trough a selected point to the geometric normal of the polygon corner in question (computed from the two surrounding polygon edges). Phong shading does not interpolate the normal smoothly but uses a private normal per face per selected point.

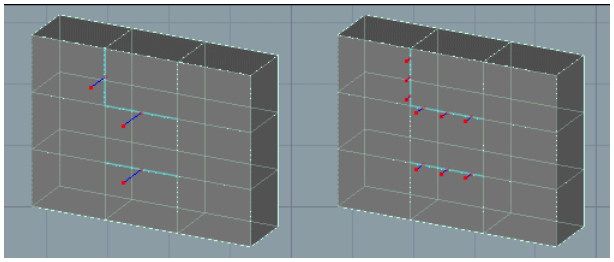
Split Edge

 Adds one or more new vertices along each selected edge thus dividing the edges into two or more pieces. The new vertices are selected.

This tool has the following option:



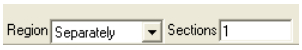
- Splits - how many new vertices to add to each selected edge.



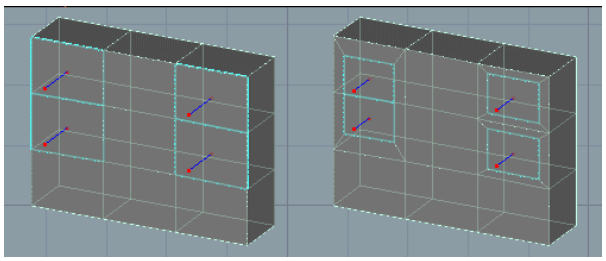
Split Edge tool with 2 splits used

Subdivide

 This tool has the following options:



- Region:
  - Groups: adjacent faces are subdivided as a single region
  - Separately: each face is subdivided separately
- Sections - The subdivision is divided into this many sections.



Subdivided faces with Region set to Groups and Separately

The first click on a view window starts the subdivision. Moving the mouse in the vertical direction adjusts the amount of subdivision. The second click finishes the tool.

To SDS



Converts selected objects to subdivision surface objects or triset objects.

The ToSDS tool has the following options:

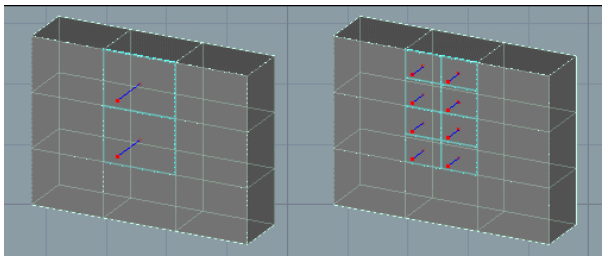


- Use Default Resolutions - Check this to use default number of faces.
- U - Number of faces in the U direction. Can be set if Use Default Resolution is unchecked.
- V - Number of faces in the V direction. Can be set if Use Default Resolution is unchecked.
- Subdivision Object - Check this to make Subdivision object instead of triset.
- Merge - Check this to merge all converted objects into a single Subdivision object.
- Join - Check this to join faces that have edges near enough.
- Distance - Maximum edge distance for joining faces.
- Del orig. - Check this to delete the originally selected objects after conversion.

Tri/Quad



Divide faces to quads or triangles

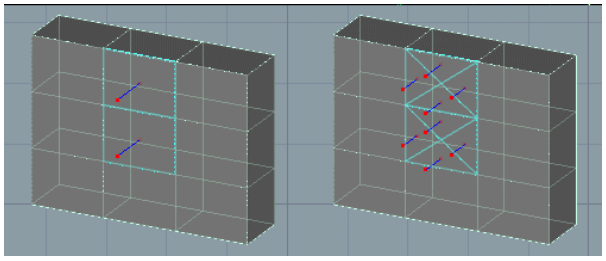


Divide Faces to Quads

Divides the selected faces to quads.

Divide Faces to Triangles

Divides the selected faces to triangles. Adds a new point to the middle of the face.

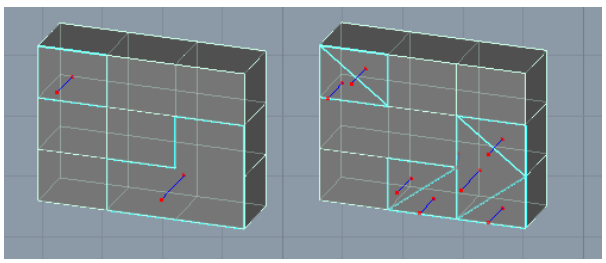


Triangulate Selected faces

Divides the selected faces to triangles without adding any new points.



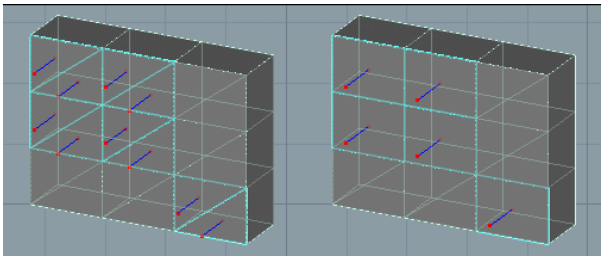
- Triangulate Quads - If checked, also quad faces will be divided into triangles.
- Convert to Quads - If checked, the triangles will be converted to quads where possible.



Triangles to Quads



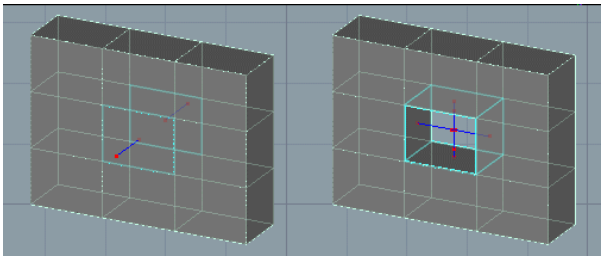
Combines selected triangular faces in pairs into quads.



Tunnel



Creates a tunnel between two selected areas. Each area can contain one or more adjacent faces.



Unwrap



Unwraps the geometry of the SDS object. Bumps and peaks become smoother, and points more evenly distributed. Usually unwrapping also shrinks the object, which side effect can be compensated by resizing the object suitably after the operation.

In edit mode, only the selected areas change.

Unwrap UV



Unwraps the current UV channel. UV vertices distribute more evenly.

UV Set



Selected faces

Creates a UV Set object containing the selected faces.

Cubic UV Sets

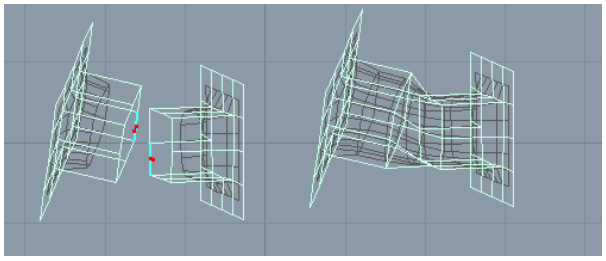
Creates a UV Set object for each main direction (Top, Bottom, Left, Right, Front, Back) containing the selected faces whose normals point into that direction. The directions are determined by the object's coordinate system.

Weld



Welds two open edge loops together by adding faces in the gap.

If there are more than two open edge loops, then selecting an edge or face at the open edge loop selects the edge loop to weld.



Weld with a two selected edge loops

Reducer



Polygon reduction tool.

Select one or more SDS objects and activate the tool. Then specify desired reduction ratio using the Ratio control shown in the Control Bar, and accept the tool.



Note

The reduction ratio is given in percentags. Using the value of 100% corresponds to no reduction.

SDS UV Set tools

Many of the SDS specific tools are also supported by the UV Set class. In addition to this, the UV Set specific control bar shows the following tools.

Connect



Connects selected areas by parent topology. Connects UV Set faces whose parent faces are already connected.

UV From View



Defines UV of the selected faces or vertices by their relative position on the current View window.

To Face Material



Copies UV Sets to the parent SDS object as face materials. UV Set object's child mapping objects define the face material names. The UV values for the face materials are taken from the UV Sets.

Simulation Tools

The Simulation tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the simulation tools.



These tools can be drag&dropped to a tool bar.

COG



Define center of gravity for the selected objects.

On a View window enter a point for the COG.

Collision



Enable / disable collision detection for the selected objects, and control the typical collision detection specific properties.

When the tool is enabled, it shows the current collision detection mode along with the options that typically need to be defined with the effect.

- Mode - Simulation mode, for enabling the collision detection. Setting this field to Full means that the object both detects collisions and is also sensitive to the collisions. Cause means that this object can affect other objects but its own kinetic energy will not change due to collisions. Affected means that the object will detect collisions, but won't affect any other objects due to collisions. See the Property Window / Sim tab for more information.
- Accuracy - Specifies the number of samples used to detect collisions. The higher the value, the more accurately collisions are detected.
- Surface Friction.
- Rebound - Value of 1 corresponds to dissipation free collision where the object maintains its kinetic energy in collisions.
- Elasticity - Objects ability to maintain its initial shape. If elasticity is zero, there is no force that restores the shape after a deformation.
- Rigidity - The lower the value, the more freely object matter can react to forces and the less internal friction resisting the shape deformations exists. If the rigidity is zero, object geometry points act independently in simulations (elasticity defines a string force that maintains the shape).
- Inertia - Describes how the mass is distributed in object volume. If the mass is concentrated in the middle of the object, the inertia is small and the object has a tendency to spin easily in collisions.

To create a simulation where two spheres collide with each other:

1. Select a sphere and enable the 'Velocity' simulation tool. Enter two points in the View window to define velocity vector for the sphere.

2. Create another sphere and multi select the two spheres. Note that the spheres must be in the same hierarchy level.

3. Enable the Collision tool, set Mode option to 'Both' and accept the tool.



Note

You can activate the Collision tool again to edit any collision detection specific properties later. The options can also be set in the Property Window / Phys and Sim tabs.

Distant Gravity



Create a distant gravity field.

On a View window enter two points that define the direction and force of the distant gravity.

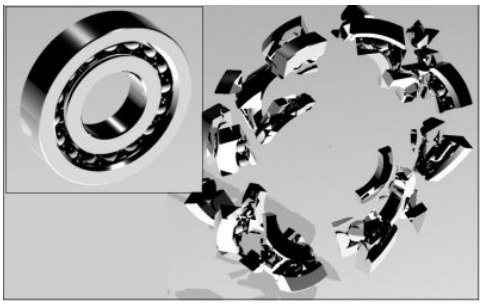
Explode



Explode the object into pieces.

Select objects to be exploded and apply the tool. Then set the desired options for the created explode control object.

Explode tool has the following options:



Ballbearing exploded with Boolean option

- Type - Specifies the type of the explosion parts generated:
  - Boolean - Use Boolean Operations to split the objects into parts.
  - Cube - Divide the surface into analytic cubes.
  - Sphere - Divide the surface into analytic spheres.
  - Mesh - Divide the surface into NURBS patches.
- Resol - Specifies the subdivision resolution.
- Generations - Controls over how many generations explosion parts explode again.
- Velocity - Default Velocity for the explosion parts generated.
- Spin - Default Spin for the explosion parts generated.
- Randomness - level of randomness that can take place during explosion.
- Life Time - Life time for the explosion parts. The value of 0 corresponds to infinite life time.
- Explode Time - Time after which the target object explodes.
- Maximum - Maximum number of explosion parts allowed.

Fan



Create a fan object for fluid dynamics.

On a View window enter three points that define the fan geometry.

Fluid



Enable / disable fluid processing for the selected objects. Set the typical fluid sensitive object properties.

When the tool is enabled, it shows the fluid specific object properties of the currently selected objects. See the Property Window /Phys tab, for more information about the Fluid Dynamics specific properties.

Gravity



Enable / disable gravity processing for the selected objects, and control the typical gravity sensitive properties.

When the tool is enabled, it shows the gravity specific object properties of the currently selected objects.

- Mode - Gravity mode. The mode 'Full' means that this object is both sensitive to the gravity as well as acts as a gravity source itself. 'Cause' means the object acts only as a gravity source. 'Affected' means that the object feels gravity caused by other gravity sources, but does not generate any gravity field itself. See the Property Window / Sim tab for more information.
- Mass - Mass controlling the strengh of the generated gravity force, as defined by Newton's laws of physics.

To create a simulation of two planet system, orbiting each other:

1. Create two analytic spheres representing the planets. The scale could be such that the radiuses of the spheres is 0.1 m and the distance between the spheres 1m.

2. Select the both spheres and enable the Gravity tool. Set Mode to 'Both', set 'Mass' to some very high value (say 10 000 000 kg). If the distance between the spheres is say 1 m, then this mass will generate noticeable gravity force.

3. Optionally, use the Velocity tool to define initial velocity for the spheres so that they will orbit around rather than just oscillate back and forth due to gravity.

Magnetism



Enable / disable magnetism processing for the selected objects. Set the typical magnetism sensitive object properties.

When the tool is enabled, it shows the magnetism specific object properties of the currently selected objects.

- Mode - Simulation mode which determines whether the object is sensitive to magnetism (Affected), acts as a source of magnetism (Cause), or both. See the Property Window / Sim tab for more information.
- Charge - The level of electric charge (coulomb). Can be positive or negative. Note that magnetism processing is hierarchical, allowing dipolic objects.

Spin



Set spin of the selected objects.

Select objects to define the spin for. Enter three points that define the spin.

Velocity



Set velocity of the selected objects.

Select objects to define the velocity for. On a View window enter two points that define the velocity vector.

The Velocity tool has the following option:



- Selected Points - Set velocity to selected points only.

Transformation Tools

Available GUI Objects

The Transformation tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the transformation tools.



These tools can be drag&dropped to a tool bar.

Bend



Bend the selected objects. If the object is in point editing mode then the tool will have effect on the selected points only.

Select the objects to be bent. Define the start position for the bend by clicking in a View window. Click to define the end position for bending. The object will be bent between the start and the end points. Move the mouse to get the desired bending and accept the tool by clicking in a View Window.

Extend



Scale the selected objects in the given direction.

Select the objects to be scaled. Enter the center of scaling by clicking in a View Window. Define another point by clicking in a View Window. Move the mouse until the desired effect is achieved and click in a View Window to accept the tool.

The Extend tool has the following options:



- About COG - Extend object about the center of gravity.

IK



Apply inverse kinematics to the selected points of the selected objects.

Select the object to be modified using Inversed Kinematics. Currently only NURBS curves and Skeletons are supported. Enter the Edit state and select the point/joint to be modified. Click the left mouse button in a View window to start I.K. Move the joint to desired position and click the left mouse button to accept the tool.

The IK tool has the following options:



- Type - I.K. type
- Full I.K. - apply inverse kinematics in both directions
- I.K. head, move tail - apply inverse kinematics only towards the head
- Move head, I.K. tail - apply inverse kinematics only towards the tail
- Local - Cause I.K. to affect the current bone only.

Move



The Move tool can be used to translate the selected objects, handles, lattice coordinates, or texture UV coordinates.

Select the objects to be moved. Specify options in the Control Bar. Click in a View Window to define the start position. Move the mouse until the desired translation is achieved and click again to accept the tool.

The Move tool has the following options:



- Target - Transformation Target.
- Collision Deformation - Activate collision detection system.

Move-to



The Move-to tool can be used to translate the selected objects to the specified position.

Select the objects to be moved. Click in a View Window to move the selected objects to the specified position.

Mirror



Mirror the selected objects about given axis.

Select the objects to be mirrored. Click the left mouse button in a View Window to specify the start position of the mirroring axis. Move the mouse until the desired mirroring effect is achieved and click the left mouse button to accept the tool.

The Mirror tool has the following options:



- About COG - Mirror object about the center of gravity.

Noise



Apply fractal noise deformation to the selected freeform objects. If the object is in point editing mode then the tool will have effect on the selected points only.

Select the objects to be deformed. Define the start position for the noise deformation by clicking in a View window. Click to define the end position for the deformation. The object will be deformed between the start and the end points. Move the mouse to control the level of deformation and accept the tool by clicking in a View Window.

Rotate



The Rotate tool can be used to rotate the selected objects, handles, lattice coordinates, or texture UV coordinates.

Select the objects to be rotated. Specify options in the Control Bar. click in a View Window to define the center of rotation. Click in a View Window to define the point you want to rotate about the rotation center. Move the mouse until desired rotation is achieved and click to accept the tool.

The Rotate tool has the following options:



- Target - Transformation Target.
- Collision Deformation - Activate collision detection system.
- About Pivot - Rotate object about the pivot point.

Rotate and Extend



The RotExt tool combines the Rotate and Extend tools and allows the selected objects to be rotatend and extended in one go.

Select the objects to be modified. Specify options in the Control Bar. Click in a View Window to define the center of transformation. Click in a View Window to define the point you want to transform about the rotation center. Move the mouse until desired rotation and scaling effect is achieved and click to accept the tool.

The tool has the following options:



- Target - Transformation Target.
- Collision Deformation - Activate collision detection system.
- About Pivot - Rotate object about the pivot point.

Scale



The Scale tool can be used to scale the selected objects, handles, lattice coordinates, or texture UV coordinates.

Select the objects to be scaled. Specify options in the Control Bar. Enter the center of scaling by clicking in a View Window. Define another point by clicking again. Move the mouse until the desired size is achieved and click to accept the tool.

The Scale tool has the following options:



- Target - Transformation Target.
- Collision Deformation - Activate collision detection system.
- About Pivot - Scale object about the pivot point.

Size



The Size tool can be used to uniformly resize the selected objects, handles, lattice coordinates, or texture UV coordinates.

Select the objects to be resized. Specify options in the Control Bar. Enter the center of scaling by clicking in a View Window. Define another point by clicking again. Move the mouse until the desired size is achieved and click to accept the tool.

The Size tool has the following options:

- Target - Transformation Target
- Deform - Deformation type
- Uniform - No deformation
- Deform in function of distance to a point
- Deform in function of distance to an axis
- Deform in function of projection to an axis
- Ctrl Curve - Control curve that defines the shape of the deformation.
  - Triangle
  - Gaussian
  - Circle
  - Random
  - Linear
- About Pivot - Size object about the pivot point.

Skew



The Skew tool can be used to skew the selected objects, handles, lattice coordinates, or texture UV coordinates.

Select the objects to be skewed. Specify options in the Control Bar. Select the base point of the skew by clicking in a View Window. Define another point from the View window to define start point for skewing. Move the mouse to skew the object. Click in a View Window to accept the tool.

The Skew tool has the following options:



- Target - Transformation Target.
- Deform - Deformation type.
- Uniform - No deformation
- Deform in function of distance to a point
- Deform in function of distance to an axis
- Deform in function of projection to an axis
- Ctrl Curve - Control curve that defines the shape of the deformation:
  - Triangle
  - Gaussian
  - Circle

- Random
- Linear

- Collision Deformation - Activate collision detection system.
- About Pivot - Skew object about the pivot point.

Twist



Twist the selected objects. If the object is in point editing mode then the tool will have effect on the selected points only.

Define the center of twisting by clicking in a View window. The twist axis travels trough that point, orthogonally to the input plane. Click second time to define the length of the twist axis. The shorter the axis, the faster objects will twist. Move the mouse around the twist axis and click third time to finish the modification.

Triset Tools

Triset object specific control bar

When a Triset object is selected the control bar shows the following Triset specific tools:

Add Face



Add face to three selected points.

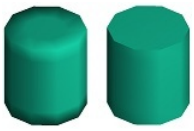
Select exactly three points of a triset object and click the tool button.

Auto Smooth



Adjust surface normals automatically.

This tool examines the true geometric angles between the faces. If an angle between two adjacent faces is lower than a given threhold value, the respective normals are set so that shading appears smooth over the edge between the faces. Otherwise normals are set to match the true geometric normal, and the edge will appear sharp.



A cylindrical triset before and after Auto Smooth

In object mode, all normals are processed. The tool also flips all normals to a consistent direction. In point/edge/face editing mode, the tool changes only selected areas of the surface, and keeps the normal direction at the same side of the face as it originally was.

Delete



Delete selected points, edges, or faces.

Flat



Flatten normals of the selected faces.

Flip



Inverts the direction of surface normals. In object mode, all normals are flipped. In edit mode, only the selected areas are flipped.

Merge



Merge selected vertices into one vertex.

Select



Selection mode. Select either by vertices, edges, or faces.

Smooth



Smooth normals of the selected points or faces.

Unified Creation Tools

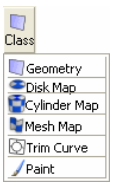


The Unified tool group in the Tools tab of the Available GUI Objects window (opened by selecting the Environments/Customization/Available Object Window... menu) contains the unified creation tools.

These tools can be drag&dropped to a tool bar.

Circle





With the Circle tool you can create objects with a circular cross section. I.e. circles, disks, cylinders, cones, cut cones, hyperboloids and ellipsoid segments. Also NURBS meshes and Subdivision Surface objects of those shapes can be created. Disk Map, Cylinder Map, Mesh Map objects and circular trim curves can also be created. The tool can also be used for 3D painting circular curves to underlying objects that have bitmap textures.

The Circle tool has the following options:

Define - How the circle is defined. The choices are:

- Center & Radius

- Center & Axes
- Corner & Corner
- Three points in arc
- Center & Corner

Class - Class of the circular object to be created.

- **Geometry** - Creates a geometry object.

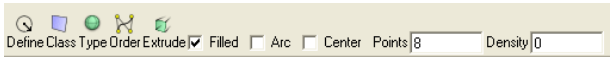


Type - type of the geometry object

- *Analytic*



  - Extrude
    - Planar - Creates circles and disks.
    - Extruded - Creates cylinders, cones and cut cones.
    - Hyperboloid - Creates hyperboloids and ellipsoid segments.
  - Filled - If checked, closes the caps
  - Arc - If checked creates open arcs
- *Nurbs*



  - Order - order of the NURBS curve or surface
    - Polygonal
    - Quadric
    - Cubic
  - Extrude
    - Planar - Creates planar circular NURBS curves or NURBS disks.
    - Extruded - Creates NURBS meshes shaped as cylinders, cones and cut cones.
    - Hyperboloid - Creates NURBS meshes shaped as hyperboloids and ellipsoid segments.
  - Filled - If checked, closes the caps.
  - Arc - If checked creates open arcs.
  - Exact - If checked creates an exact NURBS circle.
  - Points - Number of points in curve or surface.
  - Density - Number of extra curves
- *Subdivision*



- Extrude
  - Planar - Creates planar circular SDS disks.
  - Extruded - Creates SDS objects shaped as cylinders, cones and cut cones.
  - Hyperboloid - Creates SDS objects shaped as hyperboloids and ellipsoid segments.
- Arc - If checked creates open arcs
- Points - Number of points in curve or surface
- Density - Number of extra curves
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

- **Disk Map** - Creates a disk mapping.



  - Arc - If checked creates open arcs.
  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped
- **Cylinder Map** - Creates a cylinder mapping.

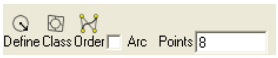


  - Arc - If checked creates open arcs
  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped
- **Mesh Map** - Creates a mesh mapping.



  - Order - order of the NURBS curve or surface
    - Polygonal
    - Quadric

    - Cubic
  - Filled - If checked, closes the caps
  - Arc - If checked creates open arcs
  - Points - Number of points in curve or surface
  - Density - Number of extra curves
  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped
- **Trim Curve** - Creates a circular trim curve.



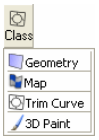
  - Order - order of the NURBS trim curve
    - Polygonal
    - Quadric
    - Cubic
  - Filled - If checked, closes the trim curve
  - Arc - If checked creates an arc
  - Points - Number of points in the trim curve
- **Paint** - Paints circular curves to underlying objects that have bitmap textures.



  - Order - order of the NURBS curve
    - Polygonal
    - Quadric
    - Cubic
  - U: - Number of points in the curve.
  - Line Width - width of the circular curve to be painted.
  - Filled - Select either a circle or a filled circle (i.e. disk).
  - Color - color of the circle or disk to be painted.

Curve

With the Curve tool you can create freeform NURBS curves, trim curves, NURBS surfaces and extruded NURBS objects with freeform intersection, polygons, polyhedrons, polymids, and mesh map objects. The tool can also be used for 3D painting freeform curves to underlying objects that have bitmap textures.



The first option of the curve tool, Class, let's you choose what kind of objects to create.

Geometry - Creates an analytic polygon, polymid, polyhedron, or a NURBS curve or a NURBS mesh.



Type:

- *Analytic* - create polygon, polyhedron or polymid



  - Extrude - If set, creates extruded objects, i.e. polyhedrons and polymids.
  - Depth - Depth of the extruded object.
  - Freehand - If set, the polygon, polyhedron, or polymid is defined by moving the mouse while holding down the left mouse button.
- *Nurbs* - create NURBS curve or mesh



  - Extrude - If set, creates extruded objects.
  - Filled - If set, creates a surface whose boundary is the defined freeform curve
  - Order - order of the NURBS curve
    - Polygonal
    - Quadric
    - Cubic
  - Interpolate - If set, the curve passes through the defined points. Available only when a Cubic curve is created.
  - Freehand - If set, the curve is defined by moving the mouse while holding down the left mouse button.
  - Closed - If set, defines a closed curve or surface.
  - Pen - Create a NURBS surface using an existing, selected curve as a pen.
    - None - creates a curve, no surface.
    - Sweep - the selected 'pen' curve is swept along the curve drawn and rotated according to the curvature of the curve drawn.
    - Rail - the selected 'pen' curve is swept along the curve drawn. The pen object is not rotated while sweeping.
    - Rotate - the pen object is used as an axis and the new curve is rotated around it.

Map - Creates a mesh mapping.



- Order - order of the NURBS curve

  - Polygonal
  - Quadric
  - Cubic
- Interpolate - If set, the curve passes through the defined points. Available only when a Cubic curve is created.
- Freehand - If set, the curve is defined by moving the mouse while holding down the left mouse button.
- Closed - If set, defines a closed curve or surface.
- Class - class of the map created
  - Material - Map a material
  - Effect - Map a particle post effect
- Effect Name - name of the material or particle post effect to be mapped

Trim Curve - Creates a trim curve. A NURBS mesh must be selected.



- Order - order of the NURBS curve
  - Polygonal
  - Quadric
  - Cubic
- Interpolate - If set, the curve passes through the defined points. Available only when a Cubic curve is created.
- Freehand - If set, the curve is defined by moving the mouse while holding down the left mouse button.
- Closed - If set, defines a closed curve.

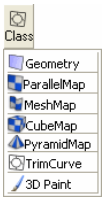
3D Paint - Paints curves to underlying objects that have bitmap textures.



- Order - order of the NURBS curve
  - Polygonal
  - Quadric
  - Cubic
- Interpolate - If set, the curve passes through the defined points. Available only when a Cubic curve is created.
- Freehand - If set, the curve is defined by moving the mouse while holding down the left mouse button.
- Closed - If set, defines a closed curve.
- Line Width - width of the curve to be painted.
- Filled - Select either filled curve or non-filled.
- Color - color of the curve to be painted.

Rectangle





With the Rectangle tool you can create objects with a rectangular cross section. I.e. rectangles, cubes, pyramids and cut pyramids. Also NURBS meshes and Subdivision Surface objects of those shapes can be created. Parallel Map, Mesh Map, Cube Map, Pyramid Map objects and rectangular trim curves can also be created. The tool can also be used for 3D painting rectangular shapes to underlying objects that have bitmap textures.

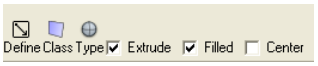
The Rectangle tool has the following options:

Define - How the rectangle is defined. The choices are:

- Corner - Corner
- Center - Corner

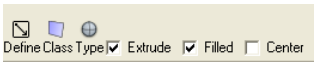
Class - Class of the rectangular object to be created.

- **Geometry** - Creates a geometry object.



Type - type of the geometry object

- *Analytic*



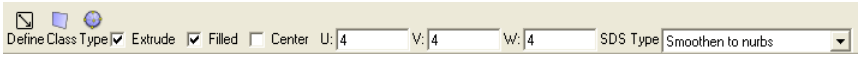
  - Extrude - If checked creates extruded objects, i.e. Cubes, pyramids, cut pyramids.
  - Filled - If checked, closes the caps.
  - Center - If checked centers the apex. Available only when creating extruded objects

- *Nurbs*



  - Order - order of the NURBS curve or surface
    - Polygonal
    - Quadric
    - Cubic
  - Extrude - If checked creates NURBS mesh objects shaped as Cubes, pyramids, cut pyramids.
  - Filled - If checked, closes the caps
  - Center - If checked centers the apex. Available only when creating extruded objects.
  - U - Number of points in the width (u) direction.
  - V - Number of points in the height (v) direction

- *Subdivision*



  - Extrude - If checked creates SDS objects shaped as Cubes, pyramids, cut pyramids.

- Filled - If checked, closes the caps
- Center - If checked centers the apex. Available only when creating extruded objects
- U - Number of points in the width (u) direction.
- V - Number of points in the height (v) direction.
- W - Number of points in the depth (w) direction.
- SDS Type - SDS object type. Possible values are:
  - Smoothen to nurbs
  - Smoothen to rectangles
  - Smoothen to triangles
  - Polygonal

- **Parallel Map** - Creates a parallel mapping.



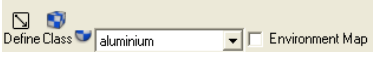
  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped

- **Mesh Map** - Creates a mesh mapping.



  - Order - order of the NURBS surface
    - Polygonal
    - Quadric
    - Cubic
  - U - Number of points in the width (u) direction
  - V - Number of points in the height (v) direction
  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped

- **Cube Map** - Creates a cube mapping.



  - Class - class of the map created

- Material - Map a material
- Effect - Map a particle post effect

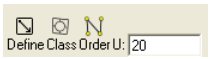
- Effect Name - name of the material or particle post effect to be mapped.
- Environment Map - check this to create an environment map.

- **Pyramid Map** - Creates a pyramid mapping.



  - Class - class of the map created
    - Material - Map a material
    - Effect - Map a particle post effect
  - Effect Name - name of the material or particle post effect to be mapped

- **Trim Curve** - Creates a rectangular trim curve.



  - Order - order of the NURBS trim curve
    - Polygonal
    - Quadric
    - Cubic
  - Filled - If checked, closes the trim curve
  - U: - Number of points in the trim curve

- **Paint** - Paints rectangular shapes to underlying objects that have bitmap textures.



  - Order - order of the NURBS curve
    - Polygonal
    - Quadric
    - Cubic
  - U: - Number of points in the painted line.
  - Line Width - width of the rectangular line to be painted.
  - Filled - Select either a filled or non-filled rectangle.
  - Color - color of the rectangle to be painted.

Sphere



With the Sphere tool you can create spherical NURBS and Subdivision Surface objects, analytic spheres, and sphere mapping objects.

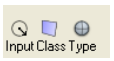
The Sphere tool has the following options:



Input - How the sphere is defined, by center and radius, two corners, or center and corner.

Class - class of the sphere created.

- *Geometry*



  Type - Type of the geometry

  - Analytic
  - Nurbs
  - Subdivision

- *Sphere Map*



  - Map - Type of class to map
    - Material - Map a particle post effect
    - Effect - Map a post particle effect
  - Effect Name - name of the material or particle post effect to be mapped.
  - Environment Map - check this to create an environment map.

User Interface

Geometry Managers

The geometry managers can be Drag&dropped to empty Drop Target windows.

To learn more about using the geometry managers in GUI editing see the User Interface / Customizing the User Interface chapter in the manual.

Placer



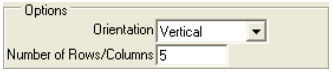
The placer can be used when you want to position the child windows or gadgets positions relative to the width and height of the window.

RowCol



The RowCol can be used when you want to lay the child windows or gadgets in multiple rows or columns.

The RowCol has the following options:



Orientation - Orientation of the RowCol (Vertical/Horizontal).

Number of Rows/Columns - Number of rows or columns in the RowCol

Packer

The packer can be used when you want to organize the children either vertically or horizontally.

The packer has the following options:



Orientation - Orientation of the packer, Vertical or Horizontal.

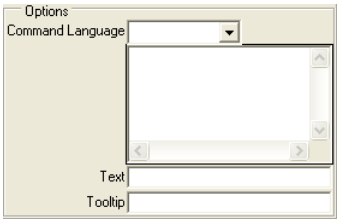
Macro GUI gadgets

The macro GUI gadgets can be inserted to the GUI by dragging from the buttons and dropping them to the GUI.

Macro Slider

The macro slider is a slider with an associated script. When the slider value is changed the script is executed.

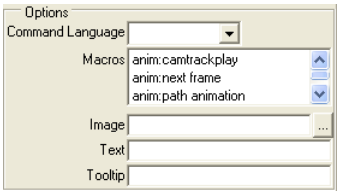
The macro slider has the following options:



- **Command Language** - Language of the script.
- **Script** - Script to be executed when the slider value changes.
- **Text** - Label text for the slider.
- **Tooltip** - Tooltip for the slider.

Macro Button

The macro button executes either a macro or a script when clicked.



The macro button has the following options:

- **Command Language** - Language of the script. Leave empty to select a macro instead of writing a script.
- **Script** - Macro to be executed when the button is clicked.
- **Macros** - Displays the list of available macros when no script language is selected. You can select the target macro from the list.
- **Image** - Image file to show on the macro button.
- **Text** - Label text for the macro button.
- **Tooltip** - Tooltip for the macro button.

Windows

The windows can be inserted to the GUI by dragging from the buttons and dropping them to the GUI. See the User Interface / Customizing the User Interface chapter in the manual to learn more about GUI editing.

Color Square

Color Square gadget for creating predefined colors.

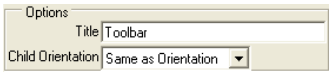
The Color Square gadget is a rectangle that shows a constant color. You can drag&drop the color from a Color Square gadget to a View window to set the color of the currently selected objects. The color can also be drag&dropped to any color gadget in e.g. property gadgets or tool gadgets. The color of the Color Square gadget can be set by double-clicking the gadget and specifying the color using the color dialog that opens.

Color

The color gadget allows to define a color numerically. You can drag&drop the color from the color canvas of a Color gadget to a View window to set the color of the currently selected objects. The color can also be drag&dropped to any color gadget in e.g. property gadgets or tool gadgets.

Client

Client windows are floating windows that can be docked to container windows.



The client window has the following options:

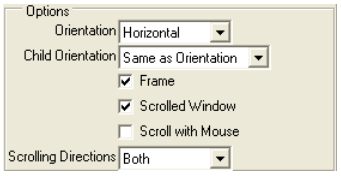
Title - Title for the window.

Child Orientation - Select whether and how the children's orientation follow the Client window's orientation.

- Same as Orientation - The children will have the same orientation as the Client window.
- Opposite - The children's orientation will be opposite to that of the Client window.
- Original - The children retain their native orientation.

Container Window

Container window is where Client windows can be docked.



The container window has the following options:

Orientation - Orientation of the container window. Vertical or Horizontal.

Child Orientation - Select how the child Client's orientation follows the Container window's orientation.

- Same as Orientation - The child Client will have the same orientation as the Container window
- Opposite - The child Client's orientation will be opposite to that of the Container window

Frame - Check to create a thin frame around the container window.

Scrolled Window - Check to make a window whose contents can be scrolled if they do not fit inside the window.

Scroll with Mouse - Check this if you want to scroll with the mouse.

Scrolling Directions - Select in which directions the window will scroll.

- Both
- Horizontal
- Vertical

Separator

Use this gadget to separate other windows.

The separator gadget has the following options:

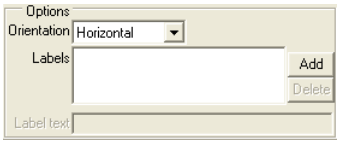


Orientation - Orientation of the separator gadget, Horizontal or Vertical.

Cycle

The cycle gadget has a drop down list gadget where different 'pages' can be selected. Other gadgets can be drag&dropped to the pages. Only one page at a time is visible.

The cycle gadget has the following options:



Orientation - Orientation of the cycle gadget, Vertical or Horizontal.

Labels - Shows the labels that are to be created in the cycle.

Add - Add another label.

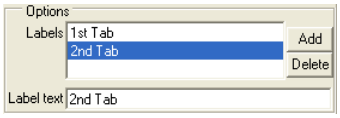
Delete - Delete the selected label. Select the label in the Labels gadget.

Label text - Change the selected label. Select the label in the Labels gadget.

Tab

The tab gadget has one or more labeled tabs. Selecting a tab opens a different page. Other gadgets can be drag&dropped to the tab pages. Only one page at a time is visible.

The tab gadget has the following options:



Labels - Shows the labels that are to be created in the tab control.

Add - Add another label.

Delete - Delete the selected label. Select the label in the Labels gadget.

Label text - Change the selected label. Select the label in the Labels gadget.

Frame

Creates a frame with text. Other gadgets can be drag&dropped inside the frame.

The frame gadget has the following options:



Text - Text label on the frame.

Checkbox - Check this if you want to create a frame that can be closed.

Splitter

Splitter is a gadget that divides its parent window either vertically or horizontally. The splitter can be dragged and moved with the mouse. Other gadgets can be drag&dropped to both sides of the splitter gadget.

The splitter gadget has the following options:

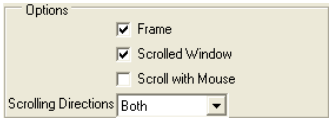


Orientation - Orientation of the splitter gadget. Vertical or Horizontal.

Window

Drop target window

The drop target window has the following options:



Frame - Check to create a thin frame around the drop target window.

Scrolled Window - Check to make a window whose contents can be scrolled if they do not fit inside the window.

Scroll with Mouse - Check this if you want to scroll with the mouse.

Scrolling Directions - Select in which directions the window will scroll.

- Both
- Horizontal
- Vertical

Load

Load a saved GUI object as part of the GUI.

The Load tool has the following options:



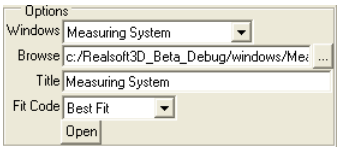
Objects - Select a GUI object to load.

Browse - Browse for a file containing a saved GUI object.

Float

Open a floating window.

The Float tool has the following options:



Windows - Select a predefined window or create a new one.

Browse - Browse for a file containing a saved floating window.

Title - Title for the floating window.

Open - Click here to open the floating window.

Windows

Advanced Selection Window

The Advanced Selection Window enables geometric object selection by object properties. For example, all material mapping objects hidden in a complex hierarchy can be selected here easily and made invisible to avoid wire frame clutter.

The window has three tabs for three different selection approaches: by name, by object class and by other object property. The three buttons on the bottom perform the actual selection operation after the condition has been defined:

- **Clear** deselects all objects providing a clean start for a new selection
- **Select** adds the objects that fulfill the given condition to the current selection
- **Deselect** removes the objects that fulfill the given condition from the current selection

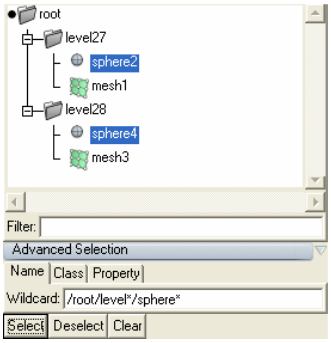


Note

Select operations do not perform a completely new selection but add and subtract from an already existing selection. This way you can combine several consecutive select operations.

Select operations search through the entire object tree (not the current Level only).

Only the condition set by the visible tab (Name/Class/Property) affects selection. Hidden conditions are not considered.

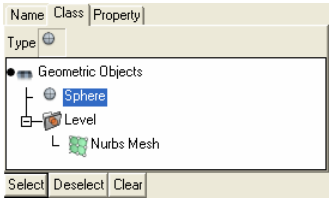


Selection by name wildcards

The string that specifies object selection should include full hierarchy path to the desired objects, for example: '/root/level*/sphere*'. The wildcard character asterisk '*' matches any number of consecutive characters.

Select By Class

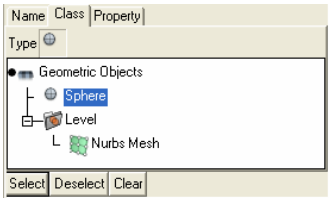
The 'Class' tab of the advanced selection window shows all object types included in the current project. You can pick for example 'Sphere' class and hit 'Select' to get all analytic spheres selected.



Selection by object class

Select By Property

The 'Property' tab shows all common on/off type attributes of geometric objects. Select adds objects, which have the activated property set. If 'Invert' is checked, select adds objects, which do not have the property set.

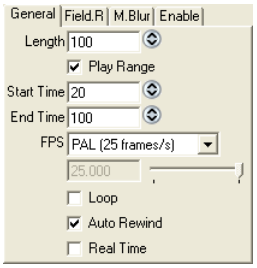


Selecting all 'wire frame invisible' objects

Animation Settings

The animation settings window is available as a 'docked' interface item below the select window and in the Windows pull down menu - the latter opens a floating window containing the animation controls.

The following animation options are available:



- *Length:* The length of animation measured using the current time system (Frames/ Real time/Seconds)
- *Field Rendering:* Activates field rendering. Field rendering means the method of rendering animation frames, which consist of two interlaced 'fields'. Fields split the time between frames to half. The result is smoother animation. Common video and broadcasting standards (PAL, NTSC, VHS, etc.) use fields.
- *Shift latter field up:* By default, the lines of the second field (which comes later in time) of a frame are computed below the lines of the first field. If this option is checked, it inverts the order.
- *Interlace latter field above:* By default, the lines of the second field are placed under the corresponding lines of the first field, when fields are combined into a single frame by interlacing them. The frame lines will then be as follows: line1 of field1, line1 of field2, line2 of field1, line2 of field2, etc. This option starts interlacing from the latter field: line1 of field2, line1 of field1, line2 of field2, line2 of field1, etc. This is known as starting with the "even" field (by default) or starting with the "odd" field. This setting is dependent on the output device you are using.
- *Motion Blur:* Turns motion blur on. Every image of the animation (field or frame) will be assembled from the given amount of sub images distributed evenly between the fields/frames. This will make motions appear smoother.
- *Samples:* The amount of sub images for motion blur. The higher the amount, the better the quality, but rendering time increases respectively
- *Loop:* If activated, animation playback rewinds and restarts automatically. This can be useful when previewing cyclic animations. Note that when using File Renderer, this option is ignored
- *Real Time:* If activated, animation playback speed is determined by the FPS (Frames Per Second) field. The system either skips frames, or delays rendering to synchronize the play back speed to the given FPS value. Note that simulations may not work correctly with this option.
- *Auto Rewind:* If activated, animation playback rewinds automatically at the end. When converting simulations to keyframe animations, auto rewinding can be turned off:
  - Turn auto rewind off
  - Set animation recording on
  - Play the simulation
  - After playback, select the root level and apply Copy tool to it
  - Rewind the animation
  - Delete the restored simulation hierarchy and paste the keyframe animation to replace the simulation.
  - Turn off Simulation for all simulated scene parts. In the choreography editor, remove all unnecessary animated attributes (velocity, spin etc). Apply the Simplify curve tool to the remaining keyframe curves as many times as necessary.
  - Turn auto rewind on again
- *Enable Simulations:* If set, physical simulations (gravity, collisions etc.) are computed during playback.

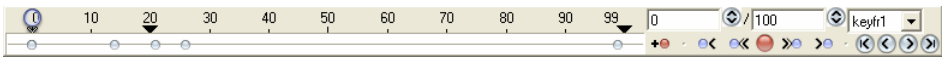
- *Animate Materials:* If set, material attribute changes can be keyframed by changing their value animation recording on. The option also controls if already keyframed material changes will be assigned during the playback.
- *Animate Image Effects:* If set, image effect attribute changes can be keyframed by changing their value animation recording on. The option also controls if already keyframed changes will be assigned during the playback.
- *Animate Particle Effects:* If set, particle effect attribute changes can be keyframed by changing their value animation recording on. The option also controls if already keyframed changes will be assigned during the playback.
- *Animate Images:* If set, image attribute changes can be keyframed by changing their value animation recording on. The option also controls if already keyframed changes will be assigned during the playback.
- *Play Range:* If set, the playback steps trough the given frame sub interval instead of the whole animation
- *Start:* The beginning of the 'Play Range' interval
- *End:* The end of the 'Play Range' interval
- *FPS:* Defines how many frames one second of animation contains. The common standards are available as presets.

Animation Window

The animation window is available in the default user interface at the bottom of the screen. It can be hidden and made visible using the pull down menu 'View/Animation Window' or by hitting F7 shortcut key.

The time line of the animation window and other time related controls follow the selected time display conventions, which can be specified in Options/Window/Animation. The window is synchronized with the Select Window showing the keyframes of the currently selected object / choreography. For example, to edit key frames of a material object, click the Material tab of the select window, select the desired material and then select the desired keyframer choreography

The following controls are available:



Play range start: Defines the beginning of the animation playback range. The control is visible only when the Play Range option is enabled. Dragging the triangular marker with the left mouse button changes the play range start.



Play range end: Defines the end of the animation playback range.



Time slider: Shows the current time of the animation on the timeline. Dragging the knob with the left mouse button changes the current time.



LMB clicking the time line around the knob changes the time one frame to the respective direction. RMB click on the time line moves the knob to the clicked position. Also the following modifiers are supported:

- Shift + RMB click moves the knob the the nearest key frame.
- Alt + RMB click plays the knob to the nearest key frame.

Current: Shows the current frame of the animation. Entering a new frame number to this field jumps (not plays) the animation time to the new frame.



Field: Shows the current field (odd/even). The field control is visible when Field Rendering is enabled. The current field can be changed using the gadget.

Sample: Shows the current motion blur sample. Entering a new index to the gadget changes the current sample. This gadget is visible when Motion Blur is enabled.



Playback controls: Rewinds the animation back to the first frame (or to the beginning of the play range if the range option enabled). Plays the animation (or a defined sub range) backwards or forwards and jumps to the last frame.



Record: Turns animation recording on/off. When recording is enabled, all object modifications become recorded into the current choreography. If the current choreography is not a key framer, one is automatically created.



Total: Controls the amount of frames in the animation. If the frame count is changed and the current project contains animated objects, the program asks if the time lines should be scaled respectively.



Keyframes: Shows the keyframes of the currently selected objects. Key frames can be edited by dragging them with the mouse. The keyframe gadget can be zoomed and panned using the standard Shift and Ctrl modifiers while dragging with RMB.



- Attributes - Sub menus can be used for enabling / disabling attributes for key framing. If an attribute is disabled, editing attributes with animation recording on will not create new key frames for it.
- Show Choreographs, Hide Choreographs - These menus can be used for controlling the visibility of choreographs in the Choreography Window. Useful when managing scenes consisting of hundreds or thousands of animated objects.
- Up, Down - These menus can be used for controlling the current choreography. Only choreographs up to the current choreography are executed.
- Reset - Animation time line can be zoomed and panned via RMB + Ctrl and Shift keys. This menu restores the default scale and pan values.
- Fix attribute list - This menu fixes the current animated attribute list so that no new attributes will be animated. However, the attributes already animated will be treated as usual.

Choreography stack: Shows the choreography stack and the current choreography.



Choreography stack contains all the choreographs applied to the object. The first choreography (typically named 'init') defines the initial state of the attribute in question. Any number of choreographs can be stacked on top of each other. Keyframer choreography can be used for animating attributes. It is automatically created when the user modifies objects when animation recording is activated. Pose choreography is typically created when the user modifies animated objects with animation recording off.

Create New Key: Creates new keyframes to the current choreography. This tool is available only when the type of the current choreography is 'Keyframer'. Another way of creating keyframes is by editing objects with animation recording on.

Previous Key: Moves the current time knob to the previous key frame.

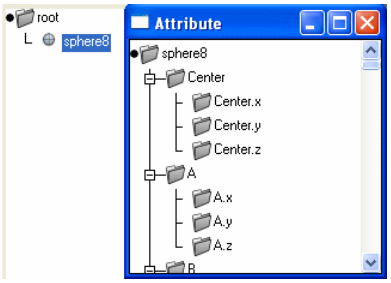
Next Key: Moves the current time knob to the next key frame.

Play to previous key: Plays animation to the previous key frame.

Play to previous key: Plays animation to the next key frame.

Attributes Window

The attributes window shows the list of an object's public attributes. The object whose attributes should be displayed can be selected by dragging and dropping it from the select window to the attributes window. All object types (not only geometric objects, but also image effects etc.) are valid for this purpose.



The displayed attributes can be drag and dropped into the scripting window. This helps in avoiding typing errors.

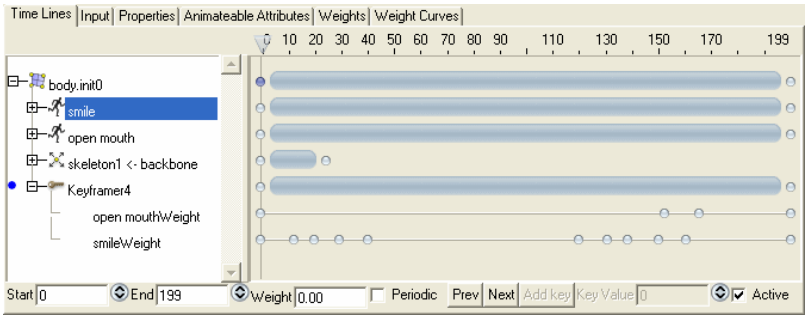
Choreography Window

The choreography window contains the tools for editing animation timing and other animation related features. The standard program configuration includes a floating choreography window which can be opened from the 'Animation/Choreographs..' pull down menu or by hitting F6 hotkey.

The window consists of several tabs, which are listed below.

Time Lines

The Time Lines tab consists of a choreography list, a time line table and a number of controls on the bottom of the window. Moving the splitter line that divides the window vertically controls the space division between the choreography list and the time lines.



The following controls are available:

Choreography list: Shows all choreographs of the current project (or alternatively only the choreographs of the selected objects - see the Only Selected Objects option below). Each animated object has an initialization choreography, which defines its state in the beginning of the animation. An initialization choreography can have any number of sub choreographs, which are executed in the order they appear in the choreography list. A LMB click activates a choreography, after which its properties can be modified using the controls on the bottom of the window, with popup menus and with other tabs of the window. The same list is visible also in many other tabs of the choreography window.

A double click on a choreography with the left mouse button opens the sub structure of the choreography, showing the sub choreographs or the animated attributes (and their time lines on the right side) included in the choreography. A second double click hides the sub structure. A complex animation often contain very many choreographs. Keeping only choreographs under construction opened helps in managing the complex time line system.

A slow double click activates the rename tool. Assigning descriptive names for choreographs helps to manage complex animations. For example, facial expression choreographs can be renamed as 'smile', 'cry', 'anger' etc.

The choreography list has the following popup menu:

- *New/Keyframer, ...:* Creates a new sub choreography for the target object of the selected choreography. The time line is initialized to fill the parent time line.

- *Duplicate:* Duplicates the selected choreography. Initialization choreographs cannot be duplicated.
- *Split:* Uses the current time to cut the time line of the selected choreography into two parts. The result is two key frame choreographs where the original defines motions before the cut and the newly created choreography defines the motions after the cut point. The actual motions remain unchanged. If the tool is applied to the root choreography, all keyframe sub choreographs are split.
- *Delete:* Deletes the selected choreography. If the deleted choreography is the initialization choreography, the object becomes unanimated.
- *Rename:* Activates the rename tool (slow double-click does the same).
- *Open:* Open the sub structure of the selected choreography.
- *Close:* Close the sub structure of the selected choreography.
- *Open All:* Opens the sub structure of all choreographs.
- *Close All:* Hides the sub structure of all choreographs.
- *Only Selected Objects:* If set, the usual object selection at the select window controls which choreographs are shown. This helps to control and edit complex animations.
- *Load:* Loads a choreography from disk and assigns it to the target object of the selected choreography.
- *Save:* Saves the selected choreography to a disk file.

Time lines: The time line gadget shows a time scale, a slider for changing the current time and a table of horizontal time lines. Dragging the knob on the time scale with the left mouse button changes the current time. A RMB click moves the time to the clicked position.

A time line can be edited by dragging the end knobs with the left mouse button, or by using the numeric 'Start' and 'End' controls on the bottom of the window. The time line can be moved by dragging it.

By default changing the lenght of a time line scales also the associated key frames. This can be avoided by holding down the Alt key, in which case the associated keys are scaled inversely so that their positions in global time remains unchanged.

If the sub structure of a keyframer choreography is opened, the time line table shows also the attribute specific animation keys. Animation keys are shown as rectangular knobs. Keys can be activated by LMB clicking them. Shift modifier allows key multi-selection. LMB-dragging can also be used for multi-selection. Keys can be moved by dragging them with the mouse. Selected keys can be moved also using the 'Key Value' gadget below the time line table. The Shift modifier actives frame grid when knots are being moved.

Keys can be duplicated by draggning the selected keys while holding down the Ctrl key.

For accurate editing, you can zoom in by dragging with the right mouse button Shift key held down. Ctrl-RMB-dragging pans the time lines (i.e. selects the time range to be displayed on the window). Mouse wheel can be used for scrolling the window.

The time line area has the following popup menu:

- *Delete Key Values:* Deletes a selected key.
- *Delete All Keys:* Deletes all keys of attributes, which have one ore more keys selected. Those attributes become unanimated.
- *Reset:* Resets zooming and positioning to show the full animation length.

Start: The start time of the selected choreography (in frames or real time depending on Options/Animation/Time System).

End: The end time of the selected choreography.

Weight: The weight of the selected choreography in the current frame.

Periodic: Repeats the selected choreography outside the actual time line.

Prev: Jumps the time to the previous key frame. Only keys show on the time line table are considered.

Next: Jumps the time to the next key frame. Only keys show on the time line table are considered.

Add key: Adds a new key frame at the current animation frame to the selected attribute or to all attributes of the current choreography.

Key value: The time of the selected key(s). If multiple keys are selected and their values are different, a red color indicates value conflict.

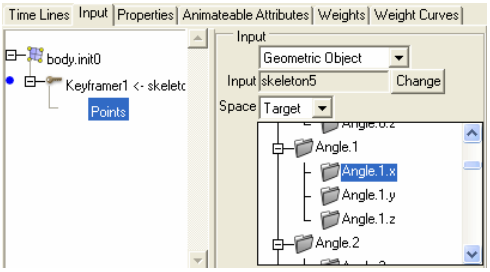
Active: Activates the selected choreography.

Input

The Input tab controls choreography parameterization. By default, choreographs are parameterized by the usual animation time, but here you can set up other kind of dependencies as well. For example, thickness of a muscle can depend on a skeleton angle.

The input parameter can be changed as follows:

By selecting an object from the select window, then choosing the corresponding object type from the selector in the 'Input' area and pressing the 'Change' button. After that, the choreography window displays the attributes of the chosen object and you can activate a suitable parameter for the choreography. For example:



- Animate the desired property (for example, the scale of a muscle).
- Go to the Input tab of the choreography window. Select the keyframer object which animates the property.
- In the select window, select the object, which should control the animated property. It can be another geometry object, a material etc.
- In the choreography winsdow, set the input object type gadget to match the object selection you did. If you selected a material, set Input=Material here, too.
- Hit the Change button.
- Select a suitable attribute from the displayed attribute list.

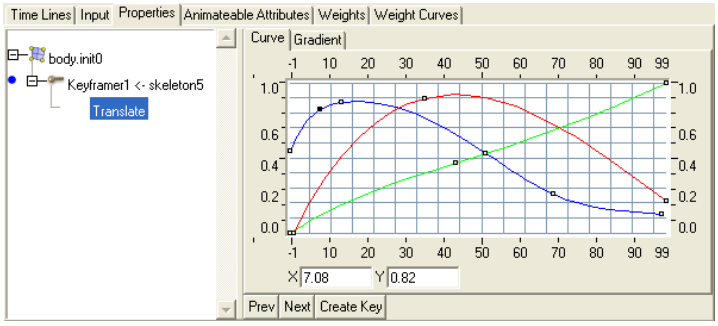
To summarize, the input control gadgets are:

- **Input type selector:** Selects the type of the input object for 'Change' operation. The available types are:
  - *Time*
  - *Geometric Object*
  - *Material*
  - *Particle Effect*
  - *Image Effect*

- ***Change:*** Changes the input object as specified by the input type selector and object selection.
- ***Attribute list:*** Shows the list of attributes of the chosen input object. The input parameter can be selected here.

Properties

The Properties tab shows the properties of the currently selected choreography. If a keyframe choreography is selected, the tab shows the actual animation curves, which define the animated attribute values. The attribute for editing can be selected by opening the sub structure of a choreography and clicking an attribute. The following keyframe specific controls are available in this tab:

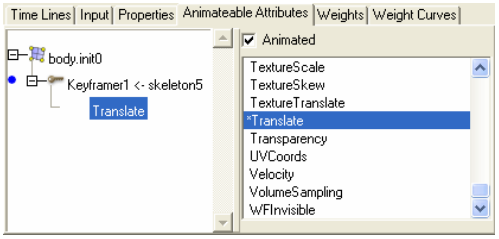


- ***Curve Gadget:*** A standard curve gadget for editing the selected animation curve.
- ***Prev:*** Jumps the time to the previous key frame of the animation curve (other hidden attributes do not count).
- ***Next:*** Jumps the time to the next key frame of the animation curve
- ***Create Key:*** Creates a key frame to the currently edited animation curves at the current animation time.

For more information about other type of choreographs, consult Choreograhy Classes reference manual page.

Animateable Attributes

This tab shows the list of all animateable attributes of the target object of the selected choreography. A new attribute can be selected from the list and made animateable using the 'Animated' checkbox above the attribute list. After that, the animation curve can be initialized at the Properties tab using the predefined curve types and edited to a suitable form. Clearing the 'Animated' checkbox for an attribute deletes the corresponding animation curve.





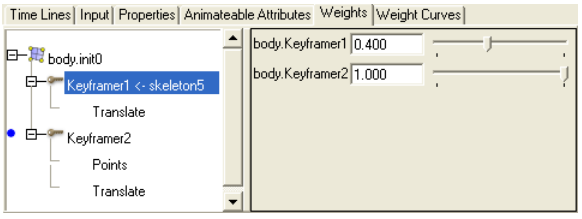
Note

The program automatically adds the attributes of the existing choreographs into a new choreograph.

Weights

The Weights tab contains a choreography weight slider for each choreograph of the current project. Decreasing the weight of a choreography from the full level 1.0 reduces the effect of the choreography. At zero level, the choreography no longer animates the target object.

Choreography weights can be animated as well. You can achieve this by creating a new keyframer choreography and adding the desired weight attributes to it in the Animateable attributes tab. Then you can edit the weight curves of the Weight Curves tab. You can also change time and weight values while animation recording is turned on.



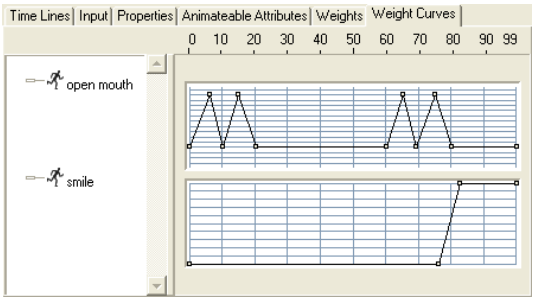
Weight Curves

The Weight Curves tab shows the animation curves, which control the weight of choreographs.



Note

A choreography may not have a weight curve. You can make the weight animateable as follows:



- Create a keyframer object to the end of the choreography list of the animated object
- Edit the weight sliders with animation recording mode enabled.

The left side of the 'Weight Curves' tab has the same popup menu tools as the time lines tab, plus some additional menu items for controlling curve visibility:

- *Show Curve:* Shows the weight curve of the selected choreography.
- *Hide Curve:* Hides the weight curve of the selected choreography.
- *Show All Curves:* Shows the weight curves of all choreographs.
- *Hide All Curves:* Hides the weight curves of all choreographs.

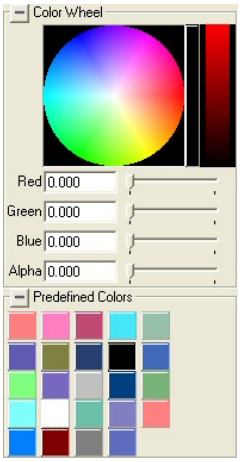
The actual curves are displayed using the standard curve gadget. The curves can be edited here.

You can also set weight curves up manually, as follows:

- Create a keyframer object to the end of the choreography list of the animated object
- Add the weight parameter into it using the 'Animateable Attributes' tab
- Use the popup menu of the curve gadget to initialize a curve for further editing.

Color Window

The Color Window is available at the 'Windows' pull down menu. The hotkey F12 of the default key map opens and closes the window. The color gadget of the color window is used as a component in many property sheets, tool controls etc. in a reduced form (without 'Predefined Colors' array). The documentation below is valid for all such color gadgets.

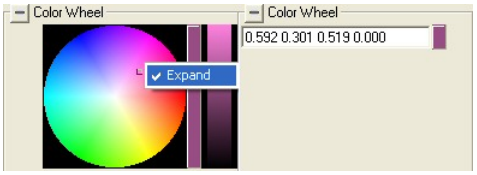


The floating color window opened from the 'Windows' pull down menu does not directly change any color attribute in the scene. It acts as a tool where you can blend colors and optionally store them to the 'Predefined Colors' array. The actual use of created colors happens by dragging and dropping colors from the color window to color gadgets of object property windows, to tool controls etc. Colors can be fetched into the color window the same way, using drag and drop.

The following controls are available:

- *Color Wheel:* You can pick a suitable base color from the circular color wheel gadget. The high, narrow color sample rectangle immediately at the right side of the wheel displays the result color while you LMB drag the mouse on the wheel. The slider at the right adjusts the darkness of the selected color. The color wheel acts as a drop target: dropping a color from another color gadget onto it sets the current color.

- *Current Color Area:* This rectangular area displays the result color defined using other controls of the window. Despite of its small size, it provides some useful functionality:
  - A LMB click opens the standard color dialog of the operating system. You can use it as an alternative way to define the color.
  - A single popup menu 'Expand' is available on it. If cleared, the color wheel and related controls collapse into a minimal state where only a single numeric rgb(a) field and the current color area visible.
  - You can drop colors into this area and drag the color from it into other color gadgets.
- *Brightness Bar:* The vertical bar on the right adjusts the brightness of the selected color. The scale starts from black and ends to the color specified using the wheel. LMB click or drag the bar to change the brightness.
- *Red, Green, Blue, Alpha:* The numeric components of the current color. In the expanded state, the values can be controlled using separate sliders. In collapsed state, a single field defines values, sub components being separated with empty spaces. The alpha channel is optional and not available in all contexts. Some tools, which need the alpha value, show the gadget with the alpha control. For example, the 3D paint airbrush uses the alpha value to define paint transparency.
- *Predefined Colors:* The array of predefined colors is not available in all color gadgets. When available, you can drag colors from it and drop them into other color gadgets. You can also store colors to this array using drag and drop (for example from the color sample area of the color wheel).

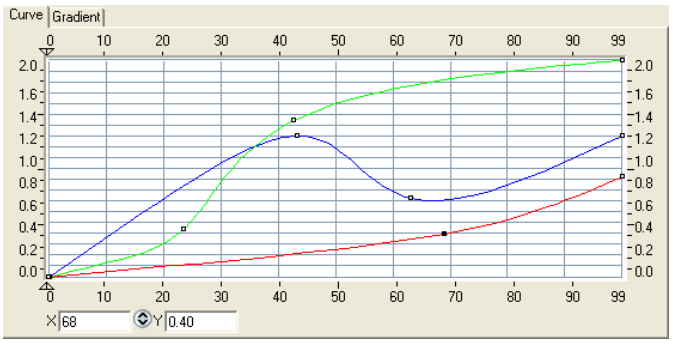


The Color Wheel in expanded and collapsed states

Curve Editor

Two-dimensional curves are used for many purposes in Realsoft 3D: as a VSL shading object, for controlling animations etc. Consequently, The Curve Editor documented below can be encountered in many areas of the user interface.

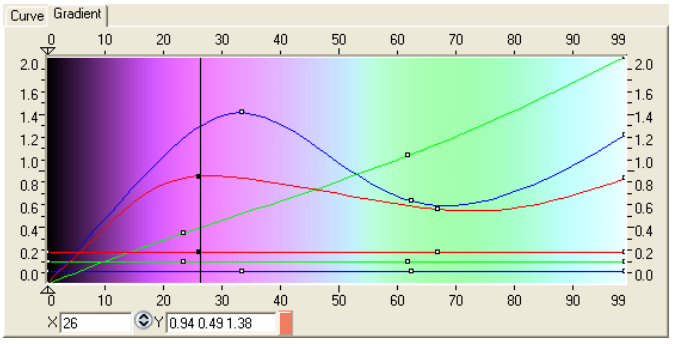
The curve editor can be used in a default state where the 1-4 transformation curves are visible. A curve object consisting of one sub curve is suitable for transforming time, whereas a three-component curve can be used for rgb transformations. The sub curves are colored as 1.=red, 2.=green, 3.=blue, 4.=black.



The Curve Editor in the default state

The Gradient state of the curve editor shows a curve-transformed rgb gradient on the background of the editor. This is especially convenient when designing color gradients for materials or when viewing animated color attributes. It is much more intuitive to see the actual color than three curves representing numeric rgb components.

The curve editor has only a few controls: the actual graphical area showing the curves, and two floating value gadgets for accurate control of selected curve points.



The Curve Editor in the gradient state

Mouse Navigation

The following mouse operations are available for quick navigation:

- *Shift+right mouse button dragging:* Zooms the curve editor in and out. Horizontal dragging zooms the x-scale, vertical dragging the y-scale.
- *Ctrl+right mouse button dragging:* Pans the curve editor.

Point Editing

Adding points: You can easily add a point to a curve by clicking the curve. A new point is inserted to all visible curves that were hit and which did not have a point at the clicked position.

Deleting points: Select the points and hit Del key or select 'Delete' from the popup menu.

Moving points: Select the points. Then press the left mouse button down on top of a point and move the mouse holding the left button down. Release the left mouse button to finish moving. You can cancel the move operation with the 'Esc' key. It cancels the operation, but it does not undo already moved points.

Point Selection

The point selection rules are similar as in object or 3D handle selection. You can select points by clicking or dragging with the left mouse button. Key modifiers add useful selection functionality. Without a modifier, a selection operation always clears the previous selection. Clicking or dragging empty area deselects all points.

The summary of point selection operations is shown below:

- *LMB click :* Select a new point
- *SHIFT+LMB click :* Add a single point to selection
- *CTRL+LMB click :* Add a selection range from the first previously selected to the clicked point
- *SHIFT+CTRL+LMB click :* Toggle selection
- *ALT+LMB click :* Deselect a point
- *LMB drag :* New selection. Deselects previous points and select new points inside the drag box
- *SHIFT+LMB drag :* Add operation. Adds points inside the drag box to selection
- *SHIFT+CTRL+LMB drag :* XOR operation. Toggles selection state of points inside the drag box
- *CTRL+LMB drag :* And operation. Points inside the drag box AND which were already selected become selected
- *ALT+LMB drag :* Subtract operation. Deselect points inside the drag box

The Popup Menu

Delete Points: Deletes the selected points. Note that a valid curve has always at least two points. The delete tool does not allow point reduction below this minimal amount.

Create Predefined Curve: Initializes an empty curve to a simple default shape. This operation is available for example in animation curve editing where new animated attributes may not have any curve initially defined. The following alternatives are available:

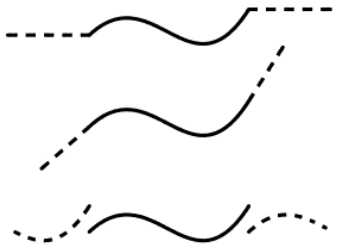
- *Constant Curve:* A horizontal line with a constant y value.
- *45 Degree Curve:* A diagonally ascending line.
- *-45 Degree Curve:* A diagonally descending line.
- *Bell:* A smoothly bending curve with horizontal ends.

Type: Selects the type of the curves from the following alternatives:

- *Nurbs:* A smooth, cubic NURBS curve is fitted to the points. The curve is C2 continuous everywhere, which property is significant in animation use.
- *Polyline:* Linear interpolation between the curve points.
- *PNurbs:* A less continuous curve type than Nurbs. Adapts better to point density variations.
- *Boolean:* A stair curve. Values jump at curve points.
- *Bezier:* The usual cubic Bezier curve, consisting of point quadruples. This curve type allows combination of smoothly curved and linear sections. A Bezier curve is not as continuous as a Nurbs curve at the quadruple joints.

Extrapolation: Defines the result of curve evaluation outside the range of curve points.

- *Flat:* The default choice. The value of the first point is applied for parameters before the curve starts. For parameters after the curve end, the result is defined by the last point's value.
- *Linear:* Curve is extended linearly to the tangent directions at end points.
- *Repeat:* The curve shape is repeated in a cyclic manner.



From top to down: Flat, Linear and Repeating extrapolation

Fit View to Curve: Resets the curve editor after zoom and pan operations so that the current editing range (see 'Set Minimum and Maximum Values' below) matches exactly the window size.

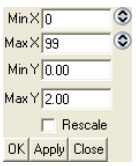
Simplify: Removes unnecessary points of the curves i.e. reduces the point count without changing the curve shape.

Align: Changes the y values of the points between the first and the last selected point of each visible curve so that the points match a straight line.

Set Minimum and Maximum Values: Opens a window for controlling the parameter and output range of the curves. The curve editor allows editing within a certain numeric range. You can define the range using this window.

The scaling dialog has the following options:

- *Min X, Max X, Min Y, Max Y:* The editing range limits. Entering new values does not immediately change the scale; you have to press the 'OK' or the 'Apply' button to set the new limits.
- *Rescale Curve:* If set, the curves are scaled to match the change in editing range.
- *Apply:* Sets the new scale. The scaling window is left open.
- *OK:* Sets the new scale and closes the scaling window.
- *OK:* Closes the scaling window without changing the scale.



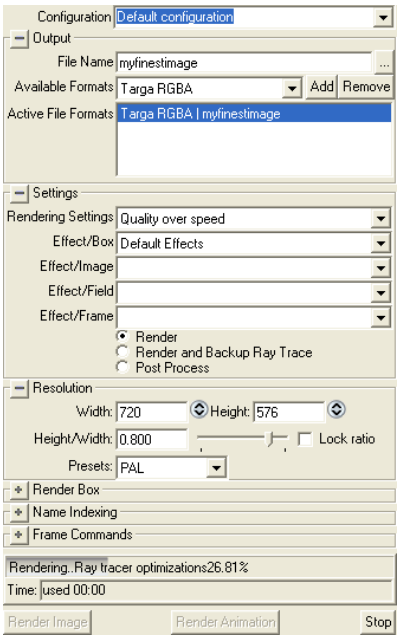
Red, Green, Blue, Black Curve: These menus control visibility of the component curves. When editing complex curves, it may be necessary to hide all but one curve. The hidden curves do not interfere with editing actions.

Show All Curves: Makes all component curves visible.

File Rendering Window

The File Rendering Window opens from the File/Render pull down menu (hotkey Ctrl R). With this window, you can render images and animations to disk files.

The top area of the window contains controls for a file rendering configuration object. The bottom area includes the following gadgets:



- **Time** A progress indicator showing the time used for rendering and the estimated total time.
- **Render Image** This button starts rendering of a single image.



Note

If motion blur or field rendering is active, use the Render Animation button to render all sub images required for one complete frame.

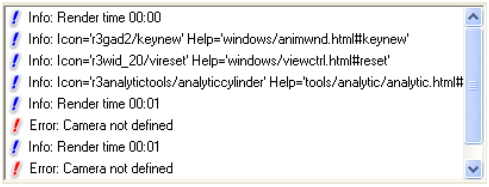
- **Render Animation** Renders all images of the current animation (or a sub interval defined by the Play Range animation option).
- **Stop** Cancels file rendering.

  In animation rendering, already rendered separate images will not be deleted. However, an intermediate animation file such as an avi file will be deleted.

  In single image rendering, the user will have an option to keep already rendered image boxes, provided that the box rendering feature is used. If the sub boxes are kept, rendering can be continued later by using the existing image sub boxes.

Info Window

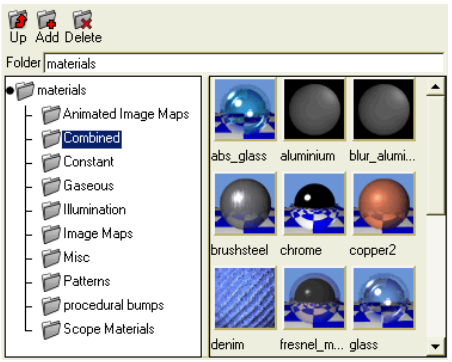
The Info Window opens from the Windows pull down menu. The window shows the history of various diagnostic messages given by the program. For example, when encountering an error in scene rendering because of some missing scene element, the warning dialog shows only the latest error message, whereas the message history on the info window may give a better idea of why the rendering failed.



Material Browse Window

The Material Browse Window opens from the material tab pop up menu of the Select Window. With this window, you can load and save materials easily.

The controls of the window are:



- **Up** Opens the parent folder of the current disk folder.
- **Add** Loads the material from the disk file whose icon is selected on the browse window.
- **Delete** Deletes the disk file containing the material whose icon is selected on the browse window.
- **Folder** The name of the current folder.
- **Folder Tree** Displays a disk folder structure. By default, the 'materials' folder is shown.
- **Material Icon Array:** Shows the material files of the selected folder. You can perform the following actions with it:
  - Double clicking a preview icon on the material browse window inserts it in the current library.
  - Drag and dropping a material icon from the browse window into the select window loads the material.
  - Drag and dropping a material icon to a view window loads the material and maps it to the selected geometric objects (just like drag and drop from the select window's material tab to a view window).
  - Drag and dropping a material from the select window to the icon array area saves the material to a disk file using material's name as the file name.

Numeric Window

The Numeric Window is docked to the bottom of the screen in the default configuration. You can open and close it from the View pull-down menu or using the F8 hotkey.

The numeric window is context sensitive. When using various tools of Realsoft 3D, the window makes you available the controls required for performing the operation by exact numeric input instead of mouse clicks. If the numeric window is open, hitting the 'n' (=numeric) key activates the window and moves the input focus to the field which represents the current interaction step. One can then just type the values and hit Enter to proceed to the next input step or to finish the tool. The Tab key moves the focus to the next field without entering a value.

The decimal accuracy and measuring units can be chosen from the File/Preferences pull down menu.

The following controls are always available on the numeric window:



Numeric window during a rotate operation

Position: The position of the 3D projected mouse pointer in absolute space coordinates. You can always enter a coordinate triplet to this field instead of LMB clicking the mouse. Note that the coordinates are usually projected to the input plane, unless the tool allows free 3D values (for example the NURBS curve tool does this).

Relative: The relative 3D position. When no interaction has been started, the relative position is measured from the grid origin. The grid origin acts therefore as a handy secondary measuring origin, which can be moved easily to a suitable place. When a tool interaction is already going on, the relative coordinates are measured from the previous entered point.

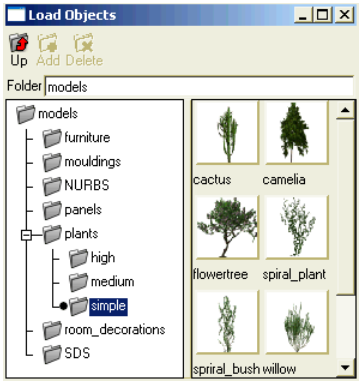
Polar: If set, relative coordinates are measured as angle coordinates (heading, pitching, distance). For example, when drawing a poly line, you can define a 20 mm long, 35 degrees descending line segment using this feature.

Other context sensitive numeric controls are described in the tool documentation.

Object Browse Window

The Object Browse Window opens from the object tab pop up menu of the Select Window. With this window, you can load and save objects easily.

The controls of the window are:



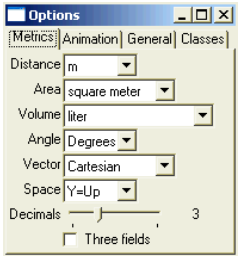
- **Up** Opens the parent folder of the current disk folder.
- **Add** Loads the object from the disk file whose icon is selected on the browse window.
- **Delete** Deletes the disk file containing the object whose icon is selected on the browse window.
- **Folder** The name of the current folder.
- **Folder Tree** Displays a disk folder structure. By default, the 'models' folder is shown.
- **Object Icon Array:** Shows the object files of the selected folder. You can perform the following actions with it:
  - Double clicking a preview icon on the object browse window inserts it in the current library.
  - Drag and dropping an object icon from the browse window into the select window loads the object.
  - Drag and dropping an object icon to a view window loads the object and translates it to match the current mouse position.
  - Drag and dropping a object from the select window to the icon array area saves the object to a disk file using object's name as the file name.

Options Window

The Options Window opens from the File/Preferences pull down menu. The window consists of four tabs described below. The window includes a large set of seldomly changing controls (user specific options, not project specific).

Metrics

Measuring unit and decimal accuracy settings.



- *Distance:* Defines the length measuring unit. All program controls that measure distance or 3D space coordinates use the selected unit.
- *Area:* Defines the area measuring unit.
- *Volume:* Defines the volume measuring unit.
- *Angle:* Selects if angles should be measured using degrees or radians.
- *Vector:* Selects if 3D points should be represented in the usual (x,y,z) Cartesian form or in the Polar coordinate form (heading, pitching, radius). In the latter form, heading defines the projected direction in the horizon plane and pitching defines the angle with the horizon plane. Radius is the distance from the origin.

- *Space:* Defines the world space orientation, Y=up axis or Z=up axis.



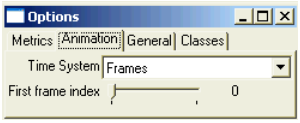
Note

Y = up axis is the native system. All data is internally stored this way. Scripting languages and other low level systems always access the data in the native form.

- *Decimals:* The number of displayed decimal units for all floating point gadgets in the program. For example, a value 1.238 is displayed as 1.24 if decimals is set to two. Note that this does not round the actual value: you can enter floating point values with as many decimals as you like, and they are all taken into account. Note also that the amount of decimals is independent from the measuring unit. Two decimals may be sufficient if you are using millimeters, but 4-5 is more appropriate when using meters.
- *Three fields:* If set, 3D coordinate gadgets consist of three separate sub fields. Then each sub coordinate can be adjusted separately with the dial control and the mouse. Also, changing one coordinate only (leaving two other intact) for a set of selected objects is easier.

Animation

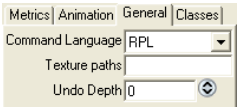
- *Time system:* You can choose one of the following three alternatives for measuring time:
  - *Frames:* Time is measured using a frame index. The number of frames per second can be selected from Animation Settings/FPS control
  - *Realtime:* Time is measured with three values: minutes, seconds, frames.
  - *Seconds:* Time is measured in seconds.



- *First frame index:* Selects if the first frame index should be zero or one. For example, in a one hundred frame animation, frame indices run from 0 to 99 (indexing starting from zero) or from 1 to 100 (indexing from one).

General

- *Command Language:* Defines which language will be used for scripting (RPL, TCL, JavaScript or another installed language).
- *Texture paths:* You can list your image search paths here. For example, after importing a complex project and placing its textures into a project specific folder, you can make all materials which use textures to work simply by adding the folder into this setting. Several folders can be defined by separating them with a semicolon. For example: 'textures;c:\tilepatterns;mytextures;'

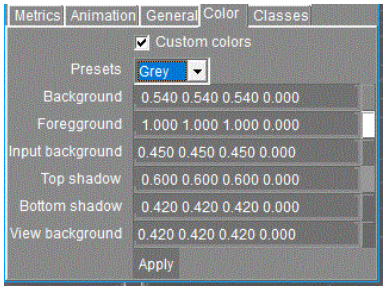


- *Undo Depth:* Defines the amount of successive steps recorder into the undo buffer.

Color

The Color tab is where you can change the colors the application uses for painting various display elements.

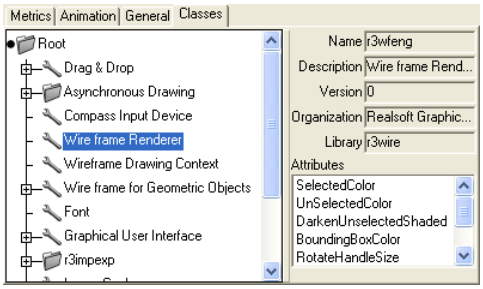
- *Custom colors*Overrides the default operating system specific color.
- *Presets* List of predefined colors. You can select one of these to define initial values for the color fields below. Note that selecting a preset does not prevent you from editing individual color fields.
- *Apply* Applies the defined colors.



Classes

The Classes tab of the Options window shows all public attributes of the installed program modules. For example, to change the highlight color of point edit handles from the default yellow to green:

- Go to the 'Classes' tab of the options window and find the item 'Wireframe Drawing Context' from the class tree. Activate it by clicking it with the left mouse button.
- The right side of the options window now shows the list of attributes for the wireframe drawing context object. Click 'Handle Fill Color'.
- Below the list, a gadget shows the current value for the selected attribute. Change yellow (1,0,0) to green (0,1,0).



Now close the options window and create an object. Hit the spacebar to activate the point edit mode; then select some points to see that selection is indicated with the new green color.



Note

Many class attributes are read only at the program startup. If you do not see any change after setting a class attribute value, just restart the program.

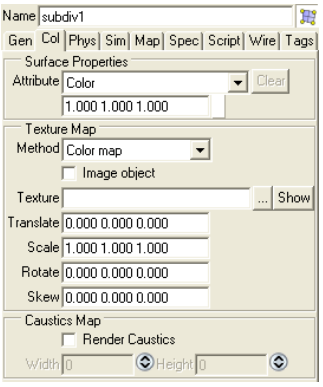
Property Window

The general purpose Property Window can be opened in many ways:

- From the popup menu 'Properties...' of the select window
- From the popup menu 'Object Properties...' of the view window
- From the 'p' hotkey
- From the 'Windows/Property Window' pulldown menu

In the default user interface, the property window is placed on a floating window. The general purpose property window is context sensitive; it adapts to the type of the selected objects. The select window's visible tab and the objects being selected in the tab together define what controls the property window includes.

There are also some 'sticky' versions of the property window available: they always show properties of selected objects for a certain object layer (independent of select window's tab selection). For example, the following windows which are all available in the 'Windows' pull down menu are such sticky property windows:



- Geometric Properties
- View Properties
- UV View Properties
- VSL Property Window

The property window has only one gadget that is common to all object classes: the **Name** gadget. You can rename one single object as well as several multi-selected objects using it. Remember to hit Tab or Enter key to confirm renaming before closing the property window.

The object property windows can also show properties of multi selected objects. If the values are different, the value of the first selected object is shown using red conflict color. Changing the value sets the same value to all selected objects (hence clearing the conflict state).

The object specific documentation about the property window can be found from the Models section of the reference manual.

Scripting Window

The Scripting Window opens from the 'Windows' pull down menu. Using this window, you can access and change the data of the program envinronment, loaded projects etc.



Command Language field lists all installed scripting languages. JavaScript is usually available there. The entered commands will be interpreted using the selected language.

You can write script language commands and expressions to the text box under the language selector. The text box provides basic text editing tools. You can naturally paste scripts instead of typing them here.

The second text box shows the output of scripts.

Expression field on the bottom of the window evaluates mathematical expressions.

Select Window

The select windows shows all project and working environment related objects that has been loaded from the hard disk into the computer's memory. You can 'select' or 'activate' items from this window in order to modify them or perform other available operations on them.

The select window can be hidden and made visible with the F2 hotkey, and from the 'View/SelectWindow' pull down menu.

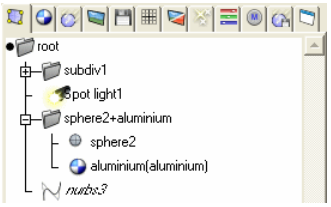
The top area of the select window contains a list of opened projects. The list can be hidden and made visible using the **Projects...** check box. The current project can be changed by opening the project list and clicking an item. Usually there is at least two projects included: the current working project and a material preview project. The 'File/New' pull down menu adds a new empty project into the list and 'File/Close' closes the selected project (note: material preview

cannot be closed). Closing the last working project automatically creates a new empty one. It should be noted that 'File/Load' loads into the current project instead of adding a new project.

A project can be renamed using the **Name** field below the project list. Renaming a project will change the suggested file name when the project is saved next time.

Geometric Objects

The Hierarchy Tree: Displays the tree like structure of the geometric objects. Geometric objects are usually grouped using level objects to obtain a sensible and manageable object hierarchy. A suitable part of the total structure can be activated here, and parts can be moved in hierarchy using drag and drop. See the Mouse Operation documentation below for details.



Filter: With the filter string, you can limit the set of objects displayed on the hierarchy tree using object names and the wildcard character '*'. For example, if you want to only see objects directly below the root level and whose name starts with the letters 'sp' (spheres, for example), set the filter string to '/root/sp*'. The filtering ability helps in searching objects from a complex model.

Mouse Operations

Dragging one or several selected objects with the left mouse button moves the objects in the hierarchy tree. Arrow icons indicate the new location already before releasing the mouse button. The following variations are available:

- *LMB dragging an item:* Moves an item to a new location in hierarchy.
- *Alt+LMB dragging an item:* Creates an instance of the selected object(s) and moves it to a new location in hierarchy.
- *Ctrl+LMB dragging an item:* Creates a copy of the selected object(s) and moves it to a new location in hierarchy.

Left mouse button dragging without hitting an object at the drag start point performs object selection.

- *LMB dragging:* All objects inside the drag box will become selected. Possible previous selection is lost.
- *Shift+LMB dragging:* All objects inside the drag box will be added to previous selection.
- *Ctrl+LMB dragging:* Boolean AND operation 'previous selection AND new selection'. All objects inside the drag box that were also previously selected will remain selected. All other objects become deselected.
- *Alt+LMB dragging:* Subtracts all objects inside the drag box from selection.
- *Shift+Ctrl+LMB dragging:* XOR operation. Inverts selection of objects inside the drag box. Selected objects become unselected and vice versa.



Note

Only objects inside the drag box change their selection state.

Left mouse button clicking activates objects. Clicking an empty area deselects all objects.

- *LMB click:* Selects the clicked object. Other objects become deselected.
- *Shift+LMB click:* Adds the clicked object to the previous selection.
- *Ctrl+LMB click:* Range selection. Adds all objects between the previous selected object (closest upwards in the hierarchy tree in the same hierarchy level with the clicked object) and the clicked object to the selection. If there are no other selected objects in the same hierarchy level, selection starts from the beginning of the level.

- *Alt+LMB click:* Deselects the clicked object.
- *Shift+Ctrl+LMB click:* Toggles (=inverts) selection of the clicked object.

Double clicking activates some additional tools:

- *Slow LMB double click:* Activates the 'Rename' tool.
- *Quick LMB double click:* Opens the object property window.
- *Shift + quick LMB double click:* Hides the hierarchy above a clicked level object. The level will appear as the root of the hierarchy. Another Shift double click will restore the original root.

The contents of the select window can be scrolled by dragging with the middle mouse button, or by the right mouse button dragging Ctrl key held down.

Hotkeys

Most hotkey events at the select window are forwarded to the active view window. For example, hitting the '0' key renders the active view. Only the 'tab' key has a special use: it sets the current level to the selected level. If the selected object is not a level object, nothing happens. The current level defines the hierarchical destination for created, loaded or pasted objects.

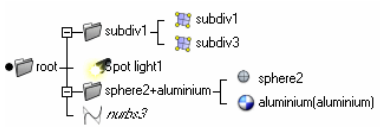
The Popup Menu

The popup menu of the object geometry tab is context sensitive. New menu items appear depending on object selection. For example, 'Rename' menu is not available if all objects are deselected. Another example: 'Open' menu appears only when one or more level objects are selected.

As usual, plugin products may insert additional items to the popup menu of the select window.

- **New** These sub items create some special purpose objects into the object hierarchy.
  - *3D Measure*
  - *Angle Measure*
  - *Measure*
  - *Position Mark*
  - *Level*
  - *Rotator*
  - *SunMoon*
  - *Fractal Tree*
  - *Post Image*
  - *Plant Seed*
  - *Choreography*
  - *Tube*
  - *Interpolator*
  - *Collision Deformer*
  - *Inverse Kinematics*
  - *Drag*
  - *Morphing*
  - *Metaball*
  - *Creator*
  - *Foot Print*
  - *Field Evaluator*
- **Open (Close)** The Open menu opens the selected level object so that its sub objects become visible. Sub objects can then be selected individually. The Close menu does the opposite i.e. hides the sub hierarchy.
- **Open All (Close All)** Opens the sub hierarchy recursively. Sub objects and their sub objects etc. become visible on the select window. Close all closes the sub hierarchy recursively.
- **Make Current** Makes the selected level-type object the current level. The 'current level' concept defines where new created, loaded or pasted objects should be placed in hierarchy.
- **Edit** Puts an object to the editing mode. Usually objects reveal some object specific handles in the editing mode, allowing the user to modify the characteristic geometric details of the object.
- **Drop to a Level** Creates a new level object and inserts all selected objects into it. The new level will be placed to the location in hierarchy where the first selected object was.

- **Set Filter** Copies the full hierarchy path from the selected object to the name filter string. The string can then be modified as appropriate. The filter tool allows the user to display only a suitable subset of the hierarchy using string pattern matching. This is a useful option when searching items from a complex hierarchy.
- **Cut** Cuts the selected objects from the object hierarchy to the Realsoft 3D geometry clipboard.
- **Copy** Copies the selected objects from the object hierarchy to the Realsoft 3D geometry clipboard. The selected objects stay in hierarchy.
- **Paste** Pastes geometric objects from the Realsoft 3D clipboard to the current level.
- **Duplicate** Creates duplicates of the selected objects. Each duplicate is placed to the same hierarchy level with its original object.
- **Delete** Deletes the selected objects. The clipboard will not change.
- **Copy to a File** Opens a file browser. Selected objects are then saved to a defined disk file.
- **Paste from a File** Opens a file browser. Objects from a selected disk file will be loaded to the current level.
- **Select Method**
  - *Don't select new objects* Created/loaded/pasted objects will not become automatically selected.
  - *Select newly created* Created/loaded/pasted objects will become automatically selected. Previous object selection is lost.
  - *Add to selection* Created/loaded/pasted objects will become automatically selected. Previously selected objects remain also selected.
- **Balanced Tree** Displays the hierarchy as a horizontally organized tree.
- **Rename** Activates a rename box for the selected object.
- **Properties..** Opens the object property window.



Balanced hierarchy mode

- **Make Choreographable** Makes an object animated. The time line and other animation properties can then be edited with the Choreography Window.



Note

Modifying an object property in the animation recording mode makes the object also choreographable.

- **Load Choreography** Loads animation properties from a disk file and attaches the loaded information to selected objects (only common attributes become utilized - a sphere radius animation will not attach to a cube).
- **Add New Choreography** Adds a new empty choreography to the selected objects. The new choreography will be the active one i.e. animation recording actions will be stored to it. Multiple choreographs can be combined in a weighted manner.
- **Delete Choreographs** Removes all animation information from the selected objects.

Drag and Drop

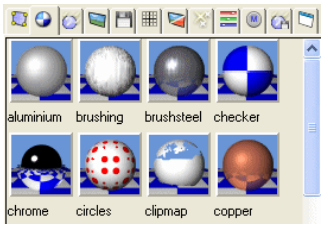
Dragging an object from the geometry tab of the select window to a view window performs an auto focus operation: the view is panned, rotated and zoomed so that the object fills most of the view window. The camera object is an exception: the view window fetches the camera view of the dropped object.

See also: Reference Manual/Models/Geometric Objects.

Materials

The second tab of the select window contains the current material library. Materials are objects which control object shading. The *Current material Library* means the set of materials loaded into the computer memory for use in the project under construction.

The program installation contains a set of predefined materials stored into the 'materials' folder and its sub folders. Part of these materials are loaded with the default startup file. When building a project, the user needs only a certain subset of materials, which are also often customized for the project needs. The current material library is a suitable place to store such project specific materials. When the users saves the project, the current material library is saved as a part of the project data, and the original material templates remain intact in the 'materials' folder.



A typical material usage workflow goes as:

- Start the program and model a scene.
- See if the default startup loaded library contains suitable materials and assign them to objects.
- Apply the 'Purge' feature to remove unused materials. This saves disk space when later saving the project.
- Load (or build) the rest of the required materials from disk files of the 'material's folder.
- Save the project

The Popup Menu

- **New** Tools for inserting a new material into the current library.
  - *From Template* Opens a file browser for loading a material that has been saved to a disk file.
  - *VSL Material, VSL Procedure Library, Level Material, V3 Material* Creates new empty materials to the current library.
- **Browse Material Library** Opens a special preview icon based window for loading disk based materials. Unlike the 'New/From Template' menu, this method allows you to load several materials at the same time. With this window, you can load a material from the disk file in one of the three following ways:
  - Double clicking a preview icon on the material browse window inserts it in the current library.
  - You can also drag and drop a material from the browse window into the select window.
  - Drag and dropping a material from the browse window to a view window loads the material and maps it to the selected geometric objects (just like drag and drop from the select window's material tab to a view window).

  Drag and dropping a material from the select window to the browse window saves the material to a disk file using material's name as the file name.
- **Cut** Cuts the selected material from the material library to the Realsoft 3D material clipboard.
- **Copy** Copies the selected material to the Realsoft 3D material clipboard. The selected material stays in the library.
- **Paste** Pastes the material from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected material. The duplicate is placed to the end of the material library.
- **Delete** Deletes the selected material. The clipboard will not change.

- **Delete All** Deletes all materials. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every material has an unique name.
- **Purge** Deletes all materials that have not been assigned to objects of the scene.
- **Copy to a File** Opens a file browser. The selected material is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The materials from a selected disk file will be loaded to the current library (this is the same as New/From Template - the menu is here for consistency).
- **Sort Alphabetically** Rearranges the materials into alphabetical order.
- **Map** The sub items of this menu create material mapping objects. A mapping object attaches the selected material to selected geometric objects using a suitable geometric 'wrapping' transformation. After the tool activation, most mapping types require some mouse clicks which define the mapping geometry. See also Mapping Objects documentation.
- **Rename** Activates a rename box for the selected material.
- **Show Previews** If set, material preview icons are shown on the select window. If cleared, only the names are shown.
- **Properties..** Opens the object property window.
- **Make Choreographable** Makes an material object animated. The time line and other animation properties can then be edited with the Choreography Window.



Note

Modifying a material property in the animation recording mode makes the material also choreographable.

- **Load Choreography** Loads animation properties from a disk file and attaches the loaded information to the selected material.
- **Add New Choreography** Adds a new empty choreography to the selected material. The new choreography will be the active one i.e. animation recording actions will be stored to it. Multiple choreographs can be combined in a weighted manner.
- **Delete Choreographs** Removes all animation information from the selected material.

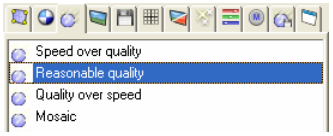
Drag and Drop

Dragging a material from the select window to a view window auto maps the material to the selected geometric objects. The recommended mapping type is used for each selected object, as specified in the material properties. If no objects are selected, a mapping is inserted to the current geometry level. If a level object is selected, the first sub object is used for mapping recommendation calculations.

See also: Reference Manual/Models/Material Objects.

Render Settings

A render settings object includes options for photorealistic rendering. The render settings library is common to all projects: if you have multiple projects open, they all share the same collection.



The Popup Menu

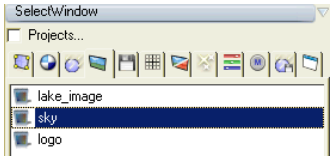
- **New** Creates a new render settings object, which can be customized as appropriate.
- **Cut** Cuts the selected settings object to the Realsoft 3D render settings clipboard.
- **Copy** Copies the selected settings object to the Realsoft 3D render settings clipboard. The selected object stays in the library.
- **Paste** Pastes the settings object from the Realsoft 3D clipboard to the render settings library.
- **Duplicate** Creates a duplicate of the selected settings object. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected object. The clipboard will not change.
- **Delete All** Deletes all objects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every setting has an unique name.
- **Purge** Deletes all unused setting objects (not used by view windows or file render objects).
- **Copy to a File** Opens a file browser. The selected render settings object is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The object from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected render settings object.
- **Properties..** Opens the object property window.

Drag and Drop

Dragging a render settings object from the select window to a view window selects the render settings for the view.

See also: Reference Manual/Models/Rendering Settings.

Images



The image library is projects specific: if you have multiple projects open, they all have their own set of images.

The Popup Menu

- **New** Creates a new empty image object.
- **Cut** Cuts the selected image to the Realsoft 3D image clipboard.
- **Copy** Copies the selected image object to the Realsoft 3D image clipboard. The selected image stays in the library.
- **Paste** Pastes an image object from the Realsoft 3D clipboard to the current image library.
- **Duplicate** Creates a duplicate of the selected image object. The duplicate is placed to the end of the image library.
- **Delete** Deletes the selected image object. The clipboard will not change.
- **Delete All** Deletes all objects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every image has an unique name.
- **Purge** Deletes all unused image objects (which are not used, for example, as view backdrops).

- **Copy to a File** Opens a file browser. The selected image object is then saved to a defined disk file.



Note

This does not save the actual RGB data to the defined file, only the 'wrapper' object!

- **Paste from a File** Opens a file browser. The object from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected image object.
- **Properties..** Opens the object property window.
- **Make Choreographable** Makes an image object animated. The time line and other animation properties can then be edited with the Choreography Window.



Note

Modifying a property in the animation recording mode makes the object also choreographable.

- **Load Choreography** Loads animation properties from a disk file and attaches the loaded information to selected image.
- **Add New Choreography** Adds a new empty choreography to the selected object. The new choreography will be the active one i.e. animation recording actions will be stored to it. Multiple choreographs can be combined in a weighted manner.
- **Delete Choreographs** Removes all animation information from the selected image.

Drag and Drop

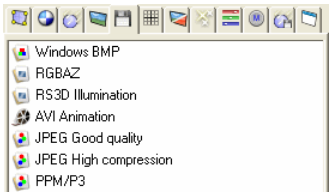
Dragging an image object from the select window to a view window sets the view backdrop image.

See also: Reference Manual/Models/Image Objects.

Output Objects

An output object defines a file format or another output target with customizable channel bindings. The output object library is a shared one: if you have multiple projects open, they use the same library.

The Popup Menu



- **New** Creates a new output object. The sub menu items include the installed output object types (supported image file formats).
- **Cut** Cuts the selected output object to the Realsoft 3D output object clipboard.
- **Copy** Copies the selected output object to the Realsoft 3D output object clipboard. The selected object stays in the library.
- **Paste** Pastes the output object from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected output object. The duplicate is placed to the end of the output object library.
- **Delete** Deletes the selected output object. The clipboard will not change.

- **Delete All** Deletes all objects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every output object has an unique name.
- **Purge** Deletes all output objects, which are not used by file render configurations etc.
- **Copy to a File** Opens a file browser. The selected output object is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The output objects from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected output object.
- **Properties..** Opens the object property window.

Drag and Drop

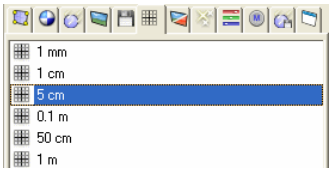
You can add output objects for file rendering by dragging and dropping them from the select window to the active output list of the file rendering window.

See also: Reference Manual/Models/Output Objects.

Grids

Grid objects round convert mouse coordinates to a defined base interval division. The grid library is project specific: if you have multiple projects open, each has its own set of grids.

The Popup Menu



- **New** Creates a new grid object to the grid library.
- **Cut** Cuts the selected grid to the Realsoft 3D grid clipboard.
- **Copy** Copies the selected grid to the Realsoft 3D grid clipboard. The selected grid stays in the library.
- **Paste** Pastes the grid object from the Realsoft 3D clipboard to the current grid library.
- **Duplicate** Creates a duplicate of the selected grid. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected grid. The clipboard will not change.
- **Delete All** Deletes all objects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every grid has an unique name.
- **Purge** Deletes all unused grids.
- **Copy to a File** Opens a file browser. The selected grid is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The grid object from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected grid object.
- **Properties..** Opens the object property window.

Drag And Drop

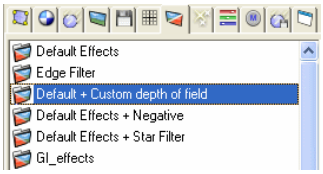
Dragging a grid object to a view window selects the current grid for the view and activates grid drawing and snapping.

See also: Reference Manual/Models/Grid Objects.

Post Image Effects

Post image effects manipulate the pixel information from the ray tracer - they are 'raster' effects. The post image library is project specific: if you have multiple projects open, each uses its own private effects.

The Popup Menu



- **New** Creates a new post image effect. Sub menu items show the all installed image effects.
- **Cut** Cuts the selected image effect to the Realsoft 3D image effect clipboard.
- **Copy** Copies the selected image effect to the Realsoft 3D image effect clipboard. The selected effect stays in the library.
- **Paste** Pastes the image effect from the Realsoft 3D clipboard to the current effect library.
- **Duplicate** Creates a duplicate of the selected image effect. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected effect. The clipboard will not change.
- **Delete All** Deletes all effects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every effect has an unique name.
- **Purge** Deletes all unused image effects.
- **Copy to a File** Opens a file browser. The selected image effect is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The image effect objects from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected image effect.
- **Properties..** Opens the object property window.
- **Make Choreographable** Makes an object animated. The time line and other animation properties can then be edited with the Choreography Window.



Note

Modifying a property in the animation recording mode makes the object also choreographable.

- **Load Choreography** Loads animation properties from a disk file and attaches the loaded information to selected image effect.
- **Add New Choreography** Adds a new empty choreography to the selected object. The new choreography will be the active one i.e. animation recording actions will be stored to it. Multiple choreographs can be combined in a weighted manner.

- **Delete Choreographs** Removes all animation information from the selected image effect.

Drag And Drop

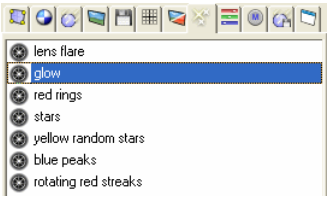
Dragging a post image object to a view window selects the post effect configuration per render box for the view. Dragging post effects to a view inserts them into the post effect configuration used by the view per render box.

See also: Reference Manual/Models/Post Image Effects.

Post Particle Effects

Post particle effects define how particles is rendered in post processing phase. In other words, you can assign different appearances to particles using post particle effects. The post particle effect library is project specific: if you have multiple projects open, each project has its own set of effects.

The Popup Menu



- **New** Creates a new particle effect. Sub menu items show the list of installed particle effects.
- **Cut** Cuts the selected particle effect to the Realsoft 3D particle effect clipboard.
- **Copy** Copies the selected particle effect to the Realsoft 3D particle effect clipboard. The selected particle effect stays in the library.
- **Paste** Pastes a particle effect from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected particle effect. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected particle effect. The clipboard will not change.
- **Delete All** Deletes all particle effects. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every particle effect has an unique name.
- **Purge** Deletes all unused particle effects.
- **Copy to a File** Opens a file browser. The selected particle effect is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The particle effect from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected particle effect.
- **Properties..** Opens the object property window.
- **Make Choreographable** Makes a particle effect animated. The time line and other animation properties can then be edited with the Choreography Window.



Note

Modifying a particle effect property in the animation recording mode makes the particle effect also choreographable.

- **Load Choreography** Loads animation properties from a disk file and attaches the loaded information to selected particle effect.

- **Add New Choreography** Adds a new empty choreography to the selected particle effect. The new choreography will be the active one i.e. animation recording actions will be stored to it. Multiple choreographs can be combined in a weighted manner.
- **Delete Choreographs** Removes all animation information from the selected particle effect.

Drag And Drop

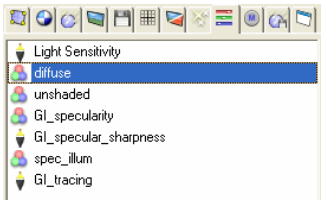
Dragging a post particle effect to a view window auto maps the effect to the selected geometric objects. This normally makes sense only when the selected objects are particles.

See also: Reference Manual/Models/Particle Effects.

Channels

Channels are user definable rendering attributes. The channel library is shared between all projects. For example, the material preview project automatically finds all new channels of a custom material shader.

The Popup Menu



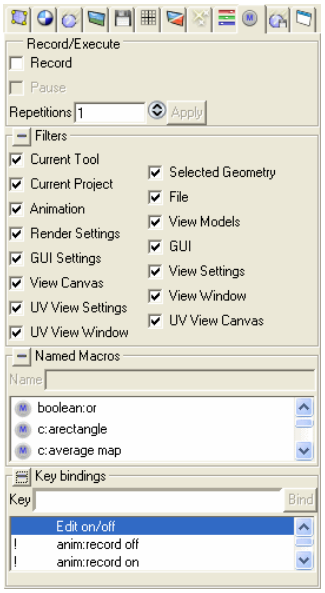
- **New** This menu has three sub items for creating new float value, color and vector channels.
- **Cut** Cuts the selected channel to the Realsoft 3D channel clipboard.
- **Copy** Copies the selected channel to the Realsoft 3D channel clipboard. The selected channel stays in the library.
- **Paste** Pastes a channel from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected channel. The duplicate is placed to the end of the channel library.
- **Delete** Deletes the selected channel. The clipboard will not change.
- **Delete All** Deletes all channels. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every channel has an unique name.
- **Purge** Deletes all unused channels.
- **Copy to a File** Opens a file browser. The selected channel is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The channels from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected channel.
- **Properties..** Opens the object property window.

See also: Reference Manual/Models/Channels.

Macros

Macros store user defined action sequences. Recording several frequently needed steps into a macro often improves working efficiency. Macros are shared between projects: if you load a project which includes a macro section, all opened projects use the new macro set.

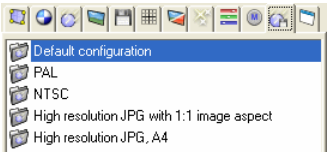
The Controls of the Macro Tab



- **Record** Turns macro recording on/off.
- **Pause** Turns macro recording temporarily off. You can then perform operations that do not record into the macro and continue recording again later.
- **Repetitions** Defines how many times the current macro is executed when the Apply button is pressed.
- **Apply** Executes the recorded macro ('Repetitions' times).
- **Filters** Filter options define what kind of actions get recorded into macro. For example, if 'View Canvas' option is disabled, activating the OpenGL fog option does not record into the macro.
- **Name** You can rename the recorded macro with this gadget. If you want to store a recorded macro with your working environment, you have to rename it.
- **Macro List** The list of all named macros. The macro collection is loaded from the 'Macros' section of a project file. You can add a new macro to the list by recording one and renaming it appropriately. You can execute a macro by dragging it to a view window.
- **Key** A field for defining a hotkey combination that executes the selected macro:
  - Record a macro and rename it to a unique name, or select an existing named macro from the list.
  - Select the geometric object which should activate a context sensitive macro. If the macro is not object specific, deselect all geometric objects.
  - Activate the 'Key' field and press the key combination you want to use for the macro.
  - Make sure that the appropriate macro is selected on the macro list above and press the 'Bind' button.
  - To use the new key binding at a later date, save the project with 'Macros' section.
- **Bind** Creates a new hot key binding between the 'Key' field contents and the currently selected named macro.
- **Key Binding List** The list of all macro key bindings. Use the popup menu to delete undesired bindings.

File Rendering Configurations

A file rendering configuration combines render settings, post processing configurations, output resolution and output file formats together into a ready-to-use package. For example, you can define a PAL video rendering configuration. The file rendering configuration library is shared between all projects: multiple opened projects use the same collection.



The Popup Menu

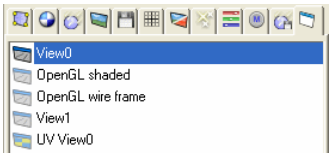
- **New** Creates a new rendering configuration with some default settings.
- **Cut** Cuts the selected rendering configuration to the Realsoft 3D file rendering clipboard.
- **Copy** Copies the selected rendering configuration to the Realsoft 3D clipboard. The selected rendering configuration stays in the library.

- **Paste** Pastes a rendering configuration from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected rendering configuration. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected rendering configuration. The clipboard will not change.
- **Delete All** Deletes all configurations. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every configuration has an unique name.
- **Purge** Deletes all unused configuration objects.
- **Copy to a File** Opens a file browser. The selected rendering configuration is then saved to a defined disk file.
- **Paste from a File** Opens a file browser. The rendering configurations from a selected disk file will be loaded to the current library.
- **Rename** Activates a rename box for the selected rendering configuration.
- **Properties..** Opens the object property window.

See also: Reference Manual/Models/File Rendering Configurations.

View Models

A view model stores settings for a view window. Therefore, just like the GUI, the view model collection is shared between opened projects. When you swap from a project to another one, the GUI and the view model library it uses remain the same. When you load a new project with a customized GUI, all opened projects will use the new GUI and new view models.



The Popup Menu

- **New** Creates a new view model object.
- **Cut** Cuts the selected view model to the Realsoft 3D view model clipboard.
- **Copy** Copies the selected view model to the Realsoft 3D view model clipboard. The selected view model stays in the library.
- **Paste** Pastes a view model from the Realsoft 3D clipboard to the current library.
- **Duplicate** Creates a duplicate of the selected view model. The duplicate is placed to the end of the library.
- **Delete** Deletes the selected view model. The clipboard will not change.
- **Delete All** Deletes all view models. The program displays a warning before deletion.
- **Delete Duplicates** Removes objects which duplicate a name of an earlier object. After the operation, every view model has an unique name.
- **Purge** Deletes all unused view models.
- **Copy to a File** Opens a file browser. The selected view model is then saved to a defined disk file.

- **Paste from a File** Opens a file browser. The view models from a selected disk file will be loaded to the current view model library.
- **Rename** Activates a rename box for the selected view model.
- **Properties..** Opens the object property window.

Drag And Drop

Dragging a view model to a view window assigns the view model for the view. If the view model is already in use, a duplicate is created first (view models cannot be shared).

See also: Reference Manual/Models/View Models.

Status Window

The Status Window is available as a docked horizontal bar at the bottom of the default user interface. The window can be hidden/made visible using the View pull down menu, or with the hotkey 'F9'.



The main purpose of the status window is to give various instructions to the user. When the user starts a tool, the window instructs how to perform the basic steps of the tool. However, because of space limitations, such information is shown in a compact form.

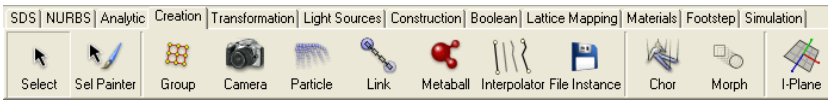
The window also shows a progress indicator during time consuming operations. For example, when ray trace rendering a view window, the progress indicator starts showing rendering time statistics. The Cancel button is also enabled and pressing it cancels the rendering.

Tool Bar

The Tool Bar Window is docked to the top of the default user interface. It contains a set of tabs, providing access to most of the tools available in the program. The contents of the toolbar is user configurable.

Toolbar, just like other program windows, can be scrolled by dragging it with the right mouse button 'Ctrl' key held down. Alternatively, the middle mouse button can be used for dragging without any key modifiers. The tool bar can be hidden and made visible from the pull down menu View/Tabbed tool bar or with the hotkey 'F3'.

Usually the toolbar works in conjunction with the tool control bar. When a tool is activated, the tool control bar shows the tool specific options.

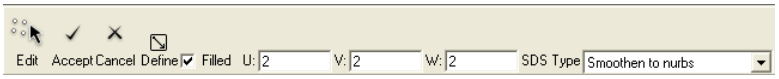


Tool Control Bar

The Tool Control Bar is docked below the toolbar. It is context sensitive: when a tool is activated, the tool control bar shows the tool specific options. Or, when an object is selected, the tool control bar shows tools which are specific to that particular object type.

If the contents of the control bar do not fit inside the visible area, you can scroll it by dragging it with the right mouse button 'Ctrl' key held down, or with the middle mouse button.

The tool control bar can be hidden and made visible from the pull down menu View/Control bar or with the hotkey 'F4'.



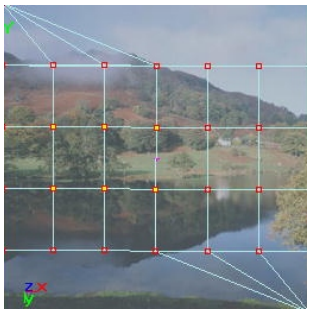
The tool control bar showing the context of the NURBS sphere tool

UV Editor

The UV editing window opens from the 'Windows/UV editor' pull down menu. F11 is the hotkey for opening the window.

With this window, you can edit:

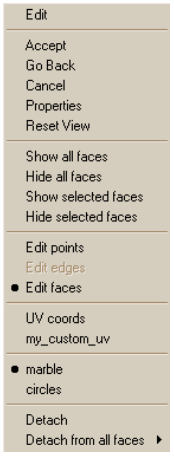
- UV coordinates of NURBS meshes
- UV coordinates and other vertex variables of SDS objects
- UV coordinates of UV Set objects
- Face UV's of SDS objects
- UV coordinates of trim curves



Editing UV coordinates of a SDS sphere

The popup menu of the UV editor is context sensitive. Below is a picture of the menu when editing a subdivision surface which has two vertex UV channels and two face mapped materials. The purpose of these menus are described below.

- **Edit:** Turns edit mode on/off. Edit mode must be enabled before UV editing can take place.
- **Accept, Go Back, Cancel:** The standard menus for interaction control.
- **Properties:** Opens the UV editor property window where various editor settings can be changed.
- **Reset View:** Resets the windows scale. The window will show exactly the unit square of the UV plane.
- **Show all faces:** Makes all faces being edited visible. This includes all SDS faces if you are editing vertex UV's. In face UV editing, all faces that have the currently selected material mapped on them become visible. With this and the three following menus, you can reduce excessive UV mesh complexity so that a suitable subset of data is left visible for editing.



- **Hide all faces:** Makes all faces being edited invisible. After applying this menu, you can make a suitable subset of faces visible with the 'Show selected faces' menu.



Note

You can perform the selection in a normal 3D view. Hiding faces in the UV editor does not make them invisible in a 3D view.

- **Show selected faces:** Makes all faces that are currently selected visible.
- **Hide selected faces:** Makes all faces that are currently selected invisible.
- **Edit Points:** Switches the UV editor to edit point handles.
- **Edit Edges:** Selects editing by using edge handles.

- **Edit Faces:** Editing with face handles.
- **UV coords and my_custom_uv:** This menu group shows all UV type vertex variables which are attached to the object. You can change the editing target to one of the vertex variables using these menu items.
- **circles and marble:** This menu group lists all materials that are face mapped to the selected SDS object. You can select from the menu which face material UV's to edit.
- **Detach:** Removes the material indicated by the menu group above (marble in this example) from the selected SDS faces.
- **Detach All:** This menu lists all attached face materials. You can remove the desired material attachment from all faces (regardless of the face selection).

Navigation

The following navigation operations are available:

- Ctrl + RMB dragging (or middle mouse button dragging) pans the UV editor
- Shift + RMB dragging zooms the UV editor in and out

UV Coordinate Types

NURBS surfaces and trim curves have only one UV channel available. Actually a NURBS surface may not have any UV channel at all, but you can attach and initialize the channel using the pull down menu 'Methods/NURBS mesh/ Define UV'. When editing this kind of objects, there is no need to select what to edit in the UV editor.

The situation is more complex when editing a SDS object. This object type not only allows several UV channels mapped to vertices, but also supports face mapped materials which have their own parametrization. Vertex UV channels store one UV value per vertex so that adjacent faces share the same UVs. Shared values make this parametrization type automatically continuous over face boundaries.

A face mapped material stores a private UV value for each vertex per face per material. The advantage is that faces can be torn apart in the UV space. A typical disadvantage is that face mapped materials do not run continuously over face boundaries.

UV Editor Backdrop

- The object has a direct UV texture (defined in Property window/Col tab/Texture field). The backdrop is drawn using a maximal accuracy, limitation being the resolution of the UV editor or the texture image itself.
- When editing of a face mapped material of a SDS object is started, the program automatically computes a backdrop image. The backdrop accuracy is defined by the Property window/Wire tab/Texture Quality field.
- You can also drag and drop any image from the select window's image tab onto the UV editor.



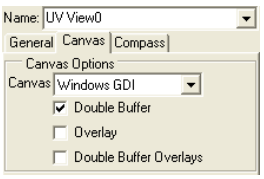
Note

Make sure that you don't have a SDS object selected an in edit mode, because you can assign images to selected faces the same way, too. Alternatively, the backdrop image can be created on the select window and activated using a UV view property window.

The contrast of the backdrop can be controlled from the UV view property window. Usually it is necessary to fade the backdrop suitably so that the UV mesh wireframe and handles become better visible over the background.

UV Editor Properties

You can edit the settings of the UV editing window by selecting its UV view model from the select window and by opening the standard property window. The property window can also be opened from the popup menu of the uv view.

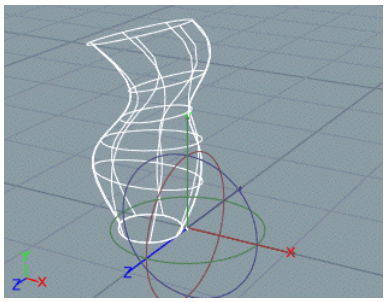


The controls of the UV editor property window are a subset of the controls of the View property window.

View

The view window is probably the most important element of the user interface. It shows you the geometric structure of the scene and projects mouse coordinates into the 3d space. The default user interface contains one large, docked view window, but you can open as many view windows as you like. An additional floating view window opens from the 'Windows/View Window' pull down menu; alternative user interfaces such as the quadview contain multiple view windows which show the scene simultaneously from several directions.

The popup menu of the view window is partly context sensitive. The topmost menu items are related to the program tools and therefore the menu set depends on the current stage of the possibly ongoing tool interaction.



The menu structure is:

- *Accept:* Finishes the current interaction
- *Go Back:* Undoes the last step of current interaction
- *Cancel:* Cancels the current interaction
- *Again:* Restarts previous interaction
- *Render:*
  - *Wire:* Refreshes the view window using the selected real time drawing system
  - *Ray Trace:* Activates photorealistic ray trace rendering into the view window
  - *Backup Ray Tracing:* Enables/disables a backup buffer for ray trace rendering. The result of rendering is stored into a disk file, which includes all channels needed by the selected post effects. The deep image can be viewed later and post effects can be repeatedly computed using it as the source data.
  - *Post Process:* Renders only the post effects. If a backup buffer is available, the post processing system uses it instead of an image coming from the ray tracer.
  - *Post Process File:* Applies the view's post effect configuration to an image file on disk. Result is shown on the view window. The image file is not scaled: if the source image does not fit the view window, additional parts become clipped.
  - *Save 2D Projection:* Saves 2D projected wireframes of the scene into a disk file. Any of the available export formats can be used, for example dxf or postscript. The created file can be edited further in a CAD program or printed. The view window does not need to be in a wireframe mode - the projection can be saved from a shaded view as well. Wireframe outside the visible view is clipped. The output scale is always such that the view width corresponds 1 native coordinate units of the export format. For example, if a view is configured to show a 100 meter wide model (view scale = 0.01 in the parallel projection mode), the scale ratio of the outputted data is 1:100.
  - *Define Render Box:* Activates render box interactor. The user can LMB drag a rectangle whose contents will be ray traced. All later ray trace renderings will be limited to the defined are until the render box is deactivated.

- *Box Active:* Activates/deactivate the render box
- *Save to File:* Saves the contents of the current render box (or the whole view if render box is unused) to a given disk file.
- *Animation Preview:* Plays the animation and saves view window contents to a given image/animation file. after each frame. The format can be selected from the view property window (.avi by default). A prerendered avi file can then be played at constant speed regardless of the scene complexity. You can stop the preview rendering using the play button of the animation window.
- *Cancel Rendering:* Stops the current rendering operation.
- *Color, ...:* The channels of the backup buffer are listed at the end of the Render menu. Picking a channel item from the menu displays the channel contents on the view.

- *Handles:* The top level switches for the general object handles.
  - *Bounding Box:* Enables drawing of a bounding box handle around the selected object. You can scale the selected objects by LMB-dragging the corners of the bounding box and move them by dragging the bounding box edges. The bounding box handle works also as a useful selection indicator in shaded OpenGL, because it is drawn outside (possibly obscuring) surfaces.
  - *Rotate:* Enables drawing of the object space rotate, translate and scale handles. These handles are drawn according to the selected object. If multiple objects are selected, the handles are drawn for the first object only. The three circular handles rotate the object. The three coordinate axis lines move the object in one dimension at a time. The end knobs of axis lines scale the object in the respective directions. The origin point of the handle system is the pivot point translation handle. All these handles can be used either by LMB-dragging them, which is a quick but inaccurate way. The handles can also be activated by a LMB click and then used with 2 clicks more. This method is more accurate because drag-snapping can be used to obtain exact results.

- *Camera:* This menu contains sub menus for controlling the view camera.
  - *Front:* Rotates the view camera about the aimpoint to the front view (-Z axis)
  - *Back:* Rotates the view camera about the aimpoint to the back view (+Z axis)
  - *Left:* Rotates the view camera about the aimpoint to the left side view (+X axis)
  - *Right:* Rotates the view camera about the aimpoint to the right side view (-X axis)
  - *Top:* Rotates the view camera about the aimpoint to the top view (-Y axis)
  - *Bottom:* Rotates the view camera about the aimpoint to the bottom view (+Y axis)
  - *Object View:* Copies the orientation (and other settings if included) from the selected object to the view camera
  - *View to Object:* Stores the orientation (and other settings if the target is a camera) to the selected objects.
  - *Perspective:* Toggles perspective and parallel camera projections
  - *IPlane View:* Rotates the view camera so that it points orthogonally against the input plane
  - *Auto Focus:* Moves, rotates and zooms the camera so that the selected objects fit into the view window suitably. If Shift key is pressed and objects are in edit mode, the focus is set to selected editing handles.
  - *Clip Focus:* Sets the near and far clipping distances to match the selected objects
  - *Zoom In/Out:* Activates an interactor that LMB drags a zoom rectangle on a view window. Left-to-right drag zooms the view window inside the defined rectangle, right to left shrinks the current view to fit into the rectangle.

    - *Tracking:* Switches camera object tracking on/off. More detailed control is available at the view property window.
- *Input Plane:*
    - *Front:* Rotates the input plane parallel to the XY plane about the grid origin
    - *Side:* Rotates the input plane parallel to the YZ plane about the grid origin
    - *Top:* Rotates the input plane parallel to the XZ plane about the grid origin
    - *Set Origin:* Activates an interactor that reads a LMB click (or drag-snap) from a view and moves the grid origin to the defined point.
    - *Match Surface:* Activates an interactor that reads a LMB click (or LMB-drag snap point) from a view and moves the grid origin to the surface point under the defined point. The input plane is rotated to fit the tangent plane of the underlying surface.
    - *Match 3 Points:* Reads three points from a view window. The points define the new input plane. The first point defines the new grid origin. The line between the two first points defines the new grid x axis.
    - *Lock to Camera:* Locks the input plane to view camera. The input plane is automatically kept orthogonal to the view camera direction.
    - *Grid Snap:* Turns grid snapping on/off.
    - *Grid Draw:* Turns grid drawing on/off
- *View Orientation:*
    - *Reset:* Initializes all view settings to the default front view (looking against z axis). The native view is reset back to the front view as well.
    - *Goto Native:* Sets the view window into the native state recorded into it.
    - *Current to Native:* Records the current view orientation and scale settings as the native state of the view.
    - *Object to Native:* Fetches orientation settings from a (camera) object and stores them as the native state.
- *World orientation:*
    - *From Object:* Fetches world coordinate orientation from the selected object. All numeric values will be measured along the object space directions.
    - *From Input Plane:* Fetches the world coordinate orientation from the current input plane orientation. Prior to using this tool, the input plane can be aligned to a surface tangent, an object space etc.
    - *Set Origin:* Reads a mouse click (or snap point) and sets the origin of world coordinates there.
    - *Reset:* Removes all world coordinate transformations.
- *Enter:*
    - *Compass Down Point:* Generates a left mouse button down event, which can be recorded to a macro. At macro execution, the click event will use the current mouse coordinates (not recording time coordinates).
    - *Compass Up Point:* Generates a left mouse button up event for macro recording.
- *Object Properties...:* Opens the property window
- *View Properties...:* Opens the view property window. With it, you can edit view window settings.

The Compass Menus

The idea of the compass menu is to provide a quick way to activate tools using the mouse. Use of normal pull-down and popup menus always requires careful aiming when an item should be picked. Compass menus are based on eight directions and two selection distances. These can be learned so that the hand remembers the motion, making the selection very quick.

The compass menu opens if you drag on a view window with the mouse, the right mouse button held down. The compass does not activate if a key modifier which selects another operation is held down - for example Shift-RMB-dragging just rotates the view.

The compass menu is context sensitive. The contents of it depends on key modifiers, object selection, object's edit state etc. The default view compass, shown above, opens if none of the selected objects specifies another kind of compass. You can also force the view compass by pressing the 'v' (=view) modifier key down before RMB-dragging.

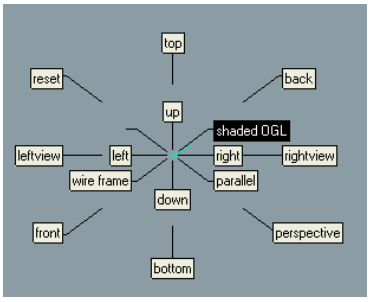
Below, some general purpose compasses are described. Many object types and tools activate their own compass. Please check the object specific documentation for details.

Note: Modifier keys 'x' and 'z' are unused. They can be used in activation of user defined compasses. Most other keys are already configured as hotkeys that activate a tool, and their use as a compass modifier may lead to conflicting operations (compass activation performs another operation as a side effect).

Navigation Compass

Description: Tools for changing the view orientation, the drawing mode and the projection

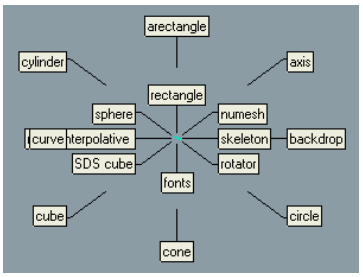
Activation: No compass modifiers active or 'v' key pressed



Items:

- *up:* Rotates view camera pitching angle upwards to the nearest 90 degree step
- *top:* Rotates the camera to the top view (direction against y axis)
- *shaded OGL:* Tuns OpenGL accelerated drawing on
- *back:* Rotates the camera to the back view (direction along z axis)
- *right:* Rotates camera heading rightwards to the nearest 90 degree step
- *rightview:* Rotates the camera to the right side view (direction against x axis)
- *parallel:* Activates parallel view projection (view camera has infinitely large focal length)
- *perspective:* Activates perspective view projection
- *down:* Rotates the view camera pitching angle downwards to the nearest 90 degree step
- *bottom:* Rotates the camera to the bottom view (direction along y axis)
- *wire frame:* Selects wire frame drawing without OpenGL acceleration
- *front:* Rotates the camera to the front view (direction against z axis)
- *left:* Rotates the camera heading to the left, to the nearest 90 degree step
- *leftview:* Rotates the camera to the left side view (direction along x axis)
- *reset:* Resets the view back to its 'native' state

Creation Compass



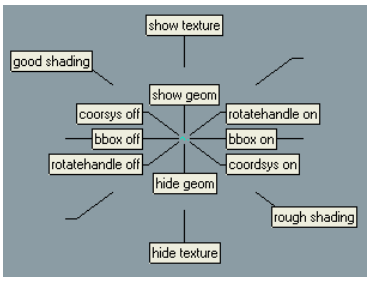
Description: Frequently needed object geometry creation tools.

Activation: 'q' key pressed

Items:

- *rectangle:* A rectangular 4*4 cubic NURBS mesh
- *arectangle:* An analytic rectangle
- *numesh:* An 8*8 cubic NURBS mesh
- *axis:* A vertical line of 2 points (suitable as a rotation axis etc.)
- *skeleton:* Creates a skeleton object
- *backdrop:* Creates a camera backdrop plane. If a camera object is not selected, creates also a camera from the current view.
- *rotator:* Creates a rotator object.
- *circle:* A circular NURBS mesh
- *fonts:* Activates the font tool
- *cone:* An analytical cone
- *SDS cube:* A 2*2*2 subdivision surface cube
- *cube:* An analytic cube
- *NURBS interpolative:* Cubic NURBS curve creation through curve knot points
- *curve:* Cubic NURBS curve creation by control polygon points
- *sphere:* An analytic sphere
- *cylinder:* An analytic cylinder

Wireframe Compass



Description: This compass controls various real time drawing options of geometric objects.

Activation: 'w' key pressed

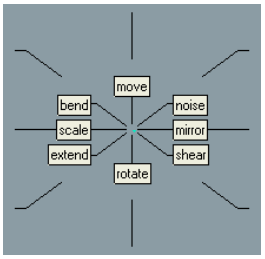
Items:

- *show geom:* Makes hidden geometry visible
- *show texture:* Sets Property window/Wire tab/Texture quality of selected objects to 64*64 pixels.
- *rotatehandle on:* Makes object space rotate handle of selected objects visible. Note: View popup manu/Handles/Rotate must be active, otherwise the handles are not visible.
- *bbox on:* Makes object space bounding box handle of selected objects visible. Note: View popup manu/Handles/Bounding Box must be active, otherwise the handles are not visible.

- *coordsys on:* Makes object space axis handle (for translation and scaling) of selected objects visible. Note: View popup manu/Handles/Rotate must be active, otherwise the handles are not visible.
- *rough shading:* Sets Property window/Wire tab/Shaded quality of selected objects to the level 2.
- *hide geom:* Hides surfaces of selected objecs. Possible sub objets or handles are not affected.
- *hide texture:* Sets Property window/Wire tab/Texture quality to 'None'
- *rotatehandle off:* Makes object space rotate handle of selected objects invisible.
- *bbox off:* Makes object space bounding box handle of selected objects invisible.
- *coordsys off:* Makes object space axis handle of selected objects invisible.
- *good shading:* Sets Property window/Wire tab/Shaded quality of selected objects to the level 4.

Transformation Compass

Description: Frequently needed transformation tools.



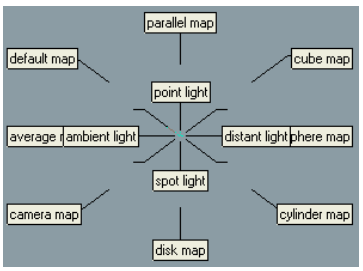
Activation: 'e' key pressed

Items:

- *move:* Activates the usual object translation tool.
- *noise:* Activates a transformation that moves points of selected objects randomly.
- *mirror:* Activates the tool which reflects the selected objects w.r.t. a given axis
- *shear:* Activates the shear (skew) tool
- *rotate:* Activates the rotate tool
- *extend:* Activates one-dimensional scaling along a given axis
- *scale:* Activates the scale tool which allows rectangular scaling. Uniform Scaling can be obtained by pressing the Shift modifier down and moving the mouse diagonally.
- *bend:* Activates the tool for bending along a circle arc

Light Source and Material Mapping Compass

Description: Tools for creating light sources and material mapping objects.



Activation: 'y' key pressed

Items: Light sources at the near distance, mapping objects at the far distance.

- *point light:* Activates creation of the usual point light source
- *parallel map:* Activates creation of a parallel material mapping object
- *cube map:* Activates creation of a cube material mapping object
- *distant light:* Activates creation of a distant light source
- *sphere map:* Activates creation of a spherical material mapping object

- *cylinder map:* Activates creation of a cylindrical material mapping object
- *spot light:* Activates creation of a spot light
- *disk map:* Activates creation of a disk material mapping object
- *camera map:* Activates creation of a default material mapping object with the 'camera' coordinate option.
- *ambient light:* Adds an ambient light source into the scene
- *average map:* Creates a level object that computes a symmetric average of selected material mapping objects.
- *default map:* Activates creation of a default (no geometry included) material mapping object.

Navigation

You can navigate (move the view's camera in space) in several ways:

- Using the mouse with some modifier keys
- With keyboard shortcuts
- The view control window contains a set of navigation controls
- The compass menu includes navigation functions
- The popup menu provides a comprehensive set of frequently needed predefined orientations (front, side, top view etc.)' and other similar controls
- With the numerical controls of the view property window

View navigation by the right mouse button:

Modifier	**Function**
	Opens the compass menu
Ctrl	Pan (move camera)
Alt	Rotate camera about aimpoint
Shift Ctrl	Move camera forward/backward
Shift Alt	Rotate in 45 degree steps
Ctrl Alt	Adjust camera banking
Shift Ctrl Alt	Adjust camera banking in 45 degree steps

View navigation by the middle mouse button:

Modifier	**Function**
	Pan (move camera)
Ctrl	Pan (move camera)
Alt	Rotate camera about viewpoint
Shift Alt	Rotate camera about viewpoint in 45 degree steps

View navigation by the mouse wheel:

Modifier	**Function**
	Zoom about the current mouse position

Modifier	Function
Shift	Zoom about the current viewpoint

View cruising with cursor keys:

Tab	View camera cruising on
Tab+Shift	Turbo boosted cruising
Tab+Up	Accelerate
Tab+Down	Decelerate
Tab+Left	Turn left
Tab+Right	Turn right
Tab+Alt+Up	Turn up
Tab+Alt+Down	Turn down
Tab+Alt+Left	Bank counterclockwise
Tab+Alt+Right	Bank clockwise
Tab+Ctrl+Up	Move up
Tab+Ctrl+Down	Move down
Tab+Ctrl+Left	Move left
Tab+Ctrl+Right	Move right
Tab+Left+Right	Turn towards horizontal flight

View Camera

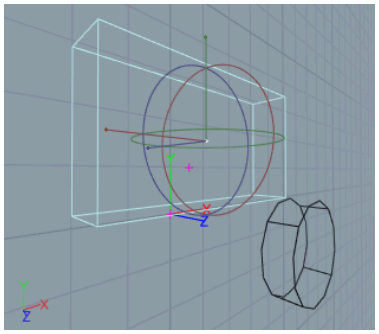
Every view window owns a private 'camera'. A view camera stores attributes which are familiar from real world camera: position, aiming direction, focal length etc. Thanks to this view camera concept, you can 'take pictures' i.e. render images on a view window without creating an actual camera object into the scene hierarchy. A view camera is identical with the camera objects in the scene hierarchy. You can easily assign all settings from a view camera to a camera object and back.

Input Plane

The input plane is a flat, infinite 2D plane in the 3D modeling space which you see through the view windows. The mouse pointer position is projected to the underlying input plane. This way 2D coordinates from the mouse become translated into 3D coordinates.

the input plane is, by default, connected to the view camera so that it always remains orthogonal to the camera. You can also unlock the input plane and model in a perspective manner. For example, you may want to view a room scene using a final camera view, place the input plane on the floor and move people inside the room along the floor to obtain an optimal composition.

The input plane becomes visible when you activate grid drawing. The input plane always matches the space division into grid snapping intervals. You can move and rotate the input plane using the view controls, numerically from the view property window or using the popup menu.



Perspective modeling on a vertical input plane

Most geometry creation and modify tools read their coordinate input from the input plane position. Certain tools (those using object handles) do not rely on the input plane, but project the mouse input to circular handles, axis systems etc. directly. For example, when you edit SDS object faces, input plane may be irrelevant.

View Control Bar

The View Control Bar is docked to the right side of the view window in the default configuration. You can open and close it from the 'View/ViewControls' pull down menu or using the F5 hotkey. If the contents of the view control bar do not fit inside the visible area, you can scroll it by dragging it with the right mouse button 'Ctrl' key held down, or with the middle mouse button.

You can open a popup menu from the left border of the view control bar. With the menu, you can turn icon labels on and off, and control the visibility of the view control groups.

- *Show labels:* Text labels under icons on/off.
- *Show Target Group:* The 'Target' group has only one selector gadget, which defines what the controls in the next 'Camera' group modify.
- *Show Camera Group:* This group contains the four controls for rotating, zooming, changing the distance and panning. By default, the target is the view's camera.
- *Show Interactor Group:* The 'interactor' group contains four tools which need mouse click input on the view window. The tools control the aimpoint and the input plane.
- *Show Orientation Group:* This group contains several tools for switching the orientation of various view items. For example 'Reset View' button is located here.
- *Show Canvas Group:* The Canvas group contains miscellaneous view controls. OpenGL, GDI drawing and Ray Trace buttons are located here as well.

The target selector gadget selects the modify target for the Camera group controls below the selector gadget. The following targets are available:



- View
- Input plane
- World orientation
- Selected objects
- Texture maps of selected objects

Camera Group

The Camera group contains four icons which control the orientation and position of the selected target. Use the icons by pressing the left mouse button down on top of an icon and moving the mouse LMB held down. The icons are (in up to down or left to right order):



- **Rotate:**Rotates the selected target.
- **Zoom:**Changes the camera scale / resizes the selected target.
- **Pan:**Moves the camera or another target sideways, up and down.

- **Distance:** Moves the camera forward/backward or another target closer/further away.

Interactor Group



The common nominator of the tools in the Interactor group is that after a tool has been selected, the program expects some mouse clicks (or LMB dragging) on the view window. The tools are:

- **Zoom:**This is a combined pan and zoom tool. To use it: Activate the tool. Then LMB drag a rectangle on a view window. Note how the dragged rectangle has automatically the proportions of the view window. After left-to-right dragging, the defined rectangle area is magnified to fill the entire view. Reversed dragging from right to left will shrink the current view area to fit the defined rectangle.
- **Aimpoint:** Sets the center point of the view. The new aimpoint position can be defined with a left mouse button click, but also snapping operations can be applied. Use this tool to center the view exactly to the desired point.
- **IP by 3P:** Reads three points and fits the input plane to match the defined plane. The line between the two first points will become the input plane's x axis. For example, you can fit the input plane to any triangle or rectangle by applying this tool: just activate the tool and then LMB drag three corner points with the depth modifier key ',' (comma) held down.
- **IP by N:** Fits the input plane to surface tangent. Activate the tool and LMB click over a surface. You may also LMB drag (preferably grid drawing on) to see how the input plane rotates; releasing the mouse button sets the input plane.

Orientation Group



The Orientation group contains tools which set predefined orientations for various view window related items. Most of these tool icons have also a side menu which opens with a LMB click. The side menus contain some additional variations of the tools.

Front: This tool puts the view camera to the front view orientation. The camera will look against z axis (heading 180 degrees, pitching zero). The camera is rotated to the front view about its aimpoint. Also the input plane is rotated so that its normal matches the z axis. Shift clicking the icon gives the opposite view (the back view).

The side menu has two additional versions: 'View' and 'Input Plane'. These tools rotate the view camera or the input plane only. Note, however, that all the three versions perform the same task when the camera and the input plane are locked together.

Side: The side view tool. Side view is the view from the left side of the object (along positive x axis). The side menu is similar to the front view above. Shift clicking the icon gives the opposite view (the right side view).

Top: The top view tool. View camera and input plane are rotated so that the camera looks down against y axis. See the front view tool above for the side menu explanations. Shift clicking the icon gives the opposite view (the bottom view).

Reset: Resets the view back to the 'Native state' that has been stored into it. This usually changes camera and input plane positions and orientation, as well as the zoom scale. The side menu has three entries:

- *View to default front view:* Total view camera reset. The native orientation may get corrupt, or otherwise to a badly defined state. This menu offers an easy way to put it back in order. Aimpoint and grid origin are set to absolute space origin, the camera takes a front view and the zoom scale is reset to the default value 1.0. The initialized values define the new native state of the view.

- *Reset world:* It is possible to define an additional global transformation for the whole scene hierarchy. This so called world transformation is useful when when the world coordinates must be aligned to a part of the scene hierarchy. For example, in a landscape model, the original world coordinates perhaps match the north-south and east-west directions. A building on the landscape seldom matches the same directions. When editing such an irregularly oriented building, it is helpful to set the world coordinates to match the walls. Numerical input for modeling tools becomes easy to interpret etc.

- *Set native orientation for the view:* Records the current state of the view window to be its native state. Next time the reset button is hit, the view returns back to the same state.

Fit: The autofocus tool. The view camera is aimed at the center of the selected objects and zoomed/moved so that the selected objects fill about 75% of the view window. Also the clipping range is adjusted so that all selected objects become visible. Shift-clicking the tool focuses on the selected edit handles, when the selected objects are set to editing state.

Fetch: Fetches the view camera orientation from the selected object. The view will be moved to its position and rotated to match its object space. For example, you can take the view from a camera object or from a spot light this way. The side menu is the following:

- *View to object:* Copies view orientation to selected objects. Usually this tool is used for recording a view to a camera object, but you can also set spot light orientation this way, for example.

- *Object to input plane:* Aligns the input plane to the selected object. Also the view camera follows if it is locked to input plane. It is easier to modify the object when mouse coordinates (and grid orientation) follow the object's orientation.

- *Object to world:* Aligns the world coordinate directions with the object space directions of the selected object.

Sync IP: Rotates the view camera so that it is orthogonal to the input plane. This tool may be useful when using a disconnected camera-input plane system. The attached menu is the following:

- *View orientation to input plane:* Same as above, but this time input plane is rotated, not the view camera.

- *Input plane orientation to world:* Aligns the world coordinates with the input plane. There are many tools which align the input plane to objects, surface tangent etc, which helps in accurate modeling. If usable reference coordinates are also needed and the input plane does not follow world's coordinate directions, you may have to match the world temporarily with the input plane.

Canvas Group

Persp: Sets perspective projection for the view on/off. The parallel camera projection is often more suitable for accurate modeling purposes.



Draw: Enables and disables snap grid drawing. A grid must be selected before it can be drawn.

Snap: Enables and disables grid snapping. A grid must be also assigned for the view, otherwise snapping does not work. You can assign the grid by drag and dropping it from the select window into the view window, or using the view property window.

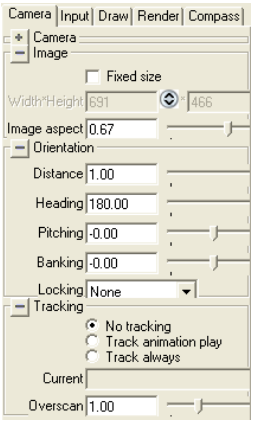
OpenGL: Activates real time drawing via the OpenGL interface. By default, it renders shaded surfaces, but you can change the options from the view property window. OpenGL is a faster alternative than Windows GDI drawing if a modern graphics accelerator is available.

Windows: Selects Windows GDI drawing which is always in wireframe mode. GDI drawing is not hardware dependent and a safe alternative if OpenGL driver compatibility problems appear.

Ray Trace: Ray trace renders the view window.

View Properties

A view window is the most important element of a 3D program. Consequently, a large set of options is available for controlling its operation. These options are collected into the view property window.



The view property window opens from several places:

- From the view's popup menu 'View Properties...'
- From the pull down menu 'Windows/View properties'
- By double clicking a view model object in the select window

The Controls of the view property window are divided into several tabs, described below. The edited view model can be selected using the **Auto select** option; if the option is enabled, the view that has been clicked last (i.e. has the input focus) will become the editing target. Clearing the 'Auto select' option shows a list of all view models and you can activate the editing target by clicking an item on the list.

Camera frame: This is a standard camera object property gadget. Every view window contains an internal camera object and you can edit its properties here.

Fixed size: If set, the pixel resolution of the view window is determined by the Width and Height fields below this control. If cleared, the resolution is defined by a host container such as a floating window.

Width*Height: The pixel resolution of the view window. These fields control the resolution when the 'Fixed size' option is enabled.

Image aspect: The ratio heigh(pixels)/width(pixels). Changing the value rescales height. Note that the aspect is computed from the plain pixel resolution, and does not take into account the pixel aspect ratio itself (non-square pixels).

Distance: The distance from the camera position to the aimpoint. Aimpoint is the origin of view navigation rotations and the focusing point for depth of field.

Heading: Cameras direction (axis projection) on horizon plane. Heading zero corresponds viewing along the positive z axis. The front view heading is 180 degrees.

Pitching: Camera angle with respect to the horizon plane. Front and side view pitching is thus zero, and top view pitching is 90 degrees.

Banking: Camera's tilt angle.

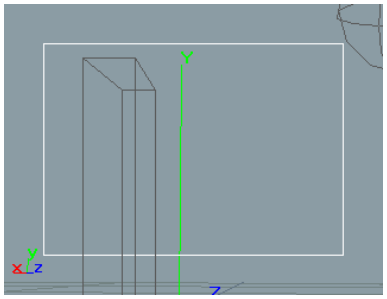
Locking: View camera's orientation and other 3D space related properties can be locked here. For example, you may want to lock one view of a quadview system to display permanently the front view. Or, you can lock a certain carefully chosen camera setup on a raytrace preview window. This prevents accidental changes on the window after rendering tests when you return back editing the model.

- None - View camera can be modified without limitations
- All - Camera's direction, distance, position, lens opening angle etc. cannot be changed
- Direction - View camera keeps its direction but you can move it and zoom it

Tracking: Tracking controls define how the view camera should be connected to the currently active camera object in the scene hierarchy. The following alternatives are available:

- *No tracking:* View camera is independent of camera objects of the scene.
- *Track animation play:* View starts showing the current camera view as soon as the animation playback starts. Tracking is automatically ended when the animation play stops.
- *Track always:* Bi-directional tracking. View updates always to show changes in the current camera. Navigating or otherwise changing the view copies the view's camera state to the current camera object in the hierarchy (if such one exists).

Current: Displays the name of the tracked camera object.



Overscan tracking with value 0.75. The white rectangle defines the view area of the the tracked camera

Overscan: Overscan allows you to view slightly larger/smaller area than the actual tracked camera shows. For example, if the overscan is 0.75, the camera view will fill only 75 % of the view window. This may be useful when analyzing visibility of objects that spread their influence, for example glow. Value 1 gives an exact match.

Input

Grid frame includes grid snapping controls. Grid snapping is view specific: each view can use different grid settings.

Grid snap: Activates grid snapping. Mouse input will be rounded to grid units.

Grid draw: Activates grid drawing, which acts as a visual aid in modeling.

Current grid: Selects the grid which will be used for snapping.

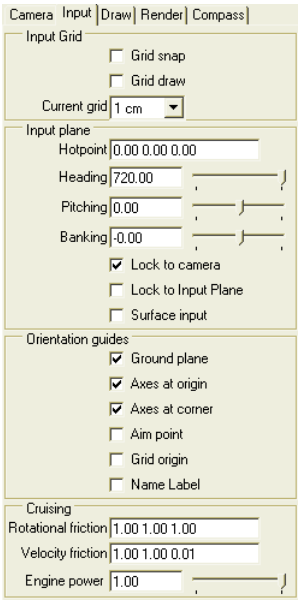
Input Plane is a two-dimensional infinite plane in the modeling space, projecting the mouse pointer movements to 3D coordinates.

Hotpoint: A point in space trough which the input plane travels. Defines also the grid origin point: grid snap intervals are computed starting from the hotpoint.

Heading: The projected direction of the input plane normal in the horizon (XZ) plane.

Pitching: The angle between the input plane normal and the horizon plane.

Banking: Tilt angle of the input plane. Banking rotates the grid around the input plane normal.



Lock to camera: If set, the input plane is automatically kept orthogonal to the view camera direction. When the user navigates on the view and rotates the view's camera, the input plane follows the rotation.

Lock to Input Plane: If set, selecting an input plane object or a level containing an input plane from the scene hierarchy automatically sets the input plane. The view updates automatically when the input plane is modified.

Surface input: If set, coordinates inputted from the mouse are projected to underlying object surfaces. For example, you can draw a path following a terrain surface.

Ground plane: Activates drawing of a horizon plane grid (XZ plane trough origin) in perspective view mode. The ground plane helps to keep track of directions when navigating on a view window. Ground plane is not drawn when using a parallel view projection.



Note

Do not mix the ground plane with visible grids. Ground plane has nothing to do with snapping.

Axes at origin: If set, a coordinate system is drawn at the origin point of the space. Each axis of the coordinate system shows a small label (X, Y or Z). The coordinate system has fixed 3D dimensions, so scaling the view window or moving camera further away from the origin makes it appear smaller, which is a useful visual aid in view navigation.

Axes at corner: Activates drawing of another, smaller direction indicator at the bottom left corner of the view window. This coordinate system is drawn at a fixed position and with a fixed size. Therefore, unlike the origin coordinate system, it remains always visible and available for orientation checking, regardless of view panning and zooming.

Aim point: Turns aimpoint drawing on/off. Aimpoint defines the position in space around which the camera navigates. Therefore, it always appears in the middle of the view window. The aimpoint marker has also 3D space defined size which indicates camera's distance from the aimpoint and zoom scale.

Grid origin: Draws a marker for the origin point of the input plane (sometimes called as the hotpoint). This point is the base point for grid snapping. Most tools measure numerical input from the grid origin in the beginning of interaction. The grid origin marker has a 3D space defined size, meaning that it shrinks and expands when the view camera is zoomed or moved.

Name Label: If enabled, the name of the view model is drawn on the top left corner of the view window. This is useful for direction locked views, which can show the name of the projected direction (front/top/side). Also other kind of view windows can inform the user via the label option: a preview window can be labeled as 'Click me to render'.

Ruler: If enabled, a line segment indicating a distance interval is shown on the top left corner of the view. The length of the line is also printed using the distance unit selected from File/Preferences window. The ruler helps to understand the current scale of the scene. It is especially useful in conjunction with a visual grid.

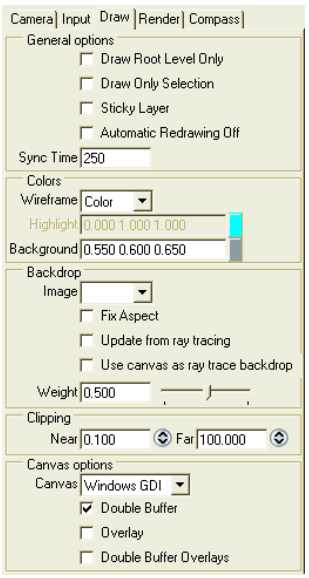
The cruising controls are available for 'vehicle like' navigation in 3D space. Cruising behavior is controlled with a set of coefficients which describe the properties of surrounding space/matter (vacuum, air, water etc.) and the aerodynamical properties of the 'virtual vehicle'. The friction factors combine the space and aerodynamics properties. The higher a factor, the harder the surrounding matter resists the respective motion component. All these factors use a relative scale so that factor 1 gives a good default behavior.

Rotational friction: Defines how the surrounding space resists rotations around vehicle axes. The first factor is the friction of the rotation around vehicle's 'up' axis (heading), second factor is friction against pitching type rotation and the last factor is the banking friction.

Velocity friction: Friction factors for movements along the three main axes: sideways, up/down and forward/backward. A dart has factors like (1, 1, 0.1). A thin metal plate in water has two almost zero factors and one factor which is very high: (0.1, 0.1, 10). A car has factors like (sideways=5, up/down=100, forward=1) (actually 'down' friction should be even higher because the car cannot pass through the ground).

Engine power: Defines how quickly the vehicle accelerates when a navigation control is applied. The suitable value depends on the dimensions of the scene, speed of the computer, complexity of the scene etc. Too strong engine power makes navigation hard to control because changes happen too rapidly.

Draw



Settings of this tab control real time rendering.

Draw Root Level Only: When this option is enabled, only the part of the object hierarchy that is visible on the select window is drawn. You can define the appropriate branch of the object hierarchy by double clicking a level object with the Shift modifier key held down. A double click on the highest level moves the 'virtual root' one step higher.

Draw Only Selection: If set, only selected objects will be drawn. FOr example when point editing mesh objects, it may not be necessary to see other objects.

Sticky Layer: If set, the view window continues drawing the project which was active when the option was set. Normally the view windows draws the current project.

Sync Time: Defines the minimal time in milliseconds, after which drawing can be interrupted. If the time is short, wiew reacts immediately to input events, when the user navigates or modifies objects. However, too rapid interruption may appear unpleasant and flickering, if only a small part of the scene gets drawn before next refresh. Setting a long sync time may be useful when working with complex scenes.

Automatic Redrawing Off: Normally view windows refresh automatically when the scene changes. This option turns automatic drawing off. The view will be redrawn only when the user changes the view orientation or selects Render/Wire, Ray Trace etc. from the view's popup menu (or uses another rendering command). The option is quite useful for ray tracing preview windows, which will then keep their latest rendered image as a reference while the user edits the scene.

Color options define view's background and wireframe drawing colors.

Wireframe: Defines the color scheme for realtime rendering of wireframes. The following alternatives are available:

- *Black:* Objects are drawn with two colors: one color for unselected objects and another highlight color for selected objects. The unselected color can be changed at Classes/Root/Wire Frame/Renderer/UnSelectedColor. The color for selected objects is defined using the Highlight gadget.
- *Own:* All objects are drawn using the exact RGB values of object's diffuse color. No selection highlighting is used.
- *Class:* This alternative allows drawing by object type specific colors. The user can define an individual color for spheres, cubes etc. from the wire frame class attributes.
- *Color:* This is the default drawing alternative. It uses a modified version of object's diffuse color. A darkened version is used for an unselected and a lightened version for a selected object. The drawing system also verifies that no object is drawn with a color that is too close to the background color (making object invisible).

Highlight: The color for selected objects, when Black wireframe coloring scheme is used.

Background: View's background color. Backdrop options below can overrule this setting.

With the Backdrop options, the user can add an image such as a photograph to the background of the view window. The backdrop can be used as a modeling guide, or for image composition purposes.

Image: Selects the image object to be used as a backdrop. You can create image objects at the Image tab of the select window. You can also assign the backdrop by drag and dropping it from the select window into a view window. You can remove the backdrop by selecting the empty entry from the image gadget.

Fix Aspect: Normally the backdrop image is scaled to fit the view window dimensions exactly. If the Fix Aspect option is set, y scaling obey's image's original proportions and pixel aspect ratio. Usually this means that part of

the backdrop is clipped at the top and bottom parts of the view, or that the backdrop does not fill the entire view window in y direction.

Update from ray tracing: If set, the result of view ray tracing is automatically set as the view backdrop. A ray traced image may provide a useful reference for further modeling actions.

Use canvas as ray trace backdrop: The opposite of the previous option. The contents of the view window is used as a background image for ray trace rendering.

Weight: The visibility of the backdrop. Value zero makes backdrop invisible, value 1 fully visible. The intermediate values are useful for making a backdrop less distracting and object wireframes better visible over the background.

Clipping values define a visible distance region in front of the view camera. Only geometry within the region will be drawn. This sometimes helps to extract the relevant target area, which is under construction or needs modifications, from the rest of the (possibly complex) geometric data of the scene. Hardware accelerated rendering (OpenGL) often uses quite limited accuracy to store distances, for example 16 bit integers. In such cases, too wide clipping range makes z buffer inaccurate and leads to incorrect object visibility and other rendering errors.

Near: The start distance of clipping. By default, 0.1 meters.

Far: The end distance of clipping. By default, 100 meters.

Canvas options depend on the selected realtime rendering system. The following two alternatives are available by default:

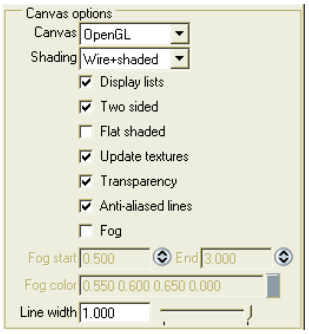
Windows GDI

Double Buffer: Double buffering means that the screen is refreshed by drawing into a hidden buffer which is made visible after the image is ready. This reduces screen flicker.

Overlay: Allocates a separate mono color buffer to which editing handles are drawn. This reduces the amount of redrawn geometry data during object editing and hence speeds up screen updating.

Double Buffer Overlays: Activates double buffering for overlay drawing, reducing flicker in object handle drawing.

OpenGL



Shading: Real-time drawing mode. The choices are:

- *Wire:* Objects are drawn using basic wireframes
- *Shaded:* Objects are drawn using shaded surfaces
- *Wire+shaded:* The combination of the two modes above. Wireframes are drawn over shaded surfaces.
- *Hidden line:* Wireframe drawing with hidden line removal

Display lists: Activates display list optimization. This may sometimes speed up drawing.

Shaded+wire: In shaded mode, draws wireframes over shaded surfaces. A small z buffer offset is applied to ensure wire visibility. The offset value is defined by Preferences/Classes/Asynchronous Drawing/WireOffset value.



Note

For SDS and NURBS objects, setting Shaded quality = Wire density in Wireframe properties often gives the best result.

Two sided: If cleared, only the polygons whose normal is facing towards a light are shaded. This speed up rendering but may create rendering errors (when using unclosed surfaces such as a rectangle).

Flat shaded: Turns off smooth normal interpolation.

Update textures: Controls texture updating. Turning updating off sometimes accelerates drawing considerably (when edition objects with high Texture quality and complex VSL materials).

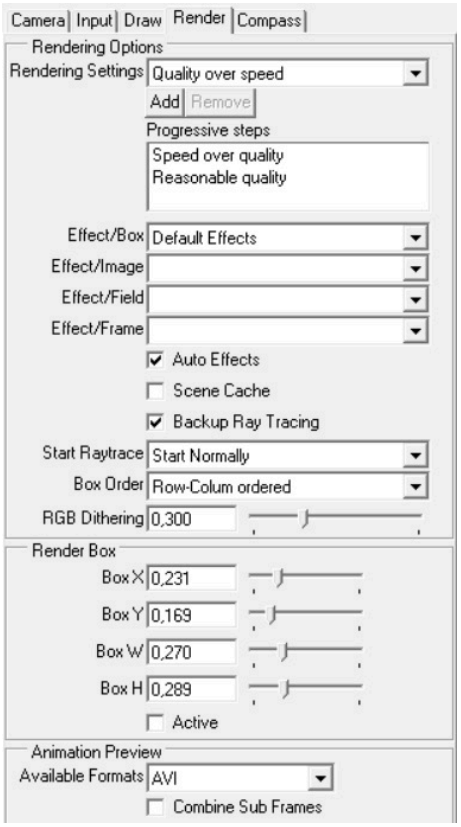
Transparency: Enables alpha channel use in OpenGL drawing. Transparency is used extensively in making edit handles visible through shaded surfaces.

Anti-aliased lines: Wireframe anti-aliasing improves drawing quality but slows down screen refresh rate.

Fog, Fog start, Fog end, Fog color: Realtime fog effect. Rendered geometry will fade into the given fog color within a distance range defined by Fog start and Fog end. Fog effect gives an additional visual cue and helps in viewing complex wireframe models. The effect works best if the fog color is the same as the view's background color.

Line width: Defines wireframe line drawing width in pixels. If Anti-aliased lines is disabled, only integer values 1, 2, 3 etc. are significant. Enabling antialiasing increases steps in the width setting.

Render



Rendering Settings: Selects ray trace rendering settings for the view window.

Progressive steps: With the **Add** and **Remove** buttons, one can construct a list of rendering settings, which are all applied to the view automatically before the final rendering quality defined in the Rendering Settings field. The purpose is to provide a quick preview which then proceeds towards the final quality with a single click of Render button. A typical setup could be: Progressive steps = Speed over quality + Reasonable quality and Render Settings = Quality over Speed.

Effect/Box: The post processing configuration that will be applied to all rendered image boxes.

Effect/Image: These post effects are performed for complete images (not for single render boxes when using box rendering).

Effect/Field: Post effects that are applied after motion sample assembly to fields. Note that even when motion blur or field rendering is off, these effects are still applied to the finished image after box and image effects.

Effect/Frame: Post effects that are applied to complete frames after field assembly.

Auto Effects: Controls view's behavior in project loading. If the user loads a project which replaces the post effect library, but does not load view models (to keep the original GUI settings), the Auto Effect system assigns the first loaded post effect configuration to every view model, which has the Auto Effect option enabled. In other words, ray trace rendering of the view window will include the loaded post effects automatically. Auto Effect feature considers the Usage Recommendation of the loaded effects (per box, per image etc.)

Disabling Auto Effects may be desirable for special purpose view windows, for example to ensure maximally quick draft rendering without any post effects.

Scene cache: Activates a cache for ray trace rendering data. Optimization data from the previous ray tracing will not be deallocated after rendering. If only materials or the camera changes, new ray tracing starts much quicker (especially when using objects that require heavy preprocessing, such as dense SDS surfaces). The option adds memory consumption.

Backup Ray Tracing: When enabled, the deep channel data resulting from ray tracing is stored into a disk file. The data can be viewed later and the post effects can be renderer using it. This saves rendering time when adjusting or experimenting with the post effects, because ray tracing needs to be computed only once.



Note

If view window size or post effect configuration (and hence the required channel set) change, the backup file may become invalid. Post processing an invalid file may produce a black image.

Start Rendering: Selects the action which starts raytracing.

- *Start Normally* - Rendering starts when the user selects raytrace from the view's popup menu, hits the respective hotkey or clicks the raytrace view button.

- *Continuous Rendering* - Rendering starts automatically whenever the scene or the view orientation changes.
- *Click To Render* - A left mouse button click starts rendering. This option is very handy for a render preview window which is not used for object editing.

Box Order: Selects the order in which boxes are rendered

- *Row-Column* - Rendering starts from the left top edge and continues from left to right, row by row.
- *Scattered* - Boxes to be rendered are scattered all over the view.
- *Centered* - Rendering is started from the middle.
- *Inverted Row-Colum* - Rendering is started from the right bottom edge.

RGB Dithering: The amount of view window specific RGB dithering that is applied when converting the 24 bit RGB data into the view to lower color depths. When using 24 bit display mode, the setting is unused.



Note

Render settings include a more accurate general dithering that works with all RGB bandwidths.

Box X: The left edge of the current render box measured in relative 0..1 coordinates. 0 = the left edge of the image, 0.5 = the middle point of the image etc.

Box Y: The top edge of the render box. 0=top edge of the image, 1=bottom edge of the image.

Box W: The relative width of the render box.

Box H: The relative height of the render box.

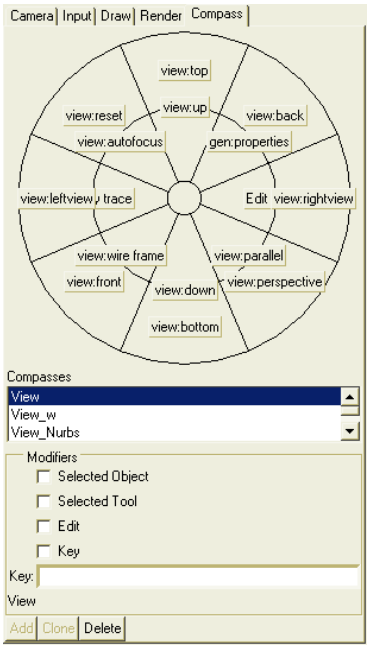
Active: Turns render box on/off. You can disable a render box to render a full view and then enable it again.

Animation Preview/Available Formats: Selects which format will be used to store the rendered images when view's popup menu Render/Animation Preview is used.

Combine Sub Frames: If set, the view shows the combination of fields and motion samples (=the actual frame) in OpenGL mode. The option has no effect if motion blur and field rendering are off, or if the view does not use OpenGL. The combined sub frames can be outputted to an animation preview file for motion blur preview purposes.

Compass

With the compass editor, you can configure a suitable set of compass menus.



Compass: The compass diagram shows the macros that have been bind to the selected compass. You can add items to the compass by drag and dropping named macros from the macro tab of the select window into the desired area of the compass. Items can be removed by applying the Delete popup menu on top of them.

Compasses: The list of already existing compasses. You can click a compass item to activate it for editing.

The user can define several compasses per view window. The modifiers attached to compasses define which compass pops up when the user RMB drags with the mouse. The defined modifiers will be set when a new compass is created.

Selected Objects: Makes the compass object type sensitive. The object that is selected when this option is set will activate the compass.

Selected Tool: Makes the compass tool sensitive. The tool which has been activated when this option is set will activate the compass.

Edit: Makes the compass point edit mode sensitive. Compass will activate only if the selected object is in point edit mode.

Key: The compass is selected by a key modifier. Set this checkbox, activate the string gadget below and hit a key to be used as a modified. For example, view orientation compass appears if you press 'v' key while RMB dragging.

Add: Creates a new compass with the defined modifiers. If the modifier set conflicts with another compass, Add button remains disabled.

Clone: Copies the selected existing compass. Before a compass can be cloned, a different modifier set for the duplicate must be defined to ensure that compasses will not conflict.

Delete: Deletes the selected compass.



Chapter 2. BatchRen

Command line rendering utility

Description

Command line based batch rendering utility for rendering Realsoft 3D project files. Batch renderer needs less resources than the full Realsoft 3D program. Not only the program code is smaller but also the actual project cosumes less memory space because only rendering specific data needs to be loaded into memory.

Batch renderer renders projects exactly the same way as File/Render window in Realsoft 3D. In fact, batchrenw and File/Render window are just two different interfaces to the Realsoft 3D rendering engine.

To use the batchren application, you need to create a project in Realsoft 3D, specify valid file rendering specifications via File/Render menu and then save the project.

Batch renderer uses the Realsoft 3D's distributed nerwork renderer so all the options you specify in Realsoft 3D will also work in batch renderer.

You can pass any number of projects to batchren. The program loads the given projects one by one and renders each project separately.

Startup Options

The program supports a set of startup switches:

-file [file;file2...]	This option can be used for specifying one or more configuration projects. These projects are loaded with the actual project to be rendered. For example, the configuration project may define light sources, camera orientation and rendering settings allowing you to easily visualize iges, vrml and other files defining only geometric objects.
-plugins [foldername]	Specifies custom plug-in folder for the renderer. The default folder is Realsoft 3D's plugins folder.

-play

Renders all the frames of the project rather than the current frame.

-first [frame]

-last [frame]

These options can be used for defining frame range for animation rendering. They have effect only if *-play* option is defined.

To render a single frame, pass the same frame number to both options.

-out [filename]

Define file name for the image / animation to be rendered. This overrides the file name specified in the project's output settings.

-frset [objectname]

This option can be used for specifying the rendering specification object to be used for rendering the project. If not given, the currently selected configuration (the one shown by the File/Render window when the project was saved) is used.

This option allows you render your projects with different resolution, quality etc. For example:

batchren -frset PAL myproject.r3d

batchren -frset NTSC myproject.r3d

-hosts ["host1;host2..."]

This option can be used for overriding the distribute network rendering servers specified in the rendering settings / hosts list of the project.

For example to render 'test.r3d' by using three additional rendering servers (moon, mars and rserv.mydomain.com):

batchren -hosts "moon;mars;rserv.mydomain.com" test.r3d

-width [xresol]

-height [yresol]

These options override the image resolution defined by the project to be rendered.

For example, to render 'myicon.r3d' project with 32 by 24 pixels:

batchren -width 32 -height 24 myicon.r3d

Examples

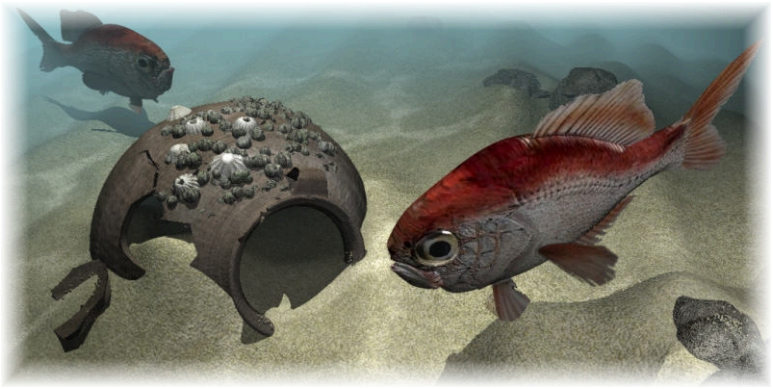
batchren -file myconfig.r3d test1.igs test2.igs

batchren -file "flyaround.r3d;red.r3d;globalillum.r3d" car.r3d

batchren -play -first 10 -last 20 -out testanim.avi myproject.r3d

batchren -play -first 10 -last 10 myanim.r3d

batchren -out testimage myproject.r3d



Chapter 3. BatchRenW

Batch Rendering Utility, with graphical user interface

Description

This program is a rendering utility with a graphical user interface. To use the program, use the Browse button to add one or more project files to the project list and specify whether the project should be rendered as a still image or as an animation. When done, click Render All to get all projects in the project list rendered. You can also render a single project using the Render Selected button.

Due to dynamic and scalable nature of Realsoft 3D, batch renderer needs significantly less resources than the full Realsoft 3D program. Not only the batch renderer application is smaller but also the actual project eats less resources because only rendering specific data is loaded into memory.

Using BatchRenW

Batch renderer renders projects exactly the same way as File/Render window in Realsoft 3D.



To use the batchrenw application, create one or more projects in Realsoft 3D, specify valid file rendering specifications via File/Render menu for them and then save the projects. In batch renderer, add the projects to the project list and click Render All. Batch renderer uses the Realsoft 3D's distributed network renderer so all the options you specify in Realsoft 3D will also work in BatchRenW.

The Servers tab has Autodetect check box. When set, the program will use all available rendering servers for rendering. If not set, then the rendering settings saved with projects determine the servers.



Chapter 4. RendD

Network Rendering Daemon

Description

RendD is the actual rendering service for the distributed rendering system. It can be started directly or through RendW program.

RendD program must be running on every workstation which is acting as a rendering server. The program listens incoming rendering requests and starts rendering threads to take advantage of the processor power available in the workstation.

The program can be used for distributing rendering over any TCP/IP based network, such as the Internet.

In addition to local threads, RendD can distribute rendering to other rendering daemons running on other workstations. This means that hierarchical rendering farms can be configured.

In distributed network rendering, all data of the rendered project needs to be sent to all rendering servers. Transmitting data becomes often the bottleneck especially when one needs to distribute rendering over slow networks (which may be the case when distributing over the Internet). By using hierarchical rendering networks, the client sends projects only to a small number of servers, which then distribute the task further to specified sub daemons of a fast LAN etc.

Two methods exist for setting up hierarchical rendering farms. Sub rendering daemons can be specified 'manually' as startup parameters. Another alternative is to the Automatic Sub Servers option, which automatically distributes rendering to the workstations available in LAN or registered to it. Automatic approach has some advantages, because it makes the configuration of the rendering farm flexible - it can change any time.

Startup Switches

-noproxy

Turns off image map caching to the local system. The proxy option is active by default. Proxy system fetches referenced texture maps (and other required image files), which cannot be directly found from the given search

paths. The files are transferred in the native compressed form and saved to a temporary subfolder in the user's document folder. This has certain advantages:

- If a cached texture file is referenced again in a later rendering session, it can be loaded quicker from the local hard disk without any increase in network traffic.
- Proxy reduces network traffic significantly in hierarchical rendering configurations. This can have a dramatical impact on rendering speed when some of the network connections are slow. It is much quicker to move the textures only once over a slow internet connection to a local proxy server which then distributes them further in a fast local network than let all the servers to access the textures directly over the slow internet connection.

Proxy feature checks file sizes and dates and updates the cache automatically if necessary.



Note

To utilize proxy feature maximally, remove all search paths from the rendering daemon. Then all textures will end up into the local temp folders.

-proxycleanup

This option controls the proxy cleanup system. The temp folder is checked frequently and files older than the given amount of days are removed. The default value is 30 - the average lifetime of image files in temp folder is one month.



Note

Do not place important files into the temp folder! Proxy cleanup will delete all outdated files.

-threads [num]

Specifies the number of rendering threads spawned for each rendering request. The default value is the number of processors in the workstation.

-clients [num]

Specifies the maximal amount of clients, which will be served simultaneously. The default amount 1 is suitable for single user network rendering. In a multi user network, the daemon stops spawning new rendering threads soon as the defined client count is reached. The clients attempting to get rendering service from the fully occupied daemon will receive a failure code (which normally just eliminates the server from that particular rendering job but does not stop rendering, provided that other vacant rendering threads will be available).

-help

Prints out brief instructions for use.

-plugins [foldername]

Specifies a custom plug-in folder for the rendering daemon. The default folder is 'Plugins' (in the program's home folder).

-xboxcount [num] -yboxcount [num]

In order to distribute a rendering task, rendering daemons must subdivide the image to a number of sub images. Each sub image is then rendered by a separate local thread or a sub daemon. When all sub images are ready, they are combined and returned back to the client. The box count parameters can be passed to override the automatic box count detection system.

Note that the sub image count should always be higher than the number of rendering threads and sub daemons available. For example, if -threads is 2 and 4 external workstations are used as sub daemons, then the recommended

box count is 6 or more. A good choice could be 6 multiplied by 4, because rendering speed varies over the image area (an empty black corner renders instantaneously whereas a sub image including a glass object is much slower to render). A suitably high box count adapts better to such variations and keeps all processors busy. On the other hand, there is some overhead in managing the sub images - a very high number of boxes can start slowing down rendering.

If the user does not define any box count, the program estimates an optimal amount from the number of rendering threads and sub servers.

-paths [path1;path2;..]

This option allows you to define daemon side texture paths, which are used for resolving texture names.



Note

When using the texture proxy option, the texture paths should not refer to folders in external hard drives. An image is fetched to the proxy cache only if it is not found otherwise. Therefore, too extensive search paths may prevent proxy operations. The result is increase in network traffic. Search paths may safely include local texture libraries. For example, the local temp folder is automatically included in search paths.

-host [hostname] -port [port number]

These switches specify the host name and the port number used for connecting to this daemon. The default host name is the hostname of the workstation. However, sometimes you may need to override the default name, for example to make the daemon accessible through the internet.

The default port number is 10050. You may need to override the default port number in case the port is already used for something else.

-autodet

Enables the Automatic Sub Servers feature, which is useful when configuring hierarchical rendering farms. When enabled, the rendering daemon searches the network and automatically connects to all sub servers available.

-mem [bytes]

This switch is considered when the daemon is accessed by a client using Automatic Network Rendering. The value specifies how much memory this system can consume in rendering. All project files, which need more memory, will not be sent to the server. The default value for this parameter is the amount of the physical memory.

Examples

Use a workstation as a rendering server

If you just want to use a workstation as a rendering server, start RendD without any parameters:

```
RendD
```

The program will detect the number of processors in the system and automatically configure itself to take advantage of all the processing power.

Set up a rendering farm

Let's assume you have a network consisting of five workstations and you want to let your employee to access them through the internet. To minimize overhead caused by data transfers, you need to set up one root daemon and four slave daemons. Render farm clients connect only to the root daemon, which then distributes the rendering requests to the other four workstations. It is much faster to send the data to the workstations through the LAN than by sending them through the Internet.

To configure this, start RendD on the four slave workstations without any parameters:

```
RendD
```

Start the root daemon by specifying the host names of the four slaves as startup parameters:

```
RendD slave1 slave2 slave3 slave4
```



Tip

In Windows XP, you can find the host name of a computer from Control Panel/System/Computer Name tab. You can use the full computer name defined there as the host name.

Use only one processor for rendering

If you have a dual processor system in the rendering network and you want to use only one of the processors for rendering, start the network rendering daemon as follows:

```
RendD -threads 1
```

Create a hierarchical rendering farm

Let's imagine you have a LAN consisting of large number of workstations. The LAN configuration or availability of its computers for rendering purposes changes constantly. In such a case, it is best to use the Automatic Sub Servers option.

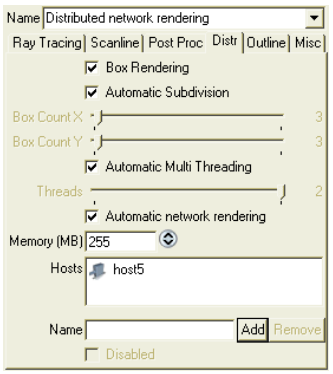
Let's assume one workstation 'host5' can be constantly available for rendering. Start the network rendering daemon in it as as follows:

```
RendD -autodet
```

Start the render daemon in other workstations as follows:

```
RendD
```

In the render settings of the client application, for example a view window of Realsoft 3D, enter the host name 'host5' into the Hosts list. Do not use the Automatic Network Rendering option, because the program will connect to the servers both directly and via host5.





Chapter 5. RendW

Network Rendering Daemon with a graphical user interface

Description

RendW is a graphical front end to the network rendering daemon RendD. RendW can start the RendD daemon any time, so that other users can start using your workstation, and stop it in case you need all the resources for yourself.

Pull-down Menus

When RendW is started, it creates a task bar icon, through which you can control various rendering options (Windows and Mac OSX versions only). There is a pop-up menu associated with the task bar icon, containing the following items:

Open — Open the window for controlling the daemon's options

On-line — Start the network rendering daemon (RendD) and allow network users to access the workstation for rendering.

Off-line — Stop the network rendering daemon. The workstation no longer provides network rendering services. Ability to enable/disable the rendering daemon is particularly useful when the rendering resources are shared using automatic network rendering. Users can add and remove resources to the rendering network in an easy, flexible way.

Shutdown — Shuts the RendW program down completely, removing the task bar icon and stopping the network rendering daemon.

Startup Options

The program accepts the same startup parameters as the RendD daemon. See RendD documentation for more information about the RendD specific startup options.

In addition to this, the program defines the following startup options:

-hide

The Control Window

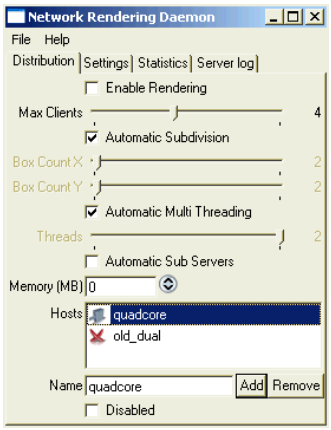
When the task bar icon 'Open' is selected, the window shown on the right is opened. The options of the window correspond the startup switches of the RendD rendering daemon.

Enable

Starts / stops the rendering daemon (RendD) with the specified options. This option can also be controlled via the On-line and Off-line pop-up menus associated with the task bar icon.

When enabled, network users can use the workstation for rendering.

Max Clients



Specifies the maximal amount of clients, which will be served simultaneously. Under normal circumstances (i.e. when the daemon is configured to use all available processors and sub servers), this control should have value 1. Then only one client will be served at a time. Providing rendering services to several clients simultaneously can cause virtual memory swapping, which slows down rendering. Powerful multi processor systems with a large memory capacity can naturally serve several clients. For example: a quad processor system with 8 gigabytes of RAM in a network of multiple users, who constantly need rendering power, could be configured as follows:

- Max Clients = 4 (or more, up to the number of network users)
- Automatic multi threading = off
- Threads = 1

With these settings, the daemon will give each client only one rendering thread. One client cannot use all resources of the workstation and therefore several clients can be served efficiently.

Automatic Subdivision

Enables automatic image sub division. If checked, the program attempts to find an optimal box count based on the image resolution and the number of threads and sub daemons in external workstations.

BoxCountX / BoxCountY

If automatic subdivision is disabled, these fields determine the number of sub images generated to distribute rendering.

Automatic Multithreading

Enables automatic thread detection to take advantage of multiple processors available in the workstation (SMP). If checked, the number of rendering threads, which are started to serve each client corresponds to the number of processors. In other words, a client gets the total rendering power of the server computer.

Threads

The number of threads spawned for rendering. This option can be controlled only when *Automatic multi threading* option is not set.

Automatic Sub Servers

This option is needed only when setting up hierarchical rendering farms. If checked, the rendering daemon searches the network and joins the other servers it finds to the rendering farm. For example: a client connects to only one rendering daemon via internet. The daemon the redistributes rendering to it own sub servers using this option. The actual client will not know how many servers eventually render the image.



Note

You should never set the Automatic Sub Servers option for multiple rendering daemons in a same local network. Similarly, the rendering client and the rendering daemon running on a same LAN should not use the automatic configuration feature simultaneously.

Memory

Specifies the maximal amount of memory (in megabytes), which the computer system is allowed to use for rendering. This option is used only when the client uses the Automatic Network Rendering option. Clients won't send projects to servers, which don't have enough memory to render the scene or whose owner does not want to use more memory than a certain limit. By default, the value is set to the amount of physical memory.

Add / Remove

Add and Remove buttons below the host list allows you to add and remove host names manually. These fields are disabled if the Automatic Sub Servers option is active.

Disabled

If the host list is configured manually (Automatic Sub Servers off), you can select a host from the list and disable it temporarily using this checkbox.

Host / Port

These fields specify the host name and port number through which clients can contact the daemon.

Texture Proxy

Turns on image map caching to the local system. The proxy option is active by default and should be used by default.

Proxy Cleanup

This option controls the proxy cleanup system. Textures older than the given amount of days are removed. Cached textures are placed into temp folder.



Note

Do not place important files into temp folder! Cleanup will remove them.

Texture Paths

This field specifies search paths for textures. This field is only needed when texture proxy is not used. More than one search paths can be defined by separating them using a semicolon.

Debug

Enables printing of additional diagnostic information into the server log tab.



Chapter 6. ShowImg

Image Viewing Utility

This program can show any image format supported by Realsoft 3D. It can also compose and show temporary image boxes generated by distributed network rendering.

Pull-down Menus

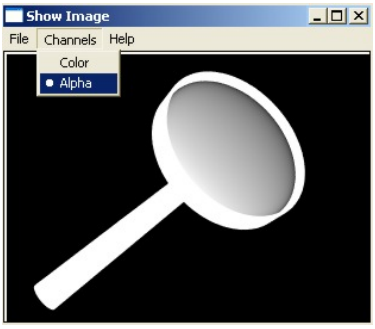
File/Open Rendering Boxes — This menu allows you to select one of the sub images generated by distributed renderer. All the other sub images are automatically loaded and shown. Missing sub images are shown as black boxes. This function is useful for viewing sub images generated by distributed rendering system. If you leave the programming running, it automatically reloads the images. This allows you to monitor how network rendering progresses.

File/Exit — Exit ShowImg.

Channels — Realsoft 3D images can consists of any number of channels, such as Color, Alpha, Depth, to name a few. This menu can be used for selecting the channel to be shown.

Help/About — Open About dialog, showing the version number of the program.

Help/Manual — Open this reference page.



Startup Parameters

Showimg accepts number of startup parameters.

-file [filename] — Specifies the default name of the image to be shown.

-boxes — If set, the program assumes the given file name refers to set of sub images which should be composed to complete image, if possible. For example, network renderer subdivides image to a number of sub images to get the rendering job distributed to several threads / workstations. In case of system crash, this switch can be used for asking the program to read set of sub images rather than single image file.

-plugins — Specifies custom plug-in folder from which the plug-in modules should be read. The default folder is 'Plugins' (in program's home folder).

-boxprefix This specifies a string to be used as a prefix for constructing box names. The default prefix corresponds to one used by Realsoft 3D's distributed rendering system.



Chapter 7. Software Updater

Software updater utility

Description

To update your Realsoft 3D software select `Help/Realsoft on the Web/Check updates` pull-down menu. This spawns `Software Updater` application which guides you through the downloading and installing process.

Software Updater can also be started separately to run on the background. It creates a small icon on the task bar indicating when new version become available. Downloading and updating new versions can be accomplished by clicking the Software Updater icon on the task bar.



Note

Save your work and close Realsoft 3D before proceeding with the installation! The setup software must close running Realsoft 3D before it can install new software.

Clicking the right mouse button on the task bar icon opens up a pop-up menu with two choices:

Open Open up user interface for manual operation.

Close Stop the auto updater process.

The user interface of Software Updater allows the user to manually control the download, and install operations and monitor their progress. This is the user user interface that opens up when Realsoft 3D `Help/Realsoft on the Web/Check Updates` is selected.

Closing the user interface will shut down the Software Updater. Use Hide pull-down menu to switch the user interface off and let the process to run on the background.



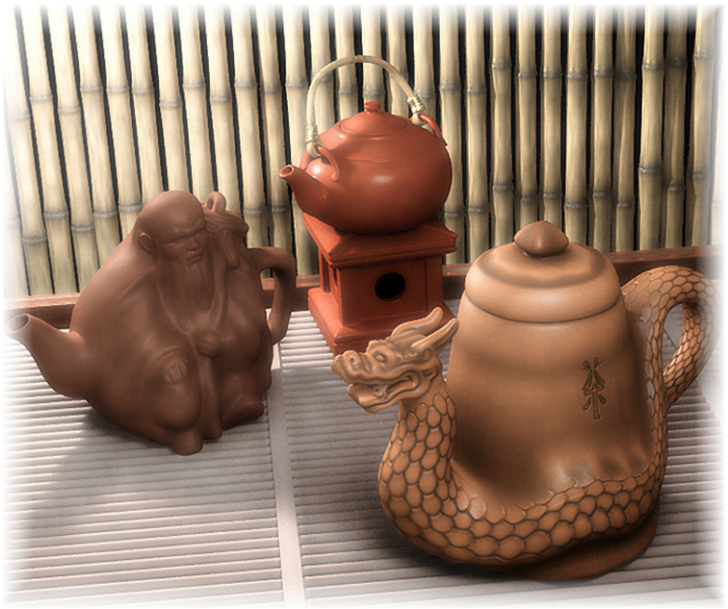
Tip

You can also install software updater to automatically run when you boot up your operating system. For more information about this consult your operating system documentation.

Startup options

You can change the following startup options:

-updatedir [dir]	Specify the download folder for the downloaded setup packages. The default folder is .Realsoft 3D folder in the users home.
-show	Opens up graphical user interface with progress bar controls for manual operation.
-autoclose	Check for new software updates. If the software is up-to-date silently exit. If new updates available open the user interface (if not open already).
-lang [lang]	You can choose between the installed languages for the graphical user interface of Realsoft 3D.
-dpi [dpi]	Dots per inc setting for icon scaling, overriding the dfault value determined by the operating system. Note that this has effect only if the current icon set is 'svg' (scalable vector graphics format).
-icon [dir]	You can choose between the installed icons for the graphical user interface of Realsoft 3D. Icons are installed into Realsoft 3D\r3icon folder.
-interval [second]	You can choose the interval for automatic update detection. By default the software checks for new updates ones per hour.



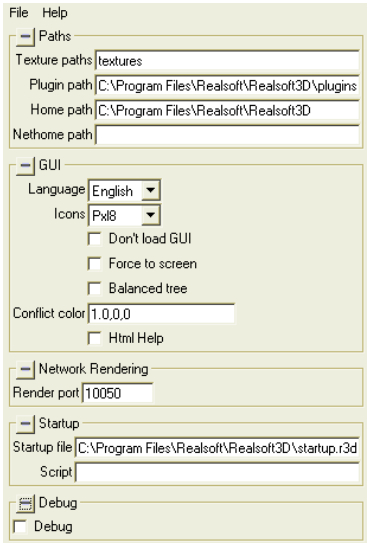
Chapter 8. StartOpts

Startup option utility

Description

StartOpts program is a utility for defining startup options for Realsoft 3D.

Startup options



You can change the following startup options:

Texture Paths

This field specifies search paths for textures. More than one search path can be defined by separating them using a semicolon.

Plugin Path

You can enter a path where Realsoft 3D searches for plugins.

Home Path

You can enter the home path for Realsoft 3D.

Network Home Path

You can enter the network home path for Realsoft 3D, if Realsoft 3D is installed on a network drive.

Language

You can choose between the installed languages for the graphical user interface of Realsoft 3D.

Icons

You can choose between the installed icons for the graphical user interface of Realsoft 3D. Icons are installed into Realsoft 3D\r3icon folder.

Don't load GUI

Specifies that the layout of the graphical user interface is not to be loaded if Realsoft 3D opens a project file. This can be useful if you are loading other people's projects which contain customized user interfaces.

Force to screen

This option moves all open windows into the visible screen area so that the user can see them. This is useful if you are loading other people's projects which use a different monitor resolution or multi monitor setups and the windows are displayed outside of your screen's bounds.

Balanced tree

Specifies that hierarchical windows are displayed as balanced trees.

Conflict color

Specifies the conflict color for gadgets. The conflict color is displayed if e.g. conflicting options are specified in a gadget. The conflict color is specified as r,g,b with float numbers. To change it e.g. to green, enter 0,1.0,0 in this field.

Html help

If enabled, the program will use the documentation in html form instead of the native format. When using this option, the html manual must be installed.

Render port

Specifies the port number that the rendering server uses.

Startup file

Specifies the startup file to be used for loading the initial configuration for Realsoft 3D.

Script

Specifies the startup script to be executed when Realsoft 3D is loaded.

Debug

If checked, Realsoft 3D reports all undefined internal attributes and methods. This can be useful to track down old plugins, for example.



Chapter 9. RsView

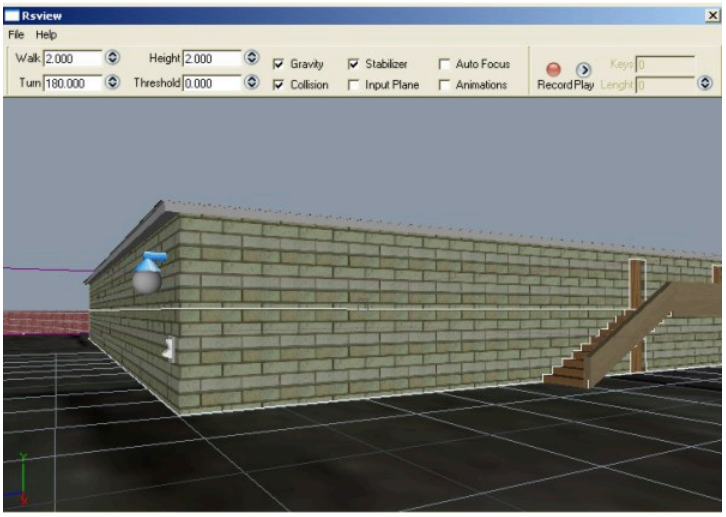
Scene Viewer Utility

Description

This program can be used for viewing .R3D project files. It allows the user to navigate in the loaded scene using the Walk-Thru tool. The utility uses collision detection for floor and wall detection, allowing the user to move in the scene easily.

The tool can also run animations included with the scene. For exmaple, one can create a simple key frame animation which opens a door. When the user looks at such a door in RsView and presses the space bar, the animation will be executed. When the user hits the space bar the second time, the animation will be run backwards and the door will close.

This program is freely distributable. One can create a .r3d project with Realsoft 3D license and send the project along with thsi utlitity to his/her customer, for example.



Pull-down Menus

File/Open	Open the scene to be explored.
File/Exit	Exit the program.
Channels	A scene to be explored.
Help/About	Open About dialog, showing the version number of the program.
Help/Manual	Open this reference page.

Startup Parameters

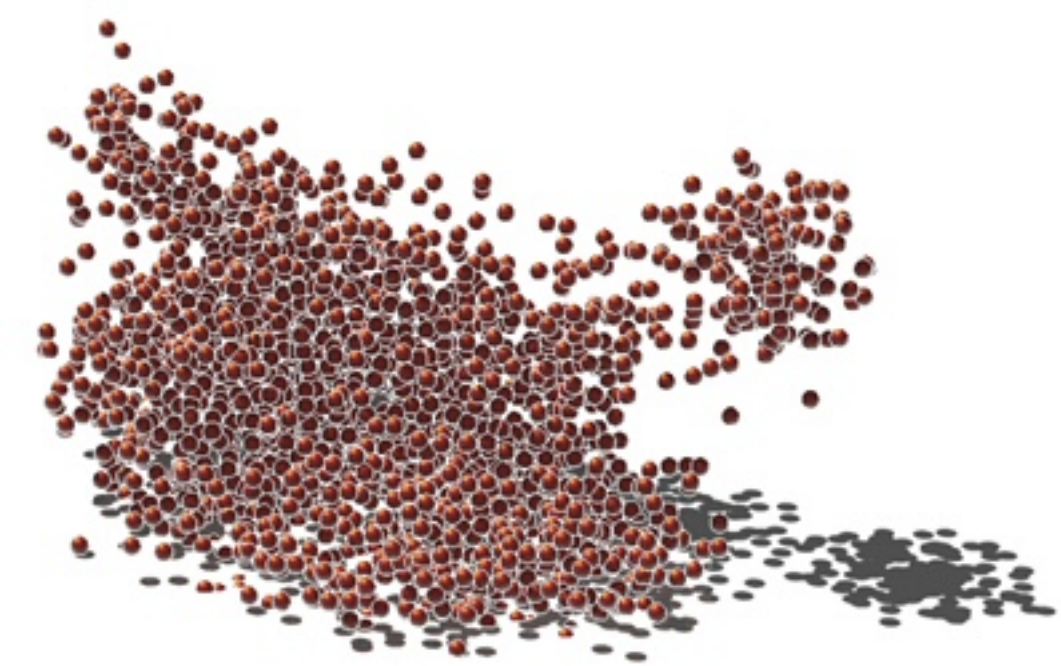
RsView can be called from a command line window with one or more project file names:

```
RsView myhouse.r3d mylightning.r3d ..
```

Projects are all loaded in and combined for browsing.

In addition to this, the program accepts the following startup options:

-file [filename]	Specifies the default name of the project to be loaded.
-plugins	Specifies custom plug-in folder from which the plug-in modules should be read. The default folder is 'Plugins' (in program's home folder).



Chapter 10. Plugins

Realsoft 3D™ plugins can be found in `plugins` sub folder.

Realsoft 3D™ and Real3D™ are properties of Realsoft Oy, and Realsoft Graphics Oy, Finland. All other trademarks are the property of their respective owners.

V8 default plugins

ac3dio	AC3D file format. AC3D is a 3D design tool, developed by Invis Limited.
AIplugin	Adobe Illustrator™ file format.
brickshdr	Brick material, for photo realistic renderer from Schmeling Consulting.
cellular	Cellular material, for photo realistic renderer from Schmeling Consulting.
Chrono	Chrono™ multi-body simuation toolset developed by Delta Knowledge.
demformat	USGS DEM file format. DEM is a geospatial file format developed by the United States Geological, for storing digital elevation models.
Dynatomic	Full featured soft-body simulation toolset from Dynadream.
gendeform	Geometry deformation toolset, for deforming geometric objects, such as NURBS and Sub-division geometrics.
IGESplugin	IGES 3D file format, for exchanging geometric data with CAD software.
md3exp	Quake 3 game file format.
md5io	File format for Doom 3 shooter game.
objformat	Wavefront .obj file format.

ppeline	Post effect for rendering particles as straight lines.
ppepoint	Post effect for rendering particles as round discs.
psdformat	Photoshop™ layered image file format from Adobe Systems.
r33dp	3D Painting toolset.
r33ds	Autodesk 3DS file format.
r3analytictools	Toolset for creating analytically defined geometric objects, such as quadric spheres.
r3avi	Microsoft's AVI audio and video file format.
r3blur	Image blur post rendering effect.
r3boujou	Boujou matchmoving file format.
r3carriage	Geometric modifier object for transforming any number of other objects.
r3cineon	Kodak's Cineon image file format.
r3constr	A plugin for for implementing scripting based intelligent geometric objects. Allows scripts, such as JavaScript™ to be used for controlling the behavior of other objects.
r3creator	Parametrized modifier object which can create new objects by duplicating existing ones.
r3deformtools	Toolset implementing various geometric deformation modifier objects.
r3detedg	Edge detection post rendering effect.
r3dither	Image dithering post rendering effect.
r3dxf	Autodesk DXF geometric file format.
r3ffmpeg	MPEG video and audio file format.
r3fieldev	Geometric modifier for evaluating any material defined field, such as color. The result can be used as an input for parametrized modeling, animations and such.
r3fileinst	Tool for instancing geometric objects residing on disk in rendering time. Allows very complex scenes to be rendered.
r3flc	Autodesk FLC video format.
r3fog	Fog post rendering effect.
r3fprint	Foot print animation toolset. Defines motion paths through the specified foot steps.
r3fretyp	Nurbs based font outline curve extraction tool set based on Freetype™ font engine. Converts text string to NURBS curves for 3D logo extrusion etc.
r3gif	GIF image and animation raster file format.
r3glow	Glow post rendering effect for the renderer.
r3gmanag	Geometric manager object for parametrized modeling and animation. Packs and aligns geometric objects.
r3gradient	Image gradient post rendering effect.
r3hook	General purpose plugin for 3rd party developers.

r3ica	Citrix's Icaros ICA file format.
r3ico	Microsoft's ICO raster image format.
r3iff	Amiga image file format (inport).
r3interpolator	Memory efficient procedural renderer generating large number of geometric objects by copying the sample object in rendering time. Can be used for modeling and rendering complex geometric objects, such as grass, hairs and fur.
r3invkin	Inverse kinematics modifier, for editing skeletons and other geometric objects in character animations.
r3jpeg	JPEG raster image format.
r3js	JavaScript interpreter for scripting.
r3jsio	JavaScript based file format for importing and exporting geometric objects. Allows scenes to be exported in JavaScript format.
r3ligspr	Light glow rendering post effect.
r3line	Post rendering effect for adding cartoon style outlines to images.
r3link	Geometric object that can refer to another geometric object. Works similar to 'symbolic link' in various disk operating systems. Used in parametrized modeling.
r3lookat	Modifier for rotating geometric objects based on the position of the given target object.
r3macmon	Macro monitor, for executing and viewing recorded macros.
r3map	Tools for mapping materials to geometric objects using parallel and spherical projections, for example.
r3measure	Measurement tool set implementing measurement line, angle and point tools.
r3metab	Metaball geometric object for creating organic freeform geometrics from points and analytic spheres.
r3ngc	G-code numerical control programming language used in computer-aided manufacturing to control automated machines.
r3nurbs	Non-Uniform Rational B-Spline geometry for creating smooth curves and surfaces of arbitrary degree.
r3nurbstools	Toolset for creating Non-Uniform Rational B-Spline geometrics.
r3openexr	Deep color raster image file format from Industrial Light and Magic.
r3opengl	OpenGL™ rendering canvas, for real time 3D rendering
r3partic	Toolset for creating 1d, 2d and 3d particles.
r3plant	Parametrized tool for trees, plants and branching fractal shapes.
r3png	Portable Network Graphics (PNG) file format.
r3pointdisplacer	Deformation modifier using pixel colors in given raster image to define the level of deformation.
r3ppm	PPM raster file format.

r3primplus	Miscellaneous tools for editing geometric primitives, such as points, cubes, and spheres.
r3primplus2	Miscellaneous tools for editing geometric primitives, such as points, cubes, and spheres.
r3ps	PostScript™ vector file format export.
r3qtime	QuickTime ™ video and audio format from Apple.
r3rz3	Realviz™ matchmoving format importer.
r3satura	Image saturation post rendering effect.
r3sdsplus	Various tools for manipulating Rational Subdivision Surface geometric objects.
r3sdstools	Tool for creating basic geometric primitives (spheres, cones, cubes etc.) using Rational Subdivision Surfaces.
r3selpainter	Selection tool based on paint brush style user interface.
r3simtool	Various simulation tools based on Newtonian physics, including gravity, collision detection etc.
r3simul	Explosion simulation tool.
r3sixdofmouse	Support for 3DConnexion ™ six degree of freedom input devices. Currently available for Windows only.
r3sunmoon	Tools for simulating lightning caused by real world sun and moon.
r3tblur	Image blur effect which blurs over a sequence of animation frames.
r3tga	Truevision TARGA image file format.
r3traject	Modifier for tracing object trajectories in animations and simulations.
r3tt	Font outline tool based on Microsoft's TrueType™ engine.
r3v3io	Real3D ™ version 3 object file format.
r3viewctrl	Toolset for managing 3D view properties, such as orientation, zoom and scale.
r3wall	Parametrized modeling tool for building houses.
r3warp	Toolset for creating various geometric deformation objects, including noise, bend, shrink wrapping etc.
r3wirecapture	Tool which captures outlines from the rendered view.
scratches	Material making the surfaces to look scratched in photo-realistic rendering.
tiffformat	TIFF tagged image file format from Adobe.
v3mat	Real3D™ version 3 material import.
VRMLplugin	VRML (Virtual Reality Modeling Language) file format import and export plugin from Softartist, Finland.



Index

Symbols

1D Particles, 38
2D Particles, 39
3D Measure, 30
3D Measure Tool, 217
3D Painting Circle Tool, 177
3D Painting Curve Tool, 177
3D Painting Polygon Tool, 179
3D Painting Rectangle Tool, 179
3D Painting Selected Tool, 179
3D Painting Tool, 176
3D Particles, 40

, 116, 214, 215

A

Add Curve Tool, 247
Advanced Selection Window, 296
Advanced VSL Controls, 89
Aeroplane Banking, 41
Aimpoint, 16
Airbrush Tool, 177
Ambient Light, 62
Ambient Light Tool, 226
Ambient Occlusion, 115
Ambient Tool, 226
Analytic Rectangle Tool, 183
Analytic Tools, 181
Analytic/Rectangle Tool, 183
Anchor, 85
Angle Measure, 9
Angle Measure Tool, 217
Animateable Attributes, 303
Animated Image Object, 88
Animated Texture, 107
Animation Object, 146
Animation Properties, 303
Animation Settings Window, 297
Animation Window, 298
Antialiasing, 140
Antialiasing Trigger Channels, 141
Assign View to Selected Objects, 218
Assign View Tool, 218
Atmosphere Object, 9
Atmosphere Tool, 204
Attributes Window, 300
Auto Map Tool, 216
Autofocus Tool, 219
Automap Tool, 216
Average Map, 55
Average Map Tool, 228
Average Mapping Tool, 228
Average of Channels, 113
Average of Sub Channels, 113
AVI Output, 121
Axis, 149

B

Backdrop Object, 10
Backdrop Tool, 205
Base, 15
Base Index, 107
Batch Rendering Utility, 352
BatchRen, 350
Begin Illumination, 101
Bend Constructor, 73
Bend Tool, 277
Bindings, 117
Bleed Option, 106
Blur Effect, 128
BMP Output, 124
Boolean Invert Tool, 188
Boolean Operation, 29
Boolean Tool, 188
Boundary Surface Solid, 29
Box Count, 143

Box Rendering, 7, 143
Branch, 10
Branch Tool, 255
Break Curve Tool, 250
Break Mesh Tool, 247
Building Object, 14
Building Tool, 189
Building Tools, 189
Built-in JavaScript Classes, 149
Bump Height, 107
Bump Map Channel, 107
Bump Map Wizard, 118
Bump Mapping, 106
Bump Smoothing, 107
Bump VSL Object, 106

C

Call VSL Object, 108
Camera Object, 15
Camera Tool, 205
Camera Tracking Tool, 220
Carriage Object, 75
Carriage Tool, 194
Caustics, 145
Caustics Map, 63
Cell VSL Object, 108
Center of Gravity, 68
Center of Gravity Tool, 274
Center Pivot Tool, 232
Central Glow, 125
Channel Object, 1
Channels Tab, 324
Choreography Classes, 2
Choreography Object, 17
Choreography Tool, 205
Choreography Window, 300
Chrome Wizard, 118
Cineon Output, 122
Circle, 21
Circular Trim Curve Tool, 237
Class Options, 312
Client Window, 293
Clip Map Wizard, 118
Close Curve Tool, 251
CloseU Tool, 248
CloseV Tool, 248
CoBind, 194
COG Tool, 274
Collision Accuracy, 70
Collision Detection, 70
Collision Tool, 274
Color of Surface, 98
Color Square, 293
Color Wheel, 304
Color Window , 293, 304
Color Wizard, 118
Comment VSL Object, 109
Common Properties, 97
Compass Menu, 333
Concatenate Curves Tool, 251
Concatenate Meshes Tool, 248
Conditional VSL Expressions, 110
Cone, 17, 185, 187
Constant VSL Object, 109
Constraint Object, 18
Constraint Tool, 205
Construction Stack, 2
Construction Tool, 193
Constructors, 73
Container Window, 293
Control Bar, 327
Coordinate System, 150
Copy VSL Object, 109
Crease Tool, 262
Create Subdivision Object Tool, 263
Creator Object, 18
Cube, 19
Cube Map, 55
Cube Map Tool, 229
Cube Mapping Tool, 229
Curve Editor, 305
Curve Gadget, 305
Curve Manager, 19
Curve Painting Tool, 177
Curve Quality, 72
Curve Types, 109
Curve VSL Object, 109
Custom Channel, 1
Custom Channel Object, 1
Cut Cone, 20
Cut Pyramid, 20
Cycle, 294
Cylinder, 20, 185, 187
Cylinder Map, 56
Cylinder Map Tool, 229
Cylinder Mapping Tool, 229

D

Default Map, 56
Default Map Tool, 229
Default Mapping Tool, 229
Define Input Plane by 3 Points, 219
Define Input Plane by Surface Normal, 219
DefineUV Tool, 248
Deformation Tool, 210
Delete Curve Points, 251
Delete Points from Mesh, 248
Delete Points Tool, 233
Density, 69
Density Wizard, 118
Depth of Field, 17
Depth Of Field Effect, 128

Derive VSL Object, 110
Destroy Faces Tool, 264
Detach Tool, 216
Digits of Texture Index, 108
Disable VSL object, 93
Disk Map, 57
Disk Map Tool, 230
Disk Mapping Tool, 230
Displace Tool, 241
Displacement, 36
Displacer, 74
Distant Gravity Object, 20
Distant Light, 61
Distant Light Tool, 226
Distribute Points Tool, 251
Distribute Tool, 222
Distributed Rendering, 143
Dithering, 142
Dithering Effect, 130
Drag Object, 75
Duplicate and Mirror Tool, 264
Duplicate Faces Tool, 264
Duplicate Selected Edges Tool, 264

E

Eaves, 43
Edge Filter, 130
Edge Handles, 48
Edge Rounding Tool, 264
Edit NURBS Mesh Selector, 248
Edit Objects Tool, 233
Elasticity, 68
Electric Charge, 68
Elevate Degree Tool, 251
ElevU Tool, 249
ElevV Tool, 249
Ellipse, 21
Ellipsoid, 21
Ellipsoid Segment, 21
Else VSL Object, 110
Emitter, 38
End Condition, 33
End Illumination, 101
Evaluate with Normal, 72
Explode Tool, 275
Extend Tool, 277
Extract Curves Tool, 248
Extrude Curve Tool, 251
Extrude NURBS Tool, 242
Extrusion Manager, 21

F

F-Stop, 16
Face Handles, 48
Faces Tool, 265
Fade Map Wizard, 118
Fading, 55
Falloff, 59
Fan, 22
Fan Tool, 275
Fetch View Tool, 219
FFMPEG Output, 122
Fidelity, 85
Field Evaluator, 22
File Instance Object, 23
File Instance Tool, 206
File Render Configuration, 5
File Rendering Configurations Tab, 325
File Rendering Window, 308
Fill Holes Tool, 265
Fillet Tool, 242, 249
Film Sensitivity, 16
Finite, 55
Fit Tool, 219
Flash Light, 16
Flatten Tool, 266
FLC Output, 122
Flip Faces Tool, 266
Float, 168
Float Window, 295
Fluid Dynamics, 70
Fluid Friction, 69
Fluid Pressure, 69
Fluid Symmetricity, 69
Fluid Tool, 275
Fluid Velocity, 69
Focal Lenghth, 16
Focus, 16
Fog Effect, 131
Fog Wizard, 119
Font Manager Object, 24
Font Tool, 243
Foot Tool, 216
Force to screen, 365
Forced Visibility, 73
Format String, 108
Fractal Dimension, 111
Fractal Plant, 46
Frame, 294
Frame Command, 8
FrameBlur, 132
Freehand Knife Tool, 266
Frequency, 147
Front View Tool, 219

G

General VSL Options, 92
Geometric Objects Tab, 314
Geometry Manager, 217, 291
GIF Output, 122
Glass Wizard, 119
Glow Effect, 132

Glow Wizard, 119
Gradient Curve, 110
Gradient in Texture Mapping, 116
Gravity, 70
Gravity Tool, 274, 276
Grid Draw Tool, 219
Grid Drawing, 87
Grid Object, 87
Grid Snap Tool, 220
Grid Snapping, 87
Grid Tool, 233
Grids Tab, 321
Group Object, 24
Group Tool, 207
GUI Windows, 292

H

Height Mapping, 107
Hide Geometry, 72
Hole Tool, 190
Hollow, 66
Hosts, 144
HPB Rotations, 66
Hyperboloid, 185
Hyperboloid Object, 24

I

I.K. Tool, 277
ICO Output, 122
Icons, 363, 365
If VSL Object, 110
Illumination Map Wizard, 119
Image Flare, 134
Image Flare Effect, 134
Image Gradient, 133
Image Object, 87
Image Processing, 105
Image Viewer, 360
Images Tab, 319
Impact Duration, 68
Impact Level, 68
In/Out Channels of VSL Object, 94
Indexing of rendered files, 7
Inertia, 68
Info Window, 308
Init UV Tool (SDS), 267
Initialize Ray, 103
Input Parameter, 302
Input Plane, 337
Input Plane Object, 25
Input Plane Tool, 233
Instance Object, 25
Instance Tool, 207
Integer, 172
Intensity of Light, 59
Interpolation of Bumps, 107
Interpolator Object, 25
Intersect Tool, 188
Intersection, 29
Inverse Kinematics Object, 75
Inverse Kinematics Tool, 195
Inversed Kinematics of Curve, 252
Invert Curve Tool, 252
Invert NURBS Mesh, 249
Invert Trim Side Tool, 254
IP by 3P, 219
IP by N, 219
Iso Value, 31

J

Join Faces Tool, 267
JPG Output, 123

K

Keyframe, 3

L

Language, 363, 364
Lathe Tool, 185
Lattice Mapping, 67, 222
Lattice Mapping Rotate Tool, 223
Lattice Mapping Tool, 222
Lattice Mapping/Rotate Tool, 226
Lattice Object, 76
Lattice Space, 67
Lattice Tool, 196
Leaf Object, 26
Leaf Tool, 255
Lens Flare Effect, 125
Level Material, 94
Level Object, 29
Level VSL Object, 111
Light Glow Effect, 134
Light Properties, 99
Light Source, 59
Light Source Tool, 226
Line Effect, 135
Line Tool, 178
Linear Interpolation of Channels, 113
Linear VSL Object, 111
Link Object, 29
Link Tool, 207
Load Window, 295
Look At Tool, 196
Loop Tool, 267

M

Macro Button, 292
Macro Slider, 292
Macros Tab, 324
Magnetism, 70
Magnetism Tool, 276

Map IK Tool, 216
Map Tool, 223
Map VSL Object, 111
Map2Obj, 230
Map2Obj Tool, 230
Mapped Shadows, 60
Mapping Object, 54
Mapping Tool, 228
Mass, 68
Material Browse Window, 308
Material Initialization, 96
Material Map, 54
Material Map Object, 54
Material Mapping Object, 54
Material Mapping Tool, 228
Material Preview, 89
Material Types, 88
Materials Tab, 317
Matrix, 157
Measure Curve Tool, 252
Measure Object, 30
Measure Tool, 217
Memory Usage, 144
Merge Objects Tool, 268
Mesh Map, 57
Mesh Object, 35
Metaball, 31
Metaball Tool, 207, 207
Metrics, 310
MipMapping, 117
Mirror Tool, 278
Moon, 51
Morphing Object, 76
Morphing Tool, 197
Move 1D, 210
Move 2D, 210
Move 3D, 211
Move Constructor, 77
Move Tool, 278
Move-to Tool, 278
Multi Map Tool, 223
Multi Reflection, 141

N

Native File Format, 123
Network Rendering Daemon, 353
New Point to Curve Tool, 252
Noise Choreography, 3
Noise Tool, 278
Noise VSL Object, 111
Normalize Channel, 92
Normalize Object Space, 234
Numeric Window, 309
NURBS Circle Tool, 237
NURBS Cone Tool, 238
NURBS Cube Tool, 238
NURBS Curve, 32
NURBS Curve Tool, 239
NURBS Cut Cone Tool, 239
NURBS Cut Pyramid Tool, 240
NURBS Cylinder Tool, 240
NURBS Disk Tool, 241
NURBS Helix Tool, 243
NURBS Mesh, 35
NURBS Pyramid Tool, 243
NURBS Rectangle Tool, 244
NURBS Rotate Tool, 244
NURBS Sphere Tool, 244
NURBS Surface, 35
NURBS Sweep Tool, 245

O

Object Browse Window, 310
Object Color, 63
Object Knife Tool, 269
Object Properties Window, 312
Object Scripts, 69
Object Space, 66
Octaves of Noise, 111
OpenEXR, 123
OpenGL Tool, 220
Operation VSL Object, 112
Options Window, 310
Orthogonal Particles, 234
Outline Rendering, 144
Output Channel Bindings, 121
Output Object, 121
Output Objects Tab, 320

P

Packer, 218, 292
Packer Object, 77
Packer Tool, 218
paint with Selected Tool, 179
Parallel / Perspective Tool, 219
Parallel Curve Tool, 252
Parallel Map, 57
Parallel Map Tool, 231
Parallel Mapping Tool, 231
Parametrization, 33
Parametrization Tool (Curve), 252
Particle Disks, 126
Particle Effect, 135
Particle Emitter Tool, 207, 208, 208, 208
Particle Filler Tool, 208, 209
Particle Lines, 127
Particle Tool, 209
Particles, 36
Paste Color on Objects, 178
Paste Colors Tool, 178
Paste Tool, 223
Path Level Constructor, 79

Path Level Object, 40
Path Tool, 225
Periodic, 147
Perspective Tool, 219
Phase, 147
photon map, 62, 145
photon map Tool, 227
Physical Properties, 68
Picker Tool, 178
Ping Pong, 107
Placer, 291
Plant, 46
Plant Seed, 46
Plant Tool, 255
PlantRoot, 10
Plugins, 368
PNG Output, 123
Point Light, 60
Point Light Tool, 228
Point Translator, 80
Point Translator Tool, 198
Polygon, 41
Polygon Painting Tool, 179
Polygon Reduction, 273
Polyhedron, 41
Polymid, 41
Pose, 2
Position Mark Object, 41
Post Image, 136
Post Image Effects Tab, 322
Post Particle Effects, 125
Post Particle Effects Tab, 323
Post Particles, 105
Post Process View Tool, 219
Post Processing, 142
PPM Output, 123
Priority, 55
Procedure, 94
Procedure VSL Object, 113
Projection, 15
Property Window, 312
Pyramid, 42
Pyramid Map, 58
Pyramid Map Tool, 231
Pyramid Mapping Tool, 231

Q

Quad View Tool, 220
Quadrics, 185, 187

R

r3auw, 362
Radial Deformator, 80
Random VSL Object, 114
Randomize Plant Tool, 255
Ray Direction, 98
Ray Hit, 104
Ray Termination, 104
Ray Tracing, 139
Raytrace VSL Object, 114
Realsoft Scene Viewer, 366
Rebound Energy, 68
Reconnect Tool, 226
Rectangle, 42
Rectangle Tool, 183
Rectangular Trim Curve Tool, 244
Recursion Depth, 139
Reduce Curve Degree, 253
ReduceU Tool, 249
ReduceV Tool, 249
Reflectivity of Surface, 98
Remove Tool, 188, 189
Render Box Tool, 218
Render Settings, 138
Render Settings Tab, 318
Render View Tool, 220
Rendering Daemon, 357
Rendering Diagram, 105
Rendering Engine, 139
RendW, 357
Repeating Texture Sequence, 107
Replace Instances, 233
Replicator, 218
Replicator Tool, 218
Reset Object Handles, 234
Reset Plant Tool, 256
Reset View Tool, 220
Resolution of file rendering, 7
Rigidity, 68
Roof Line Tool, 191
Roof Object, 42
Roof Tool, 191
Room Object, 44
Room Tool, 191
Root Tool, 255
Rotate 1D, 212
Rotate 2D, 212
Rotate 3D, 213
Rotate Constructor, 81
Rotate Curve Tool, 253
Rotate Extend Tool, 279
Rotate Tool, 278
Rotation, 66
Rotation Tool, 278
Rotator Object, 81
Rotator Tool, 199
Round Curve Tool, 253
Row Column, 291
RsView, 366
RT-Invisible Tool, 234

S

Safe Region, 17
Safety Area, 142
Saturation Effect, 138
Scale, 66
Scale 1D, 214
Scale Tool, 279
Scanline, 104
Scanline Rendering, 142
Scope, 54
Script Choreography, 3
Scripting Window, 313
SDS, 47
SDS Add Tool, 261
SDS Bevel Tool, 262
SDS Circle Tool, 256
SDS Cone Tool, 257
SDS Cube Tool, 257
SDS Cut Cone Tool, 258
SDS Cut Pyramid Tool, 259
SDS Cylinder Tool, 259
SDS Delete Tool, 263
SDS Extrude Tool, 265
SDS Hide/Show Tool, 267
SDS Invert Tool, 267
SDS Merge Tool, 268
SDS Pyramid Tool, 260
SDS Rectangle Tool, 260
SDS Smooth Tool, 269
SDS Sphere Tool, 261
SDS Tri/Quad Tool, 271
SDS Weld Tool, 273
Secondary Ray, 103
Select Tool, 235
Select Window, 313
Selection Painter, 235
Selective Visibility, 72
Self Illumination Wizard, 120
SelPainterTool, 235
Separate Tool, 269
Separator, 294
Set Start of Mesh, 249
Shaded OpenGL Tool, 220
Shaded Quality, 73
Shader, 96
Shader Tree, 90
Shader VSL Object, 115
Shadow Map VSL Object, 115
Shadows, 60
Sharpen Curve Point Tool, 253
Sharpen Mesh Tool, 250
Sharpness of Specularity, 116
ShowImg Utility, 360
Side View Tool, 220
Simulation Properties, 70
Size Tool, 279
Skeleton, 82
Skeleton Tool, 200
Skew, 67
Skew Tool, 280
Smoothen/Flatten Tool, 270
Snap Tool, 236
Soft Selections, 235, 236
Soften Selection, 236
Software Updater, 362
Solid Tool, 189
Special Light, 61
Special Light Tool, 228
Specular Color Wizard, 120
Specular VSL Object, 116
Sphere, 21, 185, 187
Sphere Map, 58
Sphere Map Tool, 232
Sphere Mapping Tool, 232
Spin, 68
Spin Tool, 276
Split Edge Tool, 270
Splitter, 295
Spot Light, 61
Spot Light Tool, 228
Start Tool (Curve), 253
Startopts, 364
Startup options, 147
Status Window, 327
Steel Wizard, 120
Stochastic Antialiasing, 140
Streaks, 126
Sub Flares, 125
Sub Wall Tool, 192
Subdivide Tool, 270
Subdivision Surface, 47
Sun, 51
SunMoon, 51
Surface Area, 30
Surface Filtering, 99
Surface Finishing, 102
Surface Friction, 68
Surface Geometry, 97
Surface Illumination, 101
Surface Properties, 98
Swap Curve Tool, 253
Swap Footprint Tool, 217
Swap Mesh Direction, 250
Swap UV of a Mesh, 250
Sweep Curve Tool, 253

T

Tab, 294
Tags, 70
Targa Output, 124
Texture Animation, 107
Texture Bindings, 117

Texture Bleeding, 106
Texture Indexing, 107
Texture Map Wizard, 120
Texture Mapping, 116
Texture Quality, 73
Texture Tiling, 106
Texture VSL Object, 116
Tiled Textures, 106
Time Curve, 147
Time Lines, 300
Time System, 311
To Face Material Tool, 273
To File Instance, 236
To SDS Tool, 271
To3D Tool, 254
ToBezier Tool (Curve), 254
ToCurves Tool, 245
ToNurbs Tool, 245
Tool Bar, 327
Tool Control Bar, 327
Tools, 176
Top View Tool, 220
Trace Distance, 115
Track Tool, 220
Transform to Material Space, 92
Transformation Tool, 277
Translation, 66
Transparency of Surface, 98
Transparency Wizard, 120
Triangles to Quads Tool, 272
Trim Curve, 52
Trim Curve Tool, 246
Trim On/Off, 254
Trim Tool, 246
Trim View Window, 328
Trim with a Curve Tool, 254
Triset Add Face, 281
Triset Auto Smooth, 281
Triset Delete Tool, 281
Triset Flat Tool, 282
Triset Flip Tool, 282
Triset Merge Tool, 282
Triset Object, 52
Triset Select Mode, 282
Triset Smooth Tool, 282
Triset Tools, 281
Tube Object, 53
Tube Tool, 187
Tunnel Tool, 272
Turbulence, 22
Twist Tool, 281

U

Undersampling, 140
Unified Circle Tool, 282
Unified Curve Tool, 286
Unified Rectangle Tool, 288
Unified Sphere Tool, 290
Unified Unified Creation Tools Tool, 282
Union Tool, 189
Unmap Tool, 226
Unwrap Tool, 272
Unwrap UV Tool, 272
User Defined Channel, 1
User Defined Color Channel, 2
User Defined Float Channel, 2
User Defined Vector Channel, 2
User Interface, 291
UV Editor, 328
UV Image Tool, 180
UV Property Window, 330
UV Set, 58
UV Set Connect Tool, 273
UV Set Coordinates From View, 273
UV Set Tool, 272
UVImage Painting Tool, 180
UVImage Tool, 180

V

V3 Material, 95
Variable VSL Object, 117
Variate Plant Tool, 256
Vector, 162
Vectorize Tool, 246
Velocity, 68
Velocity Tool, 276
Vertex Colors, 72
Vertex Painting Tool, 181
VF-Invisible Tool, 236
View Camera, 337
View Control Bar, 338
View Control Tools, 218
View Model, 146
View Models Tab, 326
View Popup Menu, 330
View Property Window, 341
View Window, 330
Visibility Options, 64
Volume Filtering, 100
Volume Illumination, 102
Volume Measure, 30
Volume Properties, 98
Volume Selector Tool, 236
Volume Shading, 102
Volumetric Light, 59
VSL antialiasing, 93
VSL Assignment Operation, 92
VSL Attributes, 90
VSL Effect, 138
VSL Input Channels, 94
VSL Material, 88
VSL Material Interface, 90

VSL Objects, 106
VSL Output Channels, 94
VSL Procedure Library, 94
VSL Samples, 95
VSL Shader Tree, 90
VSL Shaders, 96
VSL Wizards, 89, 118

W

Walk Thru Tool, 221
Wall Object, 53
Wave VSL Object, 117
Weight Curves, 304
Weight Tool (Curve), 254
Weight Tool (NURBS Mesh), 250
Weights, 303
Weld Tool, 250
Weld Tool (NURBS), 246
Weld Trim Curve Tool, 255
Window, 295
Window Output, 125
Windows, 296
Wireframe Properties, 71
Wizards, 89

X

XSect Tool, 247

Z

Zoom In/Out Tool, 222



Chrono

1 Edition

Chrono

Publication date 1985 ... 2019 (updated 2019-12-23 00:36:20)
Copyright © 1985 ... 2019 Alessandro Tasora

All rights reserved. This documentation contains proprietary information of Alessandro Tasora.

Chrono™ is a trademark Alessandro Tasora. All the other trademarks are the property of their respective owners.

Table of Contents

1. Chrono 1



Chapter 1. Chrono

Realsoft 3D version 8 includes Chrono™ - a full featured multi-body simulation system!

Chrono inserts new tab into the Realsoft 3D toolbar. The following Chrono manuals can be found in the `chrono` subfolder.

- Chrono User Manual [Chrono/docs/ch_manual_USER.pdf]
- Chrono Reference [Chrono/docs/ch_manual_REFERENCE.pdf]
- Chrono Scripting [Chrono/docs/ch_manual_SCRIPTING.pdf]



Dynatomic

1 Edition

Dynatomic

Publication date 1985 ... 2019 (updated 2019-12-23 00:36:20)
Copyright © 1985 ... 2019 Dynadream

All rights reserved. This documentation contains proprietary confidential information and trade secrets of Carlo Lanzotti.

Dynatomic™ is a trademark of Dynadream and Carlo Lanzotti. All the other trademarks are the property of their respective owners.

```
DynAtomic Software License

Except where otherwise noted, all of the documentation and
Software as well as media, images and any other file included
in the distribution is copyrighted by Carlo Lanzotti (the Author).

Copyright (C) Carlo Lanzotti 2002-2018. All rights reserved.

License

Use of the Software is permitted provided
that all of the following terms are accepted:

1. The Author grants you a non-exclusive license to use the
    Software subject to your compliance with all of the terms
    and conditions of this License.

2. You are only granted a license for the machine-readable,
    object code portion of the Software. You will not modify,
    enhance, reverse engineer or otherwise alter the Software
    from its current state.

3. You will not have any proprietary rights of the Software.
    You acknowledge and agree that the Author retains all
    copyrights and other proprietary rights of the Software.

4. The Software is provided "as-is," without any express or
    implied warranty. Without even the implied warranty of
    merchantability and fitness for a particular purpose. In
    no event shall the Author be held liable for any, direct or
    indirect, damages arising from the use of the Software.

5. None of the Software's files may be redistributed.

6. Use of the Software within the scope of this license is
    free of charge  and no royalty or licensing fees shall be
    payable by you. Use beyond the scope of this license
    shall constitute copyright infringement.

7. This license shall be effective and bind you upon your
    downloading of the Software.

8. The Author reserves his rights to modify this agreement
      in the future.

If you do not agree to all of the above terms, you are not
permitted to use the Software in any way, and all copies of
it must be deleted from your system(s).

Carlo Lanzotti - DynaDream
Web: http://www.dynadream.com/
E-Mail: info@dynadream.com
```

Table of Contents

1. Dynatomic 1

Chapter 1. Dynatomic

Realsoft 3D version 8 includes Dynatomic™ - a full featured Simulation System!

With DynAtomic you can

- Create high realistic dynamic simulation of soft-semirigid interacting body
- Create high realistic dynamic simulation of clothes and garments.

Objects in DynAtomic™ can interact togheters be they colliding, attached to others with constraints or mix-controlled by keyframed animation and by interaction of the most common fields like with Wind, Drag, Earth like Gravity, Newton and Magnetic forces.

Spacing from automatic mass distribution, 3D attributes painting, Cache playback and with an incredible amount of many other features, with DynAtomic the only limit is your creativity.

DynAtomic inserts new tab into the Realsoft 3D toolbar. All the tools support context sensitive help - just press F1 on any of the tools to open the manual page of the tool in question. You can find the documentation in the `dynatomic` subfolder.

Dynatomic Manual [dynatomic/index.html]



Revolution

1 Edition

Revolution

Publication date 1985 ... 2019 (updated 2019-12-23 00:36:20)
Copyright © 1985 ... 2019 Alessandro Tasora

All rights reserved. This documentation contains proprietary information of Alessandro Tasora.

Revolution4D™ is a trademark Alessandro Tasora, Italy. All the other trademarks are the property of their respective owners.

Revolution4D is licensed under BSD.

Table of Contents

1. Revolution 1



Chapter 1. Revolution

Realsoft 3D version 8 includes Revolution4D™ - modeling toolset.



Legal Notices

Second Edition

Legal Notices

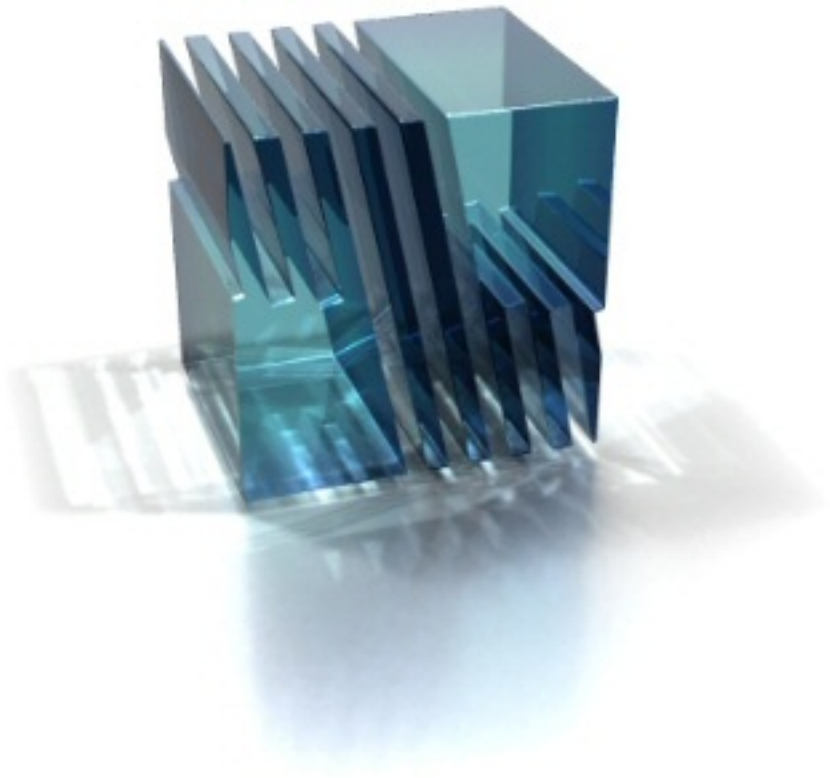
Publication date 1985 ... 2019
Copyright © 1985 ... 2019 Realsoft Graphics Oy

All rights reserved. This documentation contains proprietary confidential information and trade secrets of Realsoft Graphics Oy.

THIS SOFTWARE AND RELATED MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN NO EVENT SHALL REALSOFT GRAPHICS OY BE LIABLE FOR ANY DAMAGES CAUSED USE OF THIS MATERIAL.

Table of Contents

1. Imprint 1
2. End User Licenses 2
    Realsoft 3D SOFTWARE LICENSE AGREEMENT 2
    DynAtomic Software License 2
    Chrono End User License 3
3. Third party software 4
    ImageMagick License 4
    GD Graphics Library 6
    LIBPNG LICENSE 7
    FFmpeg Licence 9
    FreeType Licence 9
    Open CASCADE Technology 11
    SpiderMonkey JavaScript Engine 11
    BugTrap License 12
    Program Icons 12
    CURL LICENSE 12
Index 13



Chapter 1. Imprint

Realsoft Graphics Oy
Polvikatu 3 C 22, 33100 Tampere, Finland
http://www.realsoft.com

Realsoft 3D User Manual 7 1985 ... 2019, 2019-12-23 00:36:21. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without permission in writing from Realsoft Graphics Oy.

All images are copyrighted by their respective owners. All rights reserved.

Front Knife image Copyright (BT)-grafik
Learning the Basics Copyright (BT)-grafik
Modeling Copyright (BT)-grafik
Subdivision Surfaces Copyright Gunnar Radeloff
NURBS Copyright Gunter Hagedorn
Modeling Buildings Copyright Pixel Perfect
Materials Copyright (BT)-grafik
Rendering Copyright Gunnar Radeloff
Post Effects Copyright Gunnar Radeloff
Animations Copyright Gunnar Radeloff
Simulations Copyright (BT)-grafik
User Interface Copyright (BT)-grafik
Scripting Languages Copyright Martin Weber
Miscellaneous Topics Copyright (BT)-grafik
Design Copyright (BT)-grafik

Chapter 2. End User Licenses

Realsoft 3D SOFTWARE LICENSE AGREEMENT

This is a contract between you and Realsoft Graphics Oy. By installing this package you accept all the terms and conditions of this agreement. If you do not agree with the terms and conditions of this agreement, cancel the installation and return the software package to the location where you obtained it.

1. Copyright: The program and its related documentation are intellectual property of Realsoft Graphics Oy and is protected by international copyright laws. You may not use, copy, modify or transfer the program or documentation, or any copy except as expressly provided in this agreement.

2. License: You have the non-exclusive right to use the program on one computer at a time. You may install the program to more than one computer provided that only one person is using the program at a time. You may use several program installations simultaneously for image rendering purposes.

3. Disclaimer: The program is provided "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality, performance and result of the program is with you. Should the program prove defective, you assume the entire cost of all necessary servicing, repair or correction.

4. Terms: This license is effective until terminated. You may terminate it by destroying the program and documentation and all copies thereof. This license will also terminate if you fail to comply with any term or condition of this agreement. You agree upon such termination to destroy all copies of the program and documentation.

5. This license agreement shall be governed by the laws of Finland.

All rights reserved. This documentation contains proprietary confidential information and trade secrets of Realsoft Graphics Oy.

Realsoft 3D™ and Real3D™ are trademarks of Realsoft Graphics Oy and Realsoft Oy. All the other trademarks are the property of their respective owners.

THIS SOFTWARE AND RELATED MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality, performance and result of the program is with you. Should the program prove defective, you assume the entire cost of all necessary servicing, repair or correction. IN NO EVENT SHALL Realsoft Graphics Oy BE LIABLE FOR ANY DAMAGES CAUSED USE OF THIS MATERIAL.

DynAtomic Software License

Except where otherwise noted, all of the documentation and Software as well as media, images and any other file included in the distribution is copyrighted by Carlo Lanzotti (the Author).

Copyright (C) Carlo Lanzotti 2002-2017 c. All rights reserved.

License

Use of the Software is permitted provided
that all of the following terms are accepted:

1. The Author grants you a non-exclusive license to use the Software subject to your compliance with all of the terms and conditions of this License.

2. You are only granted a license for the machine-readable, object code portion of the Software. You will not modify, enhance, reverse engineer or otherwise alter the Software from its current state.

3. You will not have any proprietary rights of the Software. You acknowledge and agree that the Author retains all copyrights and other proprietary rights of the Software.

4. The Software is provided "as-is," without any express or implied warranty. Without even the implied warranty of merchantability and fitness for a particular purpose. In no event shall the Author be held liable for any, direct or indirect, damages arising from the use of the Software.

5. None of the Software's files may be redistributed.

6. Use of the Software within the scope of this license is free of charge and no royalty or licensing fees shall be payable by you. Use beyond the scope of this license shall constitute copyright infringement.

7. This license shall be effective and bind you upon your downloading of the Software.

8. The Author reserves his rights to modify this agreement in the future.

If you do not agree to all of the above terms, you are not permitted to use the Software in any way, and all copies of it must be deleted from your system(s).

Carlo Lanzotti - DynaDream
Web: http://www.dynadream.com/
E-Mail: info@dynadream.com

Chrono End User License

Chrono is open-source Multi-physics Simulation Engine, hosted at Github and governed by BSD license.

Except where otherwise noted, all of the documentation and Software as well as media, images and any other file included in the distribution is copyrighted by Alessandro Tasora, Italy.

Copyright (C) Alessandro Tasora. All rights reserved.

Chapter 3. Third party software

ImageMagick License

Realsoft 3D ™ `r3imagemagick` plugin uses ImageMagick 3rd party software to support various image formats. ImageMagic license follows:

Terms and Conditions for Use, Reproduction, and Distribution

The legally binding and authoritative terms and conditions for use, reproduction, and distribution of ImageMagick follow:

Copyright © 1999-2017 ImageMagick Studio LLC, a non-profit organization dedicated to making software imaging solutions freely available.

1. Definitions

License shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

Licensor shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

Legal Entity shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, control means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

You (or Your) shall mean an individual or Legal Entity exercising permissions granted by this License.

Source form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

Object form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

Work shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

Contribution shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as Not a Contribution.

Contributor shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

a. You must give any other recipients of the Work or Derivative Works a copy of this License; and

b. You must cause any modified files to carry prominent notices stating that You changed the files; and

c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

d. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

GD Graphics Library

Realsoft 3D **r3gif** plugin library implements GIF format using GD Graphics Library.

GD GRAPHICS LIBRARY COPYRIGHT STATEMENT FOLLOWS THIS LINE

Portions copyright 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004 by Cold Spring Harbor Laboratory. Funded under Grant P41-RR02188 by the National Institutes of Health. Portions copyright 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004 by Boutell.Com, Inc.

Portions relating to GD2 format copyright 1999, 2000, 2001, 2002, 2003, 2004 Philip Warner.

Portions relating to PNG copyright 1999, 2000, 2001, 2002, 2003, 2004 Greg Roelofs.

Portions relating to gdttf.c copyright 1999, 2000, 2001, 2002, 2003, 2004 John Ellson (ellson@graphviz.org).

Portions relating to gdft.c copyright 2001, 2002, 2003, 2004 John Ellson (ellson@graphviz.org).

Portions relating to JPEG and to color quantization copyright 2000, 2001, 2002, 2003, 2004, Doug Becker and copyright (C) 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004 Thomas G. Lane. This software is based in part on the work of the Independent JPEG Group. See the file README-JPEG.TXT for more information.

Portions relating to GIF compression copyright 1989 by Jef Poskanzer and David Rowley, with modifications for thread safety by Thomas Boutell.

Portions relating to GIF decompression copyright 1990, 1991, 1993 by David Koblas, with modifications for thread safety by Thomas Boutell.

Portions relating to WBMP copyright 2000, 2001, 2002, 2003, 2004 Maurice Szmurlo and Johan Van den Brande.

Portions relating to GIF animations copyright 2004 Jaakko Hyvätti (jaakko.hyvatti@iki.fi)

Permission has been granted to copy, distribute and modify gd in any context without fee, including a commercial application, provided that this notice is present in user-accessible supporting documentation.

This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for the authors of gd, not to interfere with your productive use of gd. If you have questions, ask. "Derived works" includes all programs that utilize the library. Credit must be given in user-accessible documentation.

This software is provided "AS IS." The copyright holders disclaim all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to this code and accompanying documentation.

Although their code does not appear in the current release, the authors also wish to thank Hutchison Avenue Software Corporation for their prior contributions.

END OF GD COPYRIGHT STATEMENT

LIBPNG LICENSE

Realsoft 3D **r3png** plugin library implements PNG (Portable Network Graphics) image format using GD Graphics Library.

LIBPNG COPYRIGHT STATEMENT FOLLOWS THIS LINE

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE:

```
If you modify libpng you may insert additional notices immediately following
this sentence.

This code is released under the libpng license.

libpng versions 1.0.7, July 1, 2000 through 1.6.28, January 5, 2017 are
Copyright (c) 2000-2002, 2004, 2006-2017 Glenn Randers-Pehrson, are
derived from libpng-1.0.6, and are distributed according to the same
disclaimer and license as libpng-1.0.6 with the following individuals
added to the list of Contributing Authors:

   Simon-Pierre Cadieux
   Eric S. Raymond
   Mans Rullgard
   Cosmin Truta
   Gilles Vollant
   James Yu
   Mandar Sahastrabuddhe

and with the following additions to the disclaimer:

   There is no warranty against interference with your enjoyment of the
   library or against infringement.  There is no warranty that our
   efforts or the library will fulfill any of your particular purposes
   or needs.  This library is provided with all faults, and the entire
   risk of satisfactory quality, performance, accuracy, and effort is with
   the user.

Some files in the "contrib" directory and some configure-generated
files that are distributed with libpng have other copyright owners and
are released under other open source licenses.

libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are
Copyright (c) 1998-2000 Glenn Randers-Pehrson, are derived from
libpng-0.96, and are distributed according to the same disclaimer and
license as libpng-0.96, with the following individuals added to the list
of Contributing Authors:

   Tom Lane
   Glenn Randers-Pehrson
   Willem van Schaik

libpng versions 0.89, June 1996, through 0.96, May 1997, are
Copyright (c) 1996-1997 Andreas Dilger, are derived from libpng-0.88,
and are distributed according to the same disclaimer and license as
libpng-0.88, with the following individuals added to the list of
```

```
Contributing Authors:

   John Bowler
   Kevin Bracey
   Sam Bushell
   Magnus Holmgren
   Greg Roelofs
   Tom Tanner

Some files in the "scripts" directory have other copyright owners
but are released under this license.

libpng versions 0.5, May 1995, through 0.88, January 1996, are
Copyright (c) 1995-1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, "Contributing Authors"
is defined as the following set of individuals:

   Andreas Dilger
   Dave Martindale
   Guy Eric Schalnat
   Paul Schmidt
   Tim Wegner

The PNG Reference Library is supplied "AS IS".  The Contributing Authors
and Group 42, Inc. disclaim all warranties, expressed or implied,
including, without limitation, the warranties of merchantability and of
fitness for any purpose.  The Contributing Authors and Group 42, Inc.
assume no liability for direct, indirect, incidental, special, exemplary,
or consequential damages, which may result from the use of the PNG
Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this
source code, or portions hereof, for any purpose, without fee, subject
to the following restrictions:

  1. The origin of this source code must not be misrepresented.

  2. Altered versions must be plainly marked as such and must not
     be misrepresented as being the original source.

  3. This Copyright notice may not be removed or altered from any
     source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without
fee, and encourage the use of this source code as a component to
supporting the PNG file format in commercial products.  If you use this
source code in a product, acknowledgment is not required but would be
appreciated.

END OF COPYRIGHT NOTICE, DISCLAIMER, and LICENSE.

TRADEMARK:

The name "libpng" has not been registered by the Copyright owner
as a trademark in any jurisdiction.  However, because libpng has
```

```
been distributed and maintained world-wide, continually since 1995,
the Copyright owner claims "common-law trademark protection" in any
jurisdiction where common-law trademark is recognized.

OSI CERTIFICATION:

Libpng is OSI Certified Open Source Software.  OSI Certified Open Source is
a certification mark of the Open Source Initiative. OSI has not addressed
the additional disclaimers inserted at version 1.0.7.

EXPORT CONTROL:

The Copyright owner believes that the Export Control Classification
Number (ECCN) for libpng is EAR99, which means not subject to export
controls or International Traffic in Arms Regulations (ITAR) because
it is open source, publicly available software, that does not contain
any encryption software.  See the EAR, paragraphs 734.3(b)(3) and
734.7(b).

Glenn Randers-Pehrson
glennrp at users.sourceforge.net
January 5, 2017
```

FFmpeg Licence

Realsoft 3D **r3ffmpeg** plugin module uses libraries from the FFmpeg project under the LGPLv2.1.

Realsoft 3D™ is a trademark of Realsoft Graphics Oy and Realsoft Oy. FFmpeg™ is a trademark of Fabrice Bellard, originator of the FFmpeg project.

As described in the FFmpeg's https://www.ffmpeg.org/legal.html / License Compliance Checklist

Realsoft 3D uses FFmpeg libraries built without "--enable-gpl" and without "--enable-nonfree" configuration option and uses only LGPL related features of FFmpeg.

Dynamic linking for linking with FFmpeg libraries is used.

FreeType Licence

Portions of Realsoft 3D r3freetype plugin are copyright © 1985 ... 2019 The FreeType Project (www.freetype.org). All rights reserved.

```
Legal Terms
===========

0. Definitions
--------------

  Throughout this license,  the terms `package', `FreeType Project',
  and  `FreeType  archive' refer  to  the  set  of files  originally
  distributed  by the  authors  (David Turner,  Robert Wilhelm,  and
  Werner Lemberg) as the `FreeType Project', be they named as alpha,
  beta or final release.

  `You' refers to  the licensee, or person using  the project, where
```

`using' is a generic term including compiling the project's source code as well as linking it to form a `program' or `executable'. This program is referred to as `a program using the FreeType engine'.

This license applies to all files distributed in the original FreeType Project, including all source code, binaries and documentation, unless otherwise stated in the file in its original, unmodified form as distributed in the original archive. If you are unsure whether or not a particular file is covered by this license, you must contact us to verify this.

The FreeType Project is copyright (C) 1996-2000 by David Turner, Robert Wilhelm, and Werner Lemberg. All rights reserved except as specified below.

1. No Warranty

THE FREETYPE PROJECT IS PROVIDED `AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL ANY OF THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY DAMAGES CAUSED BY THE USE OR THE INABILITY TO USE, OF THE FREETYPE PROJECT.

2. Redistribution

This license grants a worldwide, royalty-free, perpetual and irrevocable right and license to use, execute, perform, compile, display, copy, create derivative works of, distribute and sublicense the FreeType Project (in both source and object code forms) and derivative works thereof for any purpose; and to authorize others to exercise some or all of the rights granted herein, subject to the following conditions:

- Redistribution of source code must retain this license file (`FTL.TXT') unaltered; any additions, deletions or changes to the original files must be clearly indicated in accompanying documentation. The copyright notices of the unaltered, original files must be preserved in all copies of source files.

- Redistribution in binary form must provide a disclaimer that states that the software is based in part of the work of the FreeType Team, in the distribution documentation. We also encourage you to put an URL to the FreeType web page in your documentation, though this isn't mandatory.

These conditions apply to any software derived from or based on the FreeType Project, not just the unmodified files. If you use our work, you must acknowledge us. However, no fee need be paid to us.

3. Advertising

```
--------------

  Neither the  FreeType authors and  contributors nor you  shall use
  the name of the  other for commercial, advertising, or promotional
  purposes without specific prior written permission.

  We suggest,  but do not require, that  you use one or  more of the
  following phrases to refer  to this software in your documentation
  or advertising  materials: `FreeType Project',  `FreeType Engine',
  `FreeType library', or `FreeType Distribution'.

  As  you have  not signed  this license,  you are  not  required to
  accept  it.   However,  as  the FreeType  Project  is  copyrighted
  material, only  this license, or  another one contracted  with the
  authors, grants you  the right to use, distribute,  and modify it.
  Therefore,  by  using,  distributing,  or modifying  the  FreeType
  Project, you indicate that you understand and accept all the terms
  of this license.

4. Contacts
-----------

  There are two mailing lists related to FreeType:

    o freetype@nongnu.org

      Discusses general use and applications of FreeType, as well as
      future and  wanted additions to the  library and distribution.
      If  you are looking  for support,  start in  this list  if you
      haven't found anything to help you in the documentation.

    o freetype-devel@nongnu.org

      Discusses bugs,  as well  as engine internals,  design issues,
      specific licenses, porting, etc.

  Our home page can be found at

    http://www.freetype.org
```

Open CASCADE Technology

Realsoft 3D includes **Revolution4D** plugin, which is built on Open CASCADE Technology software version 7.1.

Realsoft 3D ™ is a trademark of Realsoft Graphics Oy, Finland.

Revolution4D™ is a trademark of Alessandro Tasora, Italy. The software is governed by BSD license.

Open CASCADE Technology software is governed by GNU Lesser General Public License (LGPL) version 2.1 with additional exception. The license can be found in opencascade.com/content/licensing.

SpiderMonkey JavaScript Engine

Realsoft 3D **JavaScriptPlugin** library (filename `r3js`) links against **SpiderMonkey** JavaScript engine.

The **JavaScriptPlugin** is a property of Realsoft Graphics Oy. **SpiderMonkey** is Mozilla's JavaScript engine written in C/C++. It is used in various Mozilla products, including Firefox, and is available under the Mozilla Public Lisense 2.0 (MPL2) https://www.mozilla.org/en-US/MPL/2.0/

JavaScriptPlugin is also available in Source Code Form, as described in Section 3.1 of MPL2.0 license. The source code is included with Realsoft 3D binary distribution package, located in `scripts/js` folder. You can also obtain the copy of the source code from Realsoft Graphics Oy `<info@realsoft.com>`.

BugTrap License

Windows version of Realsoft 3D uses **BugTrap** crash reporting software. BugTrap is a MIT license based free software originally developed by Maksim Pyatkovskiy.

Program Icons

Realsoft 3D icons ships with two icons sets **pxl8** and **svg** designed by Andy Jones. These are © copyrighted 1985 ... 2019 by Andy Jones and Realsoft Graphics Oy. All Rights Reserved.

CURL LICENSE

Realsoft 3D **Software Updater** application uses `libcurl` 3rd party software.

CURL COPYRIGHT STATEMENT FOLLOWS THIS LINE

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE:

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2017, Daniel Stenberg, daniel@haxx.se, and many contributors, see the THANKS file.

All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.



Index

Symbols