# Blender Documentation Volume I - User Guide Last modified December 07 2004 S68

**Claudio Andaur** Manuel Bastioni **Baldassarre Cesarano Alejandro Conty Estevez** Karsten Dambekalns Florian Findeiss **Alex Heizer** Tim van Howe Wouter van Heyst Joeri Kassenaar Martin Kleppman Lyubomir Kovacev **Johnny Matthews Reevan McKay** Kent Mein Martin Middleton **Eric Oberlander Jason Oppel** Willem-Paul van Overbruggen Ton Roosendaal **Bastian Samela Stefano Selleri** Kenneth Styrberg Bart Veldhuizen **Chris Williamson Carsten Wartmann** 

Blender Documentation Volume I - User Guide: Last modified December 07 2004 S68 by Claudio Andaur, Manuel Bastioni, Baldassarre Cesarano, Alejandro Conty Estevez, Karsten Dambekalns, Florian Findeiss, Alex Heizer, Tim van Howe, Wouter van Heyst, Joeri Kassenaar, Martin Kleppman, Lyubomir Kovacev, Johnny Matthews, Reevan McKay, Kent Mein, Martin Middleton, Eric Oberlander, Jason Oppel, Willem-Paul van Overbruggen, Ton Roosendaal, Bastian Samela, Stefano Selleri, Kenneth Styrberg, Bart Veldhuizen, Chris Williamson, and Carsten Wartmann

Copyright © 2002 by Stichting Blender Foundation. Permission is granted to copy, distribute and/or modify this document under the terms of the Open Content License. A copy of the license is included in the appendix.

This is the Blender Documentation project official Guide. Feel free to add or modify your changes and send them clearly marked to the Blender documentation board (bf-docboard@blender.org).

# **Table of Contents**

I. Introduction to Blender	1
1. Introduction	1
What is Blender?	1
Blender's History	1
About Free Software and the GPL	3
Getting support - the Blender community	3
2. Installation	5
Downloading and installing the binary distribution	5
Building Blender from source	9
3. Understanding the interface	13
Blender's Interface Concept	13
Navigating in 3D Space	22
The vital functions	29
4. Your first animation in 30 + 30 minutes	33
Warming up	33
Building the body	34
Let's see what Gus looks like	43
Materials and Textures	49
Rigging	
Skinning	60
Posing	65
Gus walks!	69
II. Modelling, Materials and Lights	71
5. ObjectMode	71
Selecting objects	71
Moving (translating) objects	71
Rotating objects	72
Scaling/mirroring objects	73
Transform Properties Panel	74
Duplicate	74
Parenting (Grouping)	75
Tracking	76
Other Actions	77
Boolean operations	77
6. Basic Mesh Modelling	81
Basic Objects	81
EditMode	82
Smoothing	87
Extrude	91
Spin and SpinDup	97
Screw	105
Warp Tool	106
Object Hooks	108
7. Advanced Mesh Modelling	111
Catmull-Clark Subdivision Surfaces (-)	111
Weighted creases for subdivision surfaces	118
Edge lools	120
Bevelling Tools	124
Symmetrical Modelling	124
Proportional Editing 1001	12/
INOISE Decimator Tool	130
Declinator 1001	131 12E
0. Ivieta Objects	133
7. CUIVES AIRI JUIIALES	13/ 127
Surfaces	137 1/16
Text	148
1CAL	

	150
Curve Taper	154
Curve Deform	156
Skinning	159
10. Materials and Textures	163
Diffusion	163
Specular Reflection	164
Materials in practice	166
Ramp Shaders (-)	170
Raytracing Reflections (-)	170
Raytraced Transparencies (-)	170
Multiple Materials	170
Special Materials	173
11. Textures	177
Textures	177
Environment Maps	186
Displacement Maps	191
Solid and Hollow Glass	195
UV Editor and FaceSelect	198
Unwrapping Suzanne	
Texture Plugins	
12. Lighting	
Introduction	
Lamp Types (-)	
Ray Shadows	
Buffer Shadows	
Volumetric Light	
Tweaking Light	
13. The World and The Universe	
The World Background	
Exposure (-)	254
Mist	254
Mist Stars	
Mist Stars Ambient Occlusion	254 254 256 257
Mist Stars Ambient Occlusion	
Mist	254 254 256 257 263 263 263 263 263 264 268 269 270 270 277 277 277 281 288 288
Mist	
Mist	254 254 256 257 263 263 263 263 263 264 268 269 270 270 277 277 277 277 281 288 291 291
Mist	
MistStars StarsAmbient Occlusion	254 
MistStars Ambient Occlusion <b>III. Animation</b> 14. Animation of Undeformed Objects IPO Block Key Frames The IPO Curves and IPO Keys Other applications of IPO Curves The Time Ipo Path Animation 15. Animation of Deformations Absolute Vertex Keys Relative Vertex Keys Relative VertexKeys Lattice Animation 16. Character Animation Introduction: Lights, Camera and ACTION ! General Tools The Armature Object Skinning Bacara de	254 
Mist	254 
Mist Stars Ambient Occlusion III. Animation 14. Animation of Undeformed Objects IPO Block Key Frames The IPO Curves IPO Curves and IPO Keys Other applications of IPO Curves The Time Ipo. Path Animation 15. Animation of Deformations Absolute Vertex Keys Relative Vertex Keys Relative VertexKeys Lattice Animation 16. Character Animation Introduction: Lights, Camera and ACTION ! General Tools. The Armature Object. Skinning Posemode Action Window	254 
MistStars Ambient Occlusion III. Animation	
Mist Stars Ambient Occlusion III. Animation 14. Animation of Undeformed Objects IPO Block Key Frames The IPO Curves IPO Curves and IPO Keys. Other applications of IPO Curves The Time Ipo Path Animation 15. Animation of Deformations Absolute Vertex Keys Relative Vertex Keys Lattice Animation 16. Character Animation Introduction: Lights, Camera and ACTION ! General Tools	254 
Mist Stars Ambient Occlusion III. Animation 14. Animation of Undeformed Objects IPO Block Key Frames The IPO Curves IPO Curves and IPO Keys Other applications of IPO Curves The Time Ipo Path Animation 15. Animation of Deformations Absolute Vertex Keys Relative Vertex Keys Lattice Animation 16. Character Animation Introduction: Lights, Camera and ACTION ! General Tools The Armature Object Skinning Posemode Action Window Non Linear Animation Constraints Rigging a Hand and a Foot	254 
Mist Stars Ambient Occlusion III. Animation of Undeformed Objects IPO Block Key Frames The IPO Curves IPO Curves and IPO Keys Other applications of IPO Curves The Time Ipo Path Animation 15. Animation of Deformations Absolute Vertex Keys Relative Vertex Keys Relative Vertex Keys Lattice Animation 16. Character Animation Introduction: Lights, Camera and ACTION ! General Tools The Armature Object Skinning Posemode Action Window Non Linear Animation Rigging a Hand and a Foot Rigging Mechanics	254 

IV. Rendering	339
17. Rendering	
Rendering by Parts	340
Panoramic renderings	341
Antialiasing	
Output formats	
Rendering Animations	
Motion Blur	
Depth of Field	
Cartoon Edges	
The Uninea Kenderer	
18 Padiosity	
The Blender Radiosity method	
Radiosity Rendering	363
Radiosity as a Modelling Tool	
Radiosity Juicy example	
19. Ravtracing	
V Advanced Tools	383
20 Douticles	202
20. Farticles.	
Simple Particles	
Particle Interaction	300
21 Other Effects	403
Introduction	
Build Effect	
Wave Effect	
22. Special modelling techniques	407
Introduction	407
DupliVerts	407
DupliFrames	417
Modelling with lattices	429
23. Volumetric Effects	439
24. Sequence Editor	
Learning the Sequence Editor	
Sound Sequence Editor	
Sequence Editor Plugins	471
VI. Extending Blender	473
25. Python Scripting	473
A working Python example	475
Python Reference	480
Python Scripts	
26. Blender's Plugins System	
Writing a Texture Plugin	
Specification:	
Generic Texture Plugin:	
Compiling	403 196
Writing a Societa Plugin	400 /187
VIII Descend Display	
27. Yatray as an Integrated External Renderer	
Part 1	
Part 2	
Part 3	507 515
Glossary for the geeks	
28 From Blander to VafRay Using VableY	
What is Vable?	521 571
What is fable;	

Installing the script	
The Interface	
Yable Juicy example	535
29. YafRay	539
Introduction	539
Installation	
Scene Description Language Overview	541
Shaders	
Renderable Objects	551
Lights	
Background	
Camera	
Render	
Filters	
Glossary	567

# **Chapter 1. Introduction**

# What is Blender?

Blender is an integrated suite of tools enabling the creation of a broad range of 3D content. It offers full functionality for modelling, rendering, animation, post-production, creation and playback of interactive 3D content with the singular benefits of cross-platform operability and a download file size of less than 2.5MB.

Aimed at media professionals and artists, Blender can be used to create 3D visualizations, stills as well as broadcast quality video, while the incorporation of a real-time 3D engine allows for the creation of 3D interactive content for stand-alone playback.

Originally developed by the company 'Not a Number' (NaN), Blender now is continued as 'Free Software', with the sources available under GNU GPL.

Key Features:

- Fully integrated creation suite, offering a broad range of essential tools for the creation of 3D content, including modelling, animation, rendering, video post production and game creation;
- Small executable size, for easy distribution;
- Cross platform, with OpenGL based GUI, ready to use for all flavours of Windows, Linux, OSX, FreeBSD, Irix and Sun;
- · High quality 3D architecture enabling fast and efficient creation work-flow;
- Free support channels via www.blender3d.org;
- More than 250.000 people worldwide user community;

You can download the latest version of Blender at download.blender.org.

# **Blender's History**

In 1988 Ton Roosendaal co-founded the Dutch animation studio *NeoGeo*. NeoGeo quickly became the largest 3D animation studio in the Netherlands and one of the leading animation houses in Europe. NeoGeo created award-winning productions (European Corporate Video Awards 1993 and 1995) for large corporate clients such as multi-national electronics company Philips. Within NeoGeo Ton was responsible for both art direction and internal software development. After careful deliberation Ton decided that the current in-house 3D tool set for NeoGeo was too old and cumbersome to maintain and upgrade and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation suite we all now know as *Blender*. As NeoGeo continued to refine and improve Blender it became apparent to Ton that Blender could be used as a tool for other artists outside of NeoGeo.

In 1998, Ton decided to found a new company called Not a Number (NaN) as a spinoff of NeoGeo to further market and develop Blender. At the core of NaN was a desire to create and distribute a compact, cross platform 3D creation suite for free. At the time this was a revolutionary concept as most commercial modellers cost several thousands of (US) dollars. NaN hoped to bring professional level 3D modelling and animation tools within the reach of the general computing public. NaN's business model involved providing commercial products and services around Blender. In 1999 NaN attended its first Siggraph conference in an effort to more widely promote Blender. Blender's first 1999 Siggraph convention was a huge success and gathered a tremendous amount of interest from both the press and attendees. Blender was a hit and its huge potential confirmed! On the wings of a successful Siggraph in early 2000, NaN secured financing of 4.5 million EUR from venture capitalists. This large inflow of cash enabled NaN to rapidly expand its operations. Soon NaN boasted as many as fifty employees working around the world trying to improve and promote Blender. In the summer of 2000, Blender v2.0 was released. This version of Blender added the integration of a game engine to the 3D suite. By the end of 2000, the number of users registered on the NaN website surpassed 250,000.

Unfortunately, NaN's ambitions and opportunities didn't match the company's capabilities and the market realities of the time. This overextension resulted in restarting NaN with new investor funding and a smaller company in April 2001. Six months later NaN's first commercial software product, *Blender Publisher* was launched. This product was targeted at the emerging market of interactive web-based 3D media. Due to disappointing sales and the ongoing difficult economic climate, the new investors decided to shut down all NaN operations. The shutdown also included discontinuing the development of Blender. Although there were clearly shortcomings in the current version of Blender, with a complex internal software architecture, unfinished features and a non-standard way of providing the GUI, enthusiastic support from the user community and customers who had purchased Blender Publisher in the past, Ton couldn't justify leaving Blender to disappear into oblivion. Since restarting a company with a sufficiently large team of developers wasn't feasible, in March 2002 Ton Roosendaal founded the non-profit organization *Blender Foundation*.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community-based Open Source<sup>1</sup> project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to Blender as open source. The "Free Blender" campaign sought to raise 100,000 EUR so that the Foundation could buy the rights to the Blender source code and intellectual property rights from the NaN investors and subsequently release Blender to the open source community. With an enthusiastic group of volunteers, among them several ex-NaN employees, a fund raising campaign was launched to "Free Blender." To everyone's surprise and delight the campaign reached the 100,000 EUR goal in only seven short weeks. On Sunday October 13, 2002, Blender was released to the world under the terms of the GNU General Public License (GPL). Blender development continues to this day driven by a team of far-flung, dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Blender's history and road-map

- 1.00 Jan 1995 Blender in development at animation studio NeoGeo
- 1.23 Jan 1998 SGI version published on the web, IrisGL
- 1.30 April 1998 Linux and FreeBSD version, port to OpenGL and X
- 1.3x June 1998 NaN founded
- 1.4x Sept 1998 Sun and Linux Alpha version released
- 1.50 Nov 1998 First Manual published
- 1.60 April 1999 C-key (new features behind a lock, \$95), Windows version released
- 1.6x June 1999 BeOS and PPC version released
- 1.80 June 2000 End of C-key, Blender full freeware again
- 2.00 Aug 2000 Interactive 3D and real-time engine
- 2.10 Dec 2000 New engine, physics and Python
- 2.20 Aug 2001 Character animation system
- 2.21 Oct 2001 Blender Publisher launch
- 2.2x Dec 2001 Mac OSX version
- 13 October 2002 Blender goes Open Source, 1st Blender Conference

- 2.25 Oct 2002 Blender Publisher becomes freely available
- Tuhopuul Oct 2002 The experimental tree of Blender is created, a coder's play-ground.
- 2.26 Feb 2003 The first true Open Source Blender
- 2.27 May 2003 The second Open Source Blender
- 2.28x July 2003 First of the 2.28x series.
- 2.30 October 2003 At the 2nd Blender Conference the 2.3x UI makeover is presented.
- 2.31 December 2003 Upgrade to stable 2.3x UI project.
- 2.32 January 2004 Major overhaul of internal rendering capabilities.

# About Free Software and the GPL

When one hears about "free software", the first thing that comes to mind might be "no cost". While this is true in most cases, the term "free software" as used by the Free Software Foundation (originators of the GNU Project and creators of the GNU General Public License) is intended to mean "free as in freedom" rather than the "no cost" sense (which is usually referred to as "free as in free beer"). Free software in this sense is software which you are free to use, copy, modify, redistribute, with no limit. Contrast this with the licensing of most commercial software packages, where you are allowed to load the software on a single computer, are allowed to make no copies, and never see the source code. Free software allows incredible freedom to the end user; in addition, since the source code is available universally, there are many more chances for bugs to be caught and fixed.

When a program is licensed under the GNU General Public License (the GPL):

- you have the right to use the program for any purpose;
- you have the right to modify the program, and have access to the source codes;
- you have the right to copy and distribute the program;
- you have the right to improve the program, and release your own versions.

In return for these rights, you have some responsibilities if you distribute a GPL'd program, responsibilities that are designed to protect your freedoms and the freedoms of others:

- You must provide a copy of the GPL with the program, so that the recipient is aware of his rights under the license.
- You must include the source code or make the source code freely available.
- If you modify the code and distribute the modified version, you must license your modifications under the GPL and make the source code of your changes available. (You may not use GPL'd code as part of a proprietary program.)
- You may not restrict the licensing of the program beyond the terms of the GPL. (You may not turn a GPL'd program into a proprietary product.)

For more on the GPL, check the GNU Project Web site<sup>2</sup>. For reference, a copy of the GNU General Public License is included in Volume II.

# Getting support - the Blender community

Being freely available from the start, even while closed source, helped a lot in Blender's adoption. A large, stable and active community of users has gathered around Blender since 1998.

The community showed its best in the crucial moment of freeing Blender itself, going Open Source under GNU GPL in late summer 2002.

The community itself is now subdivided into two, widely overlapping sites:

- 1. The Development Community, centered around the Blender Foundation site http://www.blender.org/. Here is the home of the development projects, the Functionality and Documentation Boards, the CVS repository with Blender sources, all documentation sources, and related public discussion forums. Developers coding on Blender itself, Python scripters, documentation writers, and anyone working for Blender development in general can be found here.
- 2. The User Community, centered around the independent site http://www.elysiun.com/. Here Blender artists, Blender gamemakers and Blender fans gather to show their creations, get feedback on them, and ask for help to get a better insight into Blender's functionality. Blender Tutorials and the Knowledge Base can be found here as well.

These two websites are not the only Blender resources. The Worldwide community has created a lot of independent sites, in local languages or devoted to specialized topics. A constantly updated listing of Blender resources can be found at the above-mentioned sites.

For immediate online feedback there are three chat boxes permanently open on irc.freenode.net. You can join these with your favorite IRC client.

Chatboxes are **#blenderchat**, **#blenderqa** and **#gameblender**. The first of these is accessible even without an IRC client, using a plain Java enabled Web Browser through the elYsiun site (http://www.elysiun.com/).

# Notes

- 1. http://www.opensource.org/
- 2. http://www.gnu.org
- 3. http://www.blender.org/
- 4. http://www.elysiun.com/
- 5. http://www.elysiun.com/

# **Chapter 2. Installation**

Relevant to Blender v2.31

Blender is available both as binary executables and as source code on the Foundation site (http://www.blender.org/). From the main page look for the 'Downloads' section.

However, for correct usage of this book, using the version as provided on the included 2.3 Guide CDROM is highly recommended. Where in the text below "download" is mentioned, we also assume retrieving it from the CDROM.

# Downloading and installing the binary distribution

The Binary distributions comes in 6 basic flavors:

- Windows
- Linux
- MacOSX
- FreeBSD
- Irix
- Solaris

The Linux flavor comes actually in 4 different sub-flavors, for Intel and PowerPC architectures, with statically linked libraries or for dynamic loading libraries.

The difference between the dynamic and the static flavor is important. The static build has the OpenGL libraries compiled in. This makes Blender running at your system without using hardware accelerated graphics. Use the static version to check if Blender runs fine when the dynamic version fails! OpenGL is used in Blender for all drawing, including menus and buttons. This dependency makes a proper and compliant OpenGL installation at your system a requirement. Not all 3D card manufacturers provide such compliancy, especially cheaper cards aimed at the gaming market.

Of course since renderings are made by Blender rendering engine in core memory and by the main CPU of your machine, a graphic card with hardware acceleration makes no difference at rendering time.

# Windows

#### **Quick Install**

Download the file blender-2.3#-windows.exe, being 2.3# the version number, from the downloads section of the Blender Website. Start the installation by double-clicking the file. This presents you with some questions, for which the defaults should be ok. After setup is complete, you can start Blender right away, or use the entry in the Start menu.

### **In-depth Instructions**

Download the file blender-2.3#-windows.exe from the downloads section of the Blender Website. Choose to download it (if prompted), select a location and click "Save". Then navigate with explorer to the location you saved the file in and double-click it to start the installation.

The first dialog presents you the license. You are expected to accept it if you want the installation to go any further. After accepting the license, select the components you wish to install (there is just one, Blender) and the additional actions you want to take. There are three: Add a shortcut to the Start menu, Add Blender's icon to desktop, associate .blend files with Blender. By default they are all checked. If you don't want some action to be taken simply uncheck it. When done, click on Next.

Select a place to install the files to (the default should do well), and click Next to install Blender. Press Close when installation is over.

Afterwards you will be asked whether you want to start Blender immediately. Blender is now installed and can be started by means of the Start menu (an entry named "Blender Foundation" has been created by the setup routine) or by double-clicking a Blender file (\*.blend).

# OSX

#### Install

Download the file blender-2.3#-darwin-6.6-powerpc.dmg from the downloads section of the Blender Website. Extract it by double-clicking the file. This will open a directory with several files.

Since Blender uses OpenGL for the entire GUI, and Mac OSX draws the entire Desktop with OpenGL as well, you will need to verify first you have sufficient VRAM in your system. Below 8 MB VRAM Blender will not run at all. Up to 16 MB VRAM you will need to set your system at "1000s of colors" (System Preferences -> Displays).

You now can use Blender by double clicking the Blender icon. Or drag the Blender icon to the Dock to make an alias there. Blender starts by default in a smaller window. Use the "+" button in the window header to maximize. More hints and tips about the OSX version can be found in the file OSX tips.rtf in the installation directory.

#### Linux

#### Quick Install

Download the file blender-2.3#-linux-glibc#.#.#-ARCH.tar.gz from the downloads section of the Blender Website. Here 2.3# is Blender version, #.#.# is glibc version and ARCH is the machine architecture, either i386 or powerpc. You should get the one matching your system, remember the choice between static and dynamic builds.

Unpack the archive to a location of your choice. This will create a directory named blender-2.3#-linux-glibc#.#.ARCH, in which you will find the **blender** binary.

To start blender just open a shell and execute ./blender, of course when running X.

#### In-depth Instructions

Download the file blender-2.3#-linux-glibc#.#.#-ARCH.tar.gz from the downloads section of the Blender Website. Choose to download it (if prompted), select a location and click "Save". Then navigate to the location you wish blender to install to (e.g./usr/local/) and unpack the archive (with tar xzf/path/to/blender-2.3#-linuxglibc#.#.#-ARCH.tar.gz). If you like, you can rename the resulting directory from blender-2.3#-linux-glibc#.#.#-ARCH to something shorter, e.g. just blender. Blender is now installed and can be started on the command line by entering **cd** /**path/to/blender** followed by pressing the enter key in a shell. If you are using KDE or Gnome you can start Blender using your file manager of choice by navigating to the Blender executable and (double-) clicking on it.

If you are using the Sawfish window manager, you might want to add a line like ("Blender" (system "blender &")) to your .sawfish/rc file.

#### To add program icons for Blender in KDE

- 1. Select the "Menu Editor" from the System submenu of the K menu.
- 2. Select the submenu labeled "Graphics" in the menu list.
- 3. Click the "New Item" button. A dialog box will appear that prompts you to create a name. Create and type in a suitable name and click "OK". "Blender" or "Blender 2.3#" would be logical choices, but this does not affect the functionality of the program.
- 4. You will be returned to the menu list, and the Graphics submenu will expand, with your new entry highlighted. In the right section, make sure the following fields are filled in: "Name", "Comment", "Command", "Type" and "Work Path".
  - The "Name" field should already be filled in, but you can change it here at any time.
  - Fill the "Comment" field. This is where you define the tag that appears when you roll over the icon.
  - Click the folder icon at the end of the "Command" field to browse to the blenderpublisher program icon. Select the program icon and click "OK" to return to the Menu Editor.
  - The "Type" should be "Application".
  - The "Work Path" should be the same as the "Command", with the program name left off. For example, if the "Command" field reads /home/user/blender-publisher-#.##-linuxglibc#.#.#-ARCH/blender, the "Work Path" would be /home/user/blender-publisher-#.##-linux-glibc#.#.#-ARCH/.
- 5. Click "Apply" and close out of the Menu Editor.

To add a link to Blender on the KPanel, right-click on a blank spot on the KPanel, then hover over "Add", then "Button", then "Graphics", and select "Blender" (or whatever you named the menu item in step 3). Alternately, you can navigate through the "Configure Panel" submenu from the K menu, to "Add", "Button", "Graphics", "Blender".

To add a Desktop icon for Blender, open Konquerer (found on the Panel by default, or in the "System" submenu of the K menu) and navigate to the blenderpublisher program icon where you first unzipped it. Click and hold the program icon, and drag it from Konquerer to a blank spot on your Desktop. You will be prompted to Copy Here, Move Here or Link Here, choose Link Here.

#### To add program icons for Blender in GNOME

- 1. Select "Edit menus" from the Panel submenu of the GNOME menu.
- 2. Select the "Graphics" submenu, and click the "New Item" button.
- 3. In the right pane, fill in the "Name:", "Comment:" and "Command:" fields. Fill the "Name:" field with the program name, for example "Blender". You can name this whatever you'd like, this is what appears in the menu, but does not affect the functionality of the program. Fill the "Comment:" field with a descriptive comment. This is what is shown on the tooltips popups. Fill the "Command:" field with

the full path of the blenderpublisher program item, for example, /home/user/blender-publisher-#.##-linux-glibc#.#.#-ARCH/blender

- 4. Click the "No Icon" button to choose an icon. There may or may not be an icon for Blender in your default location. You can make one, or look for the icon that goes with KDE. This should be /opt/kde/share/icons/hicolor/48x48/apps/blender.png. If your installation directory is different, you can search for it using this command in a Terminal or Console: find / -name "blender.png" -print
- 5. Click the "Save" button and close the Menu Editor.

To add a Panel icon, right-click a blank area of the Panel, then select "Programs", then "Graphics", then "Blender". Alternatively, you could click the GNOME menu, then select "Panel", then "Add to panel", then "Launcher from menu", then "Graphics", and "Blender".

To add a Desktop icon for Blender, open Nautilus (double-click the Home icon in the upper-left corner of your Desktop, or click the GNOME menu, then "Programs", then "Applications", and "Nautilus"). Navigate to the folder which contains the blender-publisher program icon. Right-click the icon, and drag it to the Desktop. A menu will appear asking to Copy Here, Move Here, Link Here or Cancel. Select Link Here.

# FreeBSD

#### Install

Download the file blender-2.3#-freebsd-#.#-i386.tar.gz from the downloads section of the Blender Website. Here 2.3# is Blender version, #.# is FreeBSD version and i386 is the machine architecture.

To start blender just open a shell and execute ./blender, of course when running X.

#### lrix

#### Install

Download the file blender-2.3#-irix-6.5-mips.tar.gz from the downloads section of the Blender Website. Here 2.3# is Blender version, 6.5 is Irix version and mips is the machine architecture.

To start Blender just open a shell and execute **./blender**, of course when running X. Blender was originally developed for the IRIX platform, but is currently not actively being maintained for all IRIX workstation versions. For some workstations performance troubles have been reported.

# Solaris

#### Install

Download the file blender-2.3#-solaris-2.8-sparc.tar.gz from the downloads section of the Blender Website. Here 2.3# is Blender version, 2.8 is Solaris version and sparc is the machine architecture.

Currently no further instructions for Sun Solaris are available. Please use the Blender website forums for support.

# **Building Blender from source**

#### Relevant to Blender v2.31

This document describes the tools necessary to build Blender from source, either from CVS or from a source package. Building from CVS requires the use of more tools. While this may be a bit more troublesome than building from a source package, this may be necessary for some people. For example, when you want to build Blender for an unsupported platform or when you want to implement some new features.

This is a very early version of this document. This means that it is incomplete and that some procedures or concepts might be incorrect for your system. Please keep this in mind when reading this. Also keep in mind Blender is a complex product which will require you to create the right environment for.

### Getting the sources

The following paragraphs will describe how and where to get the sources needed for building Blender.

#### Get the latest stable source package

The sources are available on CDROM accompanying this book. You can also download it from the website, http://www.blender3d.org/Download/?sub=Source

#### Get the latest sources from CVS

CVS stands for Concurrent Versioning System. It is a software configuration tool that keeps the various source files in a central repository. CVS enables developers to quickly update to the latest state of the repository and commit changes. The tool keeps track of the changes between each version of a file. To get the current state of the repository, you don't need to have a username for accessing the sources. This feature is optional, but in an opensource development, it's almost a requirement. To commit changes to the repository, however, you need to have developer access. Since this document only describes how to get the latest state of the sources, the commit procedures are not described here.

To get the latest state of the sources use:

#### export CVSROOT=:pserver:anonymous@cvs.blender.org:/cvs01

cvs login

password: Enter

#### cvs -z3 co blender

Please do not use a higher level of compression for accessing the Blender server.

If you already have a working set of files obtained from the server, you can use the **update** command to update the sources to the current state of the repository. **cd** to the blender source tree on your system and type in the following command:

cvs -z3 update.

# **External libraries needed**

Blender is a package that uses a lot of external packages for expanding its functionality. Each of these packages have, just as Blender, a history of changes. Newer versions of such a package will probably have more features and less known problems. As a developer it is exciting to work with the latest features available to get the most out of the tool. However, the number of developers out there is much lower than the number of end-users who are not interested in the latest feature, these users want an application that works. Since Blender has to run on multiple platforms, all those platforms have to have the same minimum functionality available in the external packages.

The table below displays the packages needed and the minimum version of those packages. Over time it is possible that those minimum versions are increased as the demand for the newer features is high.

Library	Version
glibc	2.2.4
libjpeg	6b
libpng	1.0.14
libsdl	1.0
libz	1.1.4
mesa	3.4.2
openAL	N/A
openGL	1.1 (1.2 for engine)
python	2.2

Table 2-1. Minimum version external libraries

Not all libraries apply to all platforms. The following table gives an overview of the currently supported platforms and the required libraries. An 'X' means that it is needed, a '-' means that it is not needed and an 'O' means that it is optional.

Library	Linux	Windows	FreeBSD	IRIX	MacOS X
glibc	X	-	X	X	x
libjpeg	X	X	X	X	x
libpng	X	X	X	X	x
libsdl	0	0	0	0	0
libz	X	X	Х	X	X
mesa	X	X	Х	-	-
openAL	X	X	Х	X	x
openGL	-	-	-	X	x
python	X	X	X	X	X

Table 2-2. Platform dependent library requirements

# **Tools needed**

Having the necessary libraries installed and the Blender sources downloaded to your system means that you're now able to build Blender. The entire build process requires

some tools to be available on your system. In the table below, the list of tools along with the minimum version is shown. The third column shows if the tool is required for CVS only ('X'). If the tool is not required for a source package build, a '-' is shown.

ΤοοΙ	Version	cvs	Note
autoconf	2.53	Х	
automake	1.6.2	Х	
cvs	1.11.1p1	Х	
docbook	3.1	0	
doxygen	N/A	0	
gawk	3.1.0	X	
gcc	2.96	-	
gettext	0.11	-	
gmake	3.79.1	-	
m4	1.4	X	
sed	3.02	X	
sh	2.05.1	-	
Visual C++	6.0 SP5	-	Windows only

#### Table 2-3. Minimum version tools

**Python:** Python is not included in this table although it is used to build Blender. The reason that it is not included is because Python is also needed as an external library and thus has to be installed already as has been written in the previous section.

### **Building Blender**

There are two build systems for using gcc or cc compilers; regular Makefiles, which stem from the period Blender was developed in the company NaN, and the automake/autoconf "configure" style one. Using "configure" can write over the NaN Makefiles, so you have to choose either one.

For Windows MSVC, Blender supports usage of project files and workspaces.

The files describing detailed build information are located in the blender root directory:

- INSTALL: general information, download links for libraries
- INSTALL.AUTO: using autoconf and configure scripts
- INSTALL.MAKE: using regular makefiles
- INSTALL.MSVC: using Microsoft Visual C project files

### **Technical support**

- portal: http://www.blender.org
- overview: http://www.blender.org/docs/get\_involved.html

# Chapter 2. Installation

- mailinglist: http://www.blender.org/mailman/listinfo/bf-committers/
- bug tracker: http://projects.blender.org/tracker/?group\_id=9
- IRC: irc.freenode.net, #blendercoders

# Notes

1. http://www.blender.org/

# Chapter 3. Understanding the interface

#### By Martin Kleppmann

If you are new to Blender, you should get a good grip on how to work with the user interface before modelling. The concepts behind Blender's interface are non-standard, and different from other 3D software packages. Windows users especially will need to get used to the different way that Blender handles controls, such as button choices and mouse movements. But this difference is in fact one of Blender's great strengths: once you understand how to work the Blender way, you will find that you can work exceedingly quickly and productively.

Furthermore, Blender's interface greatly changed in the transition from version 2.28 to version 2.3, so even experienced users might profit from this chapter.

# **Blender's Interface Concept**

Relevant to Blender v2.31

The user interface is the vehicle for two way interaction between the user and the program. The user communicates with the program via the keyboard and the mouse, the program gives feedback via the screen and its windowing system.

## Keyboard and mouse

Blender's interface makes use of three mouse buttons and a wide range of hotkeys (for a complete in-depth discussion refer to Volume II). If your mouse has only two buttons, you can emulate the middle mouse button (the Section called *User preferences and Themes* describes how). A mouse wheel can be used, but it is not necessary as there are also appropriate keyboard shortcuts.

This book uses the following conventions to describe user input:

- The mouse buttons are called LMB (left mouse button), MMB (middle mouse button) and RMB (right mouse button).
- If your mouse has a wheel, **MMB** refers to clicking the wheel as if it were a button, while **MW** means rolling the wheel.
- Hotkey letters are named by appending *KEY* to the letter, *i.e.* **GKEY** refers to the letter *g* on the keyboard. Keys may be combined with the modifiers **SHIFT**, **CTRL** and/or **ALT**. For modified keys the **KEY** suffix is generally dropped, for example **CTRL-W** or **SHIFT-ALT-A**.
- **NUM0** to **NUM9**, **NUM+** and so on refer to the keys on the separate numeric keypad. NumLock should generally be switched on.
- Other keys are referred to by their names, such as ESC, TAB, F1 to F12.
- Other special keys of note are the arrow keys, **UPARROW**, **DOWNARROW** and so on.

Because Blender makes such extensive use of both mouse and keyboard, a "golden rule" has evolved among Blender users: keep one hand on the mouse and the other on the keyboard! If you normally use a keyboard that is significantly different from the English keyboard layout, you may want to think about changing to the English or American layout for your work with Blender. The most frequently used keys are grouped so that they can be reached by the left hand in standard position (index finger on **FKEY**) on the English keyboard layout. This assumes that you use the mouse with your right hand.

### The window system

Now it's time to start Blender and begin playing around.

🚺 🗄 🗢 File Add Timeline Game Render Help ≑ SCR:2-Model 🗶 💠 SCE	Scene 🗙 🐟 www.blender.org 231 Ve:0   Fa:0	Ob:0-0   La:0   Mi
<u></u>		
o		
<sup>2</sup> <sub>×</sub> (1)Cube		
	<u></u>	{
Chippen Chippen Shadow EnviMap	Game framing settings >>	PAL
Image: Comparison of the second se	ANIM SizeX: 640 SizeY: 480	NTSC Default
S 8 11 16 f: 0.500 75% 50% 25%	Bender Daemon Aspx: 100 Aspy: 100	Preview PC
Backburf Edge Edge Settings	PLAY rt: 25 PNG Crop	PAL 16:9 PANO
Distriction Densilies Extension Sky Premul Key Border Gamma	Sta: 1 End: 250 BW RGB RGBA	FULL

#### Figure 3-1. The default Blender scene.

Figure 3-1 shows the screen you should get after starting Blender (except for the added text and arrows). At default it is separated into three windows: The main menu at the top, the large 3D Window and the Buttons Window at the bottom. Most windows have a header (the strip with a lighter grey background containing icon buttons - for this reason we will also refer to the header as the window *ToolBar*); if present, the header may be at the top (as with the Buttons window) or the bottom (as with the 3D Window) of a window's area.

If you move the mouse over a window, note that its header changes to a lighter shade of grey. This means that it is "focused;" all hotkeys you press will now affect the contents of this window.

You can easily customize Blender's window system to suit your needs and wishes. You can create a new window by splitting an existing one in half. Do so by focusing the window you want to split (move the mouse into it), clicking the border with **MMB** or **RMB**, and selecting Split Area (Figure 3-2). You can now set the new border's position by clicking with **LMB**, or cancel your action by pressing **ESC**. The new window will start as a clone of the window you split, but can then be set to a different window type, or to display the scene from a different point of view.

**Interface Items:** Labels in the interface buttons, menu entries and, in general, all text shown on the screen is highlighted in this book like this.



#### Figure 3-2. The Split menu for creating new windows.

Create a new vertical border by choosing Split Area from a horizontal border, and vice versa. You can resize each window by dragging a border with LMB. To reduce the number of windows, click a border between two windows with MMB or RMB and choose Join Areas. The resulting window receives the properties of the previously focused window.

To set a header's position click **RMB** on the header and choose Top or Bottom. You can also hide the header by choosing No Header, but this is only advisable if you know all the relevant hotkeys. You can show a hidden header again by clicking the window's border with **MMB** or **RMB** and selecting Add Header.

# Window types

Each window frame may contain different types and sets of information, depending upon what you are working on. These may include 3D models, animation, surface materials, Python scripts, and so on. You can select the type for each window by clicking its header's leftmost button with **LMB** (Figure 3-3).



#### Figure 3-3. The window type selection menu.

We'll explain the functions and usage of the respective window types later in this book. For now we only need to concern ourselves with the three window types that are already provided in Blender's default scene:

#### **3D** Viewport

Provides a graphical view into the scene you are working on. You can view your scene from any angle with a variety of options; see the Section called *Navigating in 3D Space* for details. Having several 3D Viewports on the same screen can be useful if you want to watch your changes from different perspectives at the same time.

#### **Buttons Window**

Contains most tools for editing objects, surfaces, textures, Lights, and much more. You will need this window constantly if you don't know all hotkeys by heart. You might indeed want more than one of these windows, each with a different set of tools.

#### User preferences (Main menu)

This window is usually hidden, so that only the menu part is visible - see the Section called *User preferences and Themes* for details. It's rarely used though, since it contains global configuration settings. There are several novelties in Blender 2.30. First of all window headers tend to be much cleaner, less cluttered by buttons, and menus are now present in many headers.

Most window headers, immediately next to this first "Window Type" Menu button exhibit a set of menus; this is one of the main new features of the 2.30 interface. Menus now allow you to directly access many of the features and commands which previously were only accessible via hot keys or arcane buttons. Menus can be hidden and showed via the triangular button next to them.

Menu are not only window-sensitive (they change with window type) but also context sensitive (they change with selected object) so they are always very compact, showing only actions which can actually be performed.

All Menu entries shows the relevant hotkey shortcut, if any. Blender Workflow is at his best when hotkeys are used. So the rest of this Book will mostly present you hotkeys, rather than Menu entries. Menus are anyway precious since they give a complete as possible overview of all tools and commands Blender offers.

One feature of windows that sometimes comes in handy for precise editing is that of maximizing to full screen. If you use the appropriate View>>Maximize Window Menu entry or the hotkey **CTRL-DOWNARROW**, the focused window will extend to fill the whole screen. To return to normal size, use the View>>Tile Window button again or **CTRL-UPARROW**.

### **Contexts, Panels and Buttons**

Blender's buttons are more exciting than those in most other user interfaces, and they became even nicer in 2.30. This is largely due to the fact that they are vector-based and drawn in OpenGL, which makes them elegant and zoomable.

Buttons are mostly grouped in the Button Window. As of Blender 2.3 the Button Window shows *six* main contexts, which can be chosen via the first icon row in the header (Figure 3-4), each of which might be subdivided into a variable number of sub-contexts, which can be chosen via the second icon row in the header (Figure 3-4):



#### Figure 3-4. Contexts and Sub-Contexts

- Logic shortcut F4
- *Script* no shortcut
- Shading shortcut F5
  - Lamp no shortcut
  - Material no shortcut
  - Texture shortcut F6
  - Radio no shortcut
  - World shortcut F8

- Object shortcut F7
- Editing shortcut F9
- Scene shortcut F10
  - Rendering no shortcut
  - Anim/Playback no shortcut
  - Sound no shortcut

Once the Context is selected by the user, the sub-context is usually determined by Blender on the basis of the active Object. For example, with the "Shading" context, if a Lamp Object is selected then sub-context shows Lamp Buttons, if a Mesh or other renderable Object is selected, then Material Buttons is the active sub-context, and if a Camera is selected the active sub-context is World.

The most notable novelty in the interface is probably the presence of *Panels* to logically group buttons. Each panel is the same size. They can be moved around the Button Window by **LMB** clicking and dragging on their header. Panels can be aligned by **RMB** on the Buttons Window and chosing the desired layout from the Menu which appears (Figure 3-5).

Align buttons
Free
Horizontal
Vertical

#### Figure 3-5. Button Window Menu.

**MW** scrolls Panels in their aligned direction, **CTRL-MW** and **CTRL-MMB** zooms panels in and out. Single panels can be collapsed/expanded by **LMB** clicking the triangle on the left of their header.

Particularly complex Panels are organized in *Tabs*. Clicking **LMB** on a Tab in the Panel header changes the buttons shown (Figure 3-6). Tabs can be "torn out" of a panel to form independent panels by clicking **LMB** on their header and dragging them out. In a similar way separate Panels can be turned into a single panel with Tabs by dropping one Panel's header into another.

🔻 Text	ure	Map I	nput	Map 1	Го
UV	Object				
Glob	Orco	Stick	Win	Nor	Refl
Flat	Cube		< of	sX 0.0	00 →
Tube	Sphe		< of	'sY 0.0	00 🕨
			< of	'sZ 0.0	00 🕨
X	ΥZ	]	< si	zeX 1.0	)0 🔸
X	ΥZ		≪ i si	zeV 1.0	)0 🔹 🕨
X	Y Z		< si	zeZ 1.0	)0 🕨

#### Figure 3-6. Panel with Tabs.

As a last interface item in the chain there are several kind of buttons which are disposed in the Panel's Tabs:

#### **Operation Button**

These are buttons that perform an operation when they are clicked (with **LMB**, as all buttons). They can be identified by their brownish color in the default Blender scheme (Figure 3-7).



#### Figure 3-7. An operation button

#### **Toggle Button**

Toggle buttons come in various sizes and colors (Figure 3-8). The colors green, violet, and grey do not change functionality, they just help the eye to group the buttons and recognize the contents of the interface more quickly. Clicking this type of button does not perform any operation, but only toggles a state as "on" or "off."

Some buttons also have a third state that is identified by the text turning yellow (the Ref button in Figure 3-8). Usually the third state means "negative," and the normal "on" state means "positive."

Col	Nor	Csp	Cmir	Ref	Spec
Hard	Ray Mi	Alpha	Emit	Translu	Disp

#### **Radio Buttons**

Radio buttons are particular groups of mutually exclusive Toggle buttons. No more than one Radio Button in a given group can be "on" at one time.

#### **Num Buttons**

Number buttons (Figure 3-10) can be identified by their captions, which contain a colon followed by a number. Number buttons are handled in several ways: To increase the value, click **LMB** on the right of the button, where the small triangle is shown; to decrease it, click on the left of the button, where another triangle is shown. To change the value in a wider range, hold down **LMB** and drag the mouse to the left or right. If you hold **CTRL** while doing this, the value is changed in discrete steps; if you hold **SHIFT**, you'll have finer control over the values. **ENTER** can be used in place of **LMB** here.

✓ SizeX: 640 »	∢ Size∀: 480 ≽
✓ AspX: 100 ►	≪ AspV: 100 ⊁
R 1.000	
G 0.000 ⊨ 🗕	
B 1.000	

#### Figure 3-9. Number buttons

You can enter a value from the keyboard by holding **SHIFT** and clicking **LMB**. Press **SHIFT-BACKSPACE** to clear the value; **SHIFT-LEFTARROW** to move the cursor to the beginning; and **SHIFT-RIGHTARROW** to move the cursor to the end. Press **ESC** to restore the original value.

Some number buttons contain a slider rather than just a number with side triangles. The same method of operation applies, except that single **LMB** clicks must be performed on the left or on the right of the slider, while clicking on the label or the number automatically enters keyboard input mode.

#### **Menu Buttons**

Use the Menu buttons to choose from dynamically created lists. Menu buttons are principally used to link DataBlocks to each other. (DataBlocks are structures like Meshes, Objects, Materials, Textures, and so on; by linking a Material to an Object, you assign it.) You'll see an example for such a block of buttons in Figure 3-10. The first button (with the tiny up and down pointing triangles) opens a menu that lets you select the DataBlock to link to by holding down **LMB** and releasing it over the requested item. The second button displays the type and name of the linked DataBlock and lets you edit its name after clicking **LMB**. The "X" button clears the link, the "car" button generates an automatic name for the DataBlock, and the "F" button specifies whether the DataBlock should be saved in the file even if it is unused (unlinked).

**Unlinked objects:** Unlinked data is *not* lost until you quit Blender. This is a powerful Undo feature. if you delete an object the material assigned to it becomes unlinked, but is still there! You just have to re-link it to another object or press the "F" button.



#### Figure 3-10. Datablock link buttons

# Toolbox

By pressing **SPACE** in the 3D Viewport, or by holding **LMB** or **RMB** with a still mouse for more than half a second opens the Toolbox. This contains 6 main contexts, arranged on two lines, each of which opens menus and submenus.

Three of these contexts open the same three menus present in the 3D Viewport header, of the other three, Add allows adding new Objects to the scene while Edit and Transform shows all possible operations on selected Object(s) (Figure 3-11).

Move to Layer	M	the second s			
Copy Properties	Ctrl C	Lattice			
Clear Track	Alt T	Armature			
Make Track	Ctrl T	Lamp			
Clear Parent	Alt P	Camera			
Make Parent	Ctrl P	Empty			
Make Single User	U	Text			
Copy Links	Ctrl L	MBall ►	Grouped	Shift G	
Delete	Х	Surface 🕨	Linked	Shift L	
Duplicate Linked	Alt D	Curve 🕨	(De)select All	A	
Duplicate	Shift D	Mesh 🕨	Border Select	в	
	)bject	Add Select			
E	dit	Transform View			
Enter Editmode	Tab	Grabber	g	Viewport Shading	)
Insert Kev	1	Rotate	r	Ortho/Persp	Pad 5
Object Keys	+	Scale	s	Local View	Pad
Boolean	w	Properties	n	Frame All	Home
Join Objects	CtrL	Snap	Shift S	Frame Selected	Pad
Convert Object	Alt C	Clear Location		Centre Cursor	C
	11110	Clear Rotation		Play Back	Alt A
		Clear Size	1.1	,	
		Apply Rot/Size	Ctrl A		
		Apply Deform	Shift Ctrl A		

**Figure 3-11. The Toolbox** 

#### Screens

Blender's flexibility with windows lets you create customized working environments for different tasks, such as modeling, animating, and scripting. It is often useful to quickly switch between different environments within the same file. This is made possible by creating several screens: All changes to windows as described in the Section called *The window system* and the Section called *Window types* are saved within one screen, so if you change your windows in one screen, other screens won't be affected. But the scene you are working on stays the same in all screens.

Three different default screens are provided with Blender; they are available via the SCR Menu Buttons in the User Preferences Window header shown in Figure 3-12. To change to the next screen alphabetically, press **CTRL-RIGHTARROW**; to change to the previous screen alphabetically, press **CTRL-LEFTARROW**.



Figure 3-12. Screen and Scene selectors

#### **Scenes**

It is also possible to have several scenes within the same Blender file. The scenes may use one another's objects or be completely separate from one another. You can select and create scenes with the SCE Menu Button buttons in the User Preferences Window header (Figure 3-12).

When you create a new scene, you can choose between four options to control its contents:

- Empty creates an empty scene.
- Link Objects creates the new scene with the same contents as the currently selected scene. Changes in one scene will also modify the other.
- Link ObData creates the new scene based on the currently selected scene, with links to the same meshes, materials, and so on. This means that you can change objects' positions and related properties, but modifications to the meshes, materials, and so on will also affect other scenes unless you manually make single-user copies.
- Full Copy creates a fully independent scene with copies of the currently selected scene's contents.

# Navigating in 3D Space

Relevant to Blender v2.31

Blender lets you work in three-dimensional space, but our monitor screens are only two-dimensional. To be able to work in three dimensions, you must be able to change your viewpoint as well as the viewing direction of the scene. This is possible in all of the 3D Viewports.

Even if we will describe the 3D Viewport Window, most non-3D windows use an equivalent series of functions, for example it is even possible to translate and zoom a Buttons Window and its Panels.

# The viewing direction (rotating)

Blender provides three default viewing directions: Side, Front, and Top. As Blender uses a right-hand coordinate system with the Z axis pointing upwards, "side" corresponds to looking along the X axis, in the negative direction; "front" along the Y axis; and "top" along the Z axis. You can select the viewing direction for a 3D Viewport with the View Menu entries (Figure 3-13) or by pressing the hotkeys **NUM3** for "side", **NUM1** for "front", and **NUM7** for "top".

**Hotkeys:** Remember that most hotkeys affect the window that has focus, so check that the mouse cursor is in the area you want to work in before your use the hotkeys!

	Play Back Animation	Alt A		
	Centre Cursor	С		
	Align View to Selected	d NumPad*		
	Frame Selected	NumPad.		
	Frame All	Home		
	Viewport Navigation	•	Orbit Left	NumPad 4
~	Global View	NumPad /	Orbit Right	NumPad 6
	Local View	NumPad /	Orbit Up	NumPad 8
	Orthographic	NumPad 5	Orbit Down	NumPad 2
Ť	Parenactiva	NumPad 5	Zoom In	NumPad +
-	i cispective	INGINI AU O	Zoom Out	NumPad -
	Side	NumPad 3	Reset Zoom	NumPad Enter
	Front	NumPad 1		
~	Тор	NumPad 7		
	Camera	NumPad 0		
	User			
	Maximize Window	Ctrl UpArrow		
	Background Image			
٥	View Properties			
Vie	ew Select Object	ধ Object Mode	🗢 🕼 t 🖸 t	

Figure 3-13. A 3D Viewport's view menu.

Apart from these three default directions, the view can be rotated to any angle you wish. Click and drag **MMB** on the Viewport's area: If you start in the middle of the window and move up and down or left and right, the view is rotated around the middle of the window. If you start at the edge and don't move towards the middle, you can rotate around your viewing axis. Play around with this function until you get the feeling for it.

To change the viewing angle in discrete steps, use **NUM8** and **NUM2**, which correspond to vertical **MMB** dragging. Or use **NUM4** and **NUM6**, which correspond to horizontal **MMB** dragging.

# Translating and Zooming the View

To translate the view, hold down **SHIFT** and drag **MMB** in the 3D Viewport. For discrete steps, use the hotkeys **CTRL-NUM8**, **CTRL-NUM2**, **CTRL-NUM4** and **CTRL-NUM6** as with rotating.

You can zoom in and out by holding down **CTRL** and dragging **MMB**. The hotkeys are **NUM+** and **NUM-**. The View>>Viewport Navigation sub-menu holds these functions too.

Wheel Mouse: If you have a wheel mouse, you can perform all of the actions that you would do with **NUM+** and **NUM-** by rotating the wheel (**MW**). The direction of rotation selects the action.

If You Get Lost...: If you get lost in 3D space, which is not uncommon, two hotkeys will help you: HOME changes the view so that you can see all objects (View>>Frame All Menu entry,) while NUM. zooms the view to the currently selected objects (View>>Frame Selected Menu entry.)

# **Perspective and Orthographic Projection**

Each 3D Viewport supports two different types of projection. These are demonstrated in Figure 3-14: orthographic (left) and perspective (right).



#### Figure 3-14. Orthographic (left) and perspective (right) projection.

Our eye is used to perspective viewing because distant objects appear smaller. Orthographic projection often seems a bit odd at first, because objects stay the same size independent of their distance: It is like viewing the scene from an infinitely distant point. Nevertheless, orthographic viewing is very useful (it is the default in Blender and most other 3D applications), because it provides a more "technical" insight into the scene, making it easier to draw and judge proportions.

**Perspective and Orthographic:** Perspective view is geometrically constructed this way: you have a scene in 3D and you are an observer palced in a point O. The 2D perspective scene is built by placing a plane, a sheet of paper where the 2D scene is to be drawn in front of point O, perpendicular to the viewing direction. For each point P in the 3D scene a line is drawn, passing from O and P. The intersection point S between this line and the plane is the perspective projection of that point. By projecting all points P of the scene you get a perspective view.

In an orthographic projection, also called "orthonormal", on the other hand, you have a viewing direction but not a viewing point O. The line is then drawn through point P so that it is parallel to the viewing direction. The intersections S between the line and the plane is the orthographic projection. And by projecting all point P of the scene you get the orthographic view.

To change the projection for a 3D Viewport, choose View>>Orthographic or View>>Perspective Menu entries (Figure 3-13). The hotkey **NUM5** toggles between the two modes.

**Camera projection:** Note that changing the projection for a 3D Viewport does not affect the way the scene will be rendered. Rendering is in perspective by default. If you need to

create an Orthographic rendering, select the camera and press  $\tt ortho$  in the EditButtons (F9)  $\tt Camera$  Panel.

The View>>Camera Menu entry sets the 3D Viewport to camera mode (Hotkey: **NUM0**). The scene is then displayed as it will be rendered later (see Figure 3-15): the rendered image will contain everything within the outer dotted line. Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.



Figure 3-15. Demonstration of camera view.

### Draw mode

Depending on the speed of your computer, the complexity of your Scene, and the type of work you are currently doing, you can switch between several drawing modes:

- Textured Attempts to draw everything as completely as possible, though it is still no alternative to rendering. Note that if you have no lighting in your scene, everything will remain black.
- Shaded Draws solid surfaces including the lighting calculation. As with textured drawing, you won't see anything without lights.
- Solid Surfaces are drawn as solids, but the display also works without lights.
- Wireframe Objects only consist of lines that make their shapes recognizable. This is the default drawing mode.
- Bounding Box Objects aren't drawn at all; instead this mode shows only the rectangular boxes that correspond to each object's size and shape.

#### Chapter 3. Understanding the interface

The drawing mode can be selected with the appropriate Menu Button in the header (Figure 3-16) or with hotkeys: **ZKEY** toggles between wireframe and solid display, **SHIFT-Z** toggles between wireframe and shaded display.



Figure 3-16. A 3D Viewport's draw mode button.

# Local view

When in local view, only the selected objects are displayed, which can make editing easier in complex scenes. To enter local view, first select the objects you want (see the Section called *Selecting objects* in Chapter 5) and then use the View>Local View Menu entry; use the View>Global View Menu entry to go back to Global View. (Figure 3-13). The hotkey is **NUM**/, which toggles Local/Global View.

### The layer system

3D scenes often become exponentially more confusing with growing complexity. To get this under control, objects can be grouped into "layers," so that only the layers you select are displayed at any one time. 3D layers differ from the layers you may know from 2D graphics applications: they have no influence on the drawing order and are there (except for some special functions) solely to provide the modeler with a better overview.

Blender provides 20 layers; you can choose which are to be displayed with the small unlabeled buttons in the header (Figure 3-17). To select only one layer, click the appropriate button with **LMB**; to select more than one, hold **SHIFT** while clicking.



Figure 3-17. A 3D Viewport's layer buttons.
To select layers via the keyboard, press **1KEY** to **0KEY** (on the main area of the keyboard) for layers 1 through 10 (the top row of buttons), and **ALT-1** to **ALT-0** for layers 11 through 20 (the bottom row). The **SHIFT** key for multiple selection works for these hotkeys too.

By default, the lock button directly to the right of the layer buttons is pressed; this means that changes to the viewed layers affect all 3D Viewports. To select only certain layers in one window, deselect locking first.

To move selected objects to a different layer, press **MKEY**, select the layer you want from the pop-up dialog, then press the Ok button.

# The vital functions

Relevant to Blender v2.31

# Loading files

Blender uses the .blend file format to save nearly everything: Objects, scenes, textures, and even all your user interface window settings.

To load a Blender file from disk, press **F1**. The focused window then temporarily transforms into the File Selection Window as shown in Figure 3-18. The bar on the left can be dragged with **LMB** for scrolling. To load a file, select it with **LMB** and press **ENTER**, or simply click it with **MMB**.

Ρ	terial/ chapters/Chapter16-	-to-xype-10-11	-03/Chapter 16 - Eff	'ects/ L	OAD FILE
	Explosion00.blend				Cancel
		1 190			
		204			
	.DS_Store	12 292			
	Campfire00.blend	168 024			
	Campfire01.blend	170 452			
	Campfire02.blend	170 784			
	Campfire03.blend	137 000			
	Campfire04.blend	175 012			
	Explosion00.blend	425 872			
	Explosion01.blend	429 028			
	Explosion02.blend	429 028			
	Explosion03.blend	429 012			
	Explosion04.blend	429 012			
	Fireworks00.blend	39 772			
	Fireworks01.blend	40 308			
	Fireworks02.blend	39 940			
	Fireworks03.blend	40 624			
	Fireworks04.blend	40 948			
2 :		FILE	Free: 9675.000 Mb	Files: (0)	) 33 (0.00

Figure 3-18. File Selection Window - loading.

The upper text box displays the current directory path, and the lower one contains the selected filename. The P button (**PKEY**) moves you up to the parent directory and the button with the dash maintains a list of recently used paths. On Windows operating systems, the latter also contains a list of all drives (C:, D:, etc).

**Note:** Blender expects that you know what you are doing! When you load a file, you are not asked to save unsaved changes to the scene you were previously working on: completing the file load dialog is regarded as being enough confirmation that you didn't do this by accident. Make sure that you save your files.

# Saving files

Saving files is like loading files: When you press **F2**, the focused window temporarily changes into a File Selection Window, as shown in Figure 3-19. Click the lower edit box to enter a filename. If it doesn't end with ".blend," the extension is automatically appended. Then press **ENTER** to write the file. If a file with the same name already exists, you will have to confirm that you want to save the file at the overwrite prompt.

Cance			impleUStinished.blend
	948	40	Fireworks05.blend
	948	40	Fireworks06.blend
	764	36	Simple00.blend
	764	36	Simple01.blend
	764	36	Simple02.blend
	980	32	Simple03.blend
	468	39	Simple04.blend
	268	42	Simple05.blend
	172	40	Tornado00.blend
	820	41	Tornado01.blend
	820	41	Tornado02.blend
	988	44	tornado03.blend
	880	224	Wuschel.blend
	512	224	Wushel00.blend
	144	224	Wushel01.blend
	880	224	Wushel02.blend
	880	224	Wushel03.blend
	512 144 880 880	224 224 224 224 224	Wushel00.blend Wushel01.blend Wushel02.blend Wushel03.blend

Figure 3-19. File Selection Window - saving.

The save dialog contains a little feature to help you to create multiple versions of your work: Pressing **NUM+** or **NUM-** increments or decrements a number contained in the filename. To simply save over the currently loaded file and skip the save dialog, press **CTRL-W** instead of **F2** and just confirm at the prompt.

## Rendering

This section will give you only a quick overview of what you'll need in order to render your scene. You'll find a detailed description of all options in Chapter 17.

The render settings are in the Scene Context and Rendering Buttons Sub-context (Figure 3-20) which is reached by clicking the ..., or by pressing **F10**.

	@ <u>@ % (</u> <b>E</b>			
▼ Output				
2 /tmp/ 2 // 2 //		RENDER         Shadov         Env Map           Pano         Radio           OSA         MBLUB         100%           5         8         11         16         fr.0.500         75%         50%         25%	ANIM Do Sequence Render Daemon	Game framing settings         PAL           < SizeX: 640
Backbuf DispVi	Edge Edge Settings ew DispWin Extension	<pre>     Xparts: 1 +</pre>	PLAV          rt: 25 >           <	PNG         © Crop         PAL 16:9           Quality: 100         Frs/sec: 25 +         FULL           BW         RGB         RGBA         Unified Rend

#### Figure 3-20. Rendering options in the RenderingButtons.

For now we are only interested in the Format Panel. The size (number of pixels horizontally and vertically) and file format of the image to be created are picked here. You can set the size using the Sizex and SizeY buttons. Clicking the selection box below (in Figure 3-20, "Targa" is chosen) opens a menu with all available output formats for images and animations. For still images we might choose Jpeg, for example.

Now that the settings are complete, the scene may be rendered by hitting the REN-DER button in the Render Panel or by pressing **F12**. Depending on the complexity of the scene, this usually takes between a few seconds and several minutes, and the progress is displayed in a separate window. If the scene contains an animation, only the current frame is rendered. (To render the whole animation, see the Section called *Rendering Animations* in Chapter 17.)

If you don't see anything in the rendered view, make sure your scene is constructed properly. Does it have lighting? Is the camera positioned correctly, and does it point in the right direction? Are all the layers you want to render visible?

**Note:** A rendered image is not automatically saved to disk. If you are satisfied with the rendering, you may save it by pressing **F3** and using the save dialog as described in the Section called *Saving files*. The image is saved in the format you selected previously in the DisplayButtons.

**File Extensions:** Blender *does not* add the type extension automatically to image files! You should type the extension explicitly, if you need it.

# **User preferences and Themes**

Blender has a few options that are not saved with each file, but which apply to all of a user's files instead. These preferences primarily concern user interface handling details, and system properties like mouse, fonts, and languages.

As the user preferences are rarely needed, they are neatly hidden behind the main menu. To make them visible, pull down the window border of the menu (usually the topmost border in the screen). The settings are grouped into seven categories which can be selected with the violet buttons shown in Figure 3-21.

Display:		Snap to grid:	Menu Buttons:	Toolbox Thre:	sh.:View rotation:	Middle mouse button:	Mousewheel:	
ToolTips	Object Info	Grab	Rotate Auto Open	LMB: 5	Trackball	Rotate View Pan View	Scroll Lines: 3 →	-
Globa	I Scene	Size		← ThresB: 2 →	Turntable	Emulate 3 Buttons	Invert Wheel Zoom	]
View & Co	ontrols	Edit Methods	Language & Font	Themes	Auto Save	e System & OpenGL	. File Paths	
t = ▼ File A	dd Timeline	Game Rende	r Help 🗢 SCR:2-Mode	I X + SCE:S	icene :	× 🗟 www.blender.org 2	231 Ve:0   Fa:0   Ob:0-0   L	.a:0   M

#### Figure 3-21. User preferences window.

Because most buttons are self-explanatory or display a helpful tool-tip if you hold the mouse still over them, we won't describe them in detail here. Instead, we will just give you an overview of the preference categories:

#### View & Controls

Settings concerning how the user interface should react to user input, such as which method of rotation should be used in 3D views. Here you can also activate 3-button mouse emulation if you have a two-button mouse. **MMB** can then be input as **ALT-LMB**.

## Edit Methods

Lets you specify the details for the workings of certain editing commands like duplicate.

#### Language & Fonts

Select an alternative TrueType font for display in the interface, and choose from available interface languages.

#### Themes

Since version 2.30 Blender allows the utilization of Themes to define custom interface colors. You can create and manage themes from here.

### Auto Save

Auto saves can be set so that you will have an emergency backup in case something goes wrong. These files are named Filename.blend1, Filename.blend2, etc.

## System & OpenGL

You should consult this section if you experience problems with graphics or sound output, or if you don't have a numerical keypad and want to emulate it (for laptops). Furthermore here you can set the light scheme for Solid and shaded draw modes.

#### File Paths

Choose the default paths for various file load dialogs.

# Setting the default scene

You don't like Blender's default window set-up, or want specific render settings for each project you start, or you want to save your Theme? No problem. You can use any scene file as a default when Blender starts up. Make the scene you are currently working on the default by pressing **CTRL-U**. The scene will then be copied into a file called .B.blend in your home directory.

You can clear the working project and revert to the default scene anytime by pressing **CTRL-X**. But remember to save your changes to the previous scene first!

# Chapter 4. Your first animation in 30 + 30 minutes

This chapter will guide you step-by-step through the animation of a small "Gingerbread Man" character. We'll describe all actions completely, but we'll assume that you have read the entire Chapter 3, and that you understand the conventions used throughout this book.

In the first 30 minutes of this tutorial we'll build a *still* gingerbread man. Then, in the next 30 minutes, we'll give him a skeleton and animate a walk cycle.

# Warming up

Relevant to Blender v2.31

Let's begin.

1. Fire up Blender by double clicking its icon or running it from the command line. Blender will open showing you, from top view, the default set-up: a camera and a plane. The plane is pink, meaning it is selected (Figure 4-1). Delete the plane with **XKEY** and confirm by clicking the Erase Selected entry in the dialog which appears.



## Figure 4-1. Blender window as soon as you start it.

Now select the camera with **RMB** and press **MKEY**. A small toolbox, like the one in Figure 4-2, will appear beneath your mouse, with the first button checked. Check the rightmost button on the top row and then the OK button. This will move your camera to layer 10.

Blender provides you with 20 layers to help you organize your work. You can see which layers are currently visible from the group of twenty buttons in the 3D window toolbar (Figure 4-3). You can change the visible layer with **LMB** and toggle visibility with **SHIFT-LMB**.

		۰ II
		_

Figure 4-2. Layer control toolbox.

	Ω÷		
<li>&lt; 1</li>	•		

Figure 4-3. Layer visibility controls.

# Building the body

## Relevant to Blender v2.31

Change to the front view with **NUM1** and add a cube by pressing **SPACE** and selecting menu Add, submenu Mesh, sub-submenu Cube. (In the book we will use **SPACE**>>ADD>>Mesh>>Cube as shorthand for these kinds of actions). A cube will appear (Figure 4-4). This newly added cube is in *EditMode*, a mode in which you can move the single vertices that comprise the mesh. By default, all vertices are selected (highlighted in yellow - unselected vertices are pink).



Figure 4-4. Our cube in EditMode, all vertices selected.

We will call our Gingerbread man "Gus". Our first task is to build Gus's body by working on our cube in EditMode. To see the Blender tools that we'll use for this purpose, press the button showing a square with yellow vertices in the Button window header (Figure 4-5), or press F9.



Figure 4-5. The Edit Buttons Window button.

Now locate the Subdivide button in the Mesh Tools panel and press it once (Figure 4-6). This will split each side of the cube in two, creating new vertices and faces (Figure 4-7).

Link and Materials			▼ Mesh		
♦ ME:Cube	ME:Cube F OB:Cube				
Vertex Groups	Material		Material		<ul> <li>✓ Degr: 30 →</li> </ul>
	< 1 Ma	at: 1 🕨 ?	SubSur Catmu 🗢 TexM		
Nou Doloto	New	Delete	Subdiv: 1 > <1 >		
New Delete	Select	Deselect	Optimal		
Assign Remove	Delect	Deselect			
Select Desel.	Ass	ign	Sticky Make		
			VertCol Make Slowe		
AutoTexSpace	Set Smooth	Set Solid	TexFac Make Faste		

Figure 4-6. The Edit Buttons window for a Mesh.



Figure 4-7. The cube, subdivided once.

With your cursor hovering in the 3D window press **AKEY** to deselect all elements. Vertices will turn pink. Now press **BKEY**; the cursor will change to a couple of orthogonal grey lines. Move the cursor above the top left corner of the cube, press and hold **LMB**, then drag the mouse down and to the right so that the grey box encom-

passes all the leftmost vertices. Now release the **LMB**. This sequence, which lets you select a group of vertices in a box, is summarized in Figure 4-8.

**Box Select:** On many occasions you may have vertices hidden behind other vertices, as is the case here. Our subdivided cube has 26 vertices, yet you can only see nine because the others are hidden.

A normal **RMB** click selects only one of these stacked vertices, whereas a box select selects all. Thus, in this case, even if you see only three vertices turning yellow you have actually selected nine vertices.



Figure 4-8. The sequence of Box selecting a group of vertices.

Now press **XKEY** and, from the menu that pops up, select Vertices to erase the selected vertices (Figure 4-9).



Figure 4-9. The pop-up menu of the Delete (XKEY) action.

**Undo:** Introduced in version 2.3, Blender has a Mesh Undo feature. Pressing **UKEY** in EditMode makes Blender Undo the last Mesh edit. Keep pressing **UKEY** to roll back changes as long as the Undo buffer will allow, while **SHIFT-U** re-does changes. **ALT-U** opens a menu with a list of possible undoes, so that you can easily find the point to which you want to revert to.

Mesh Undo works only in EditMode and only for one mesh at a time. Undo data is not lost when you switch out of EditMode, only when you start editing a *different* mesh.

Another way to revert to the previously saved state is to press **ESC** in the middle of an action. This cancels the action, reverting to the previous state.

Now, using the sequence you just learned, Box Select the two top-rightmost vertices (Figure 4-10, left). Press **EKEY** and click on the Extrude menu entry to extrude them. This will create new vertices and faces which you can move and which will follow the mouse. Move them to the right.

To constrain the movement horizontally or vertically, click **MMB** while moving. You can switch back to unconstrained movement by clicking **MMB** again. Alternatively you can use **XKEY** to constrain movement to x axis, **YKEY** for y axis and so on.

Let's create Gus's arms and legs. Move these new vertices one and a half squares to the right, then click **LMB** to fix their position.

Extrude again with **EKEY** then move the new vertices another half square to the right. Figure 4-10 show this sequence.



Figure 4-10. Extruding the arm in two steps.

Gus should now have a left arm (he's facing us). We will build the left leg the same way by extruding the lower vertices. Try to produce something like that shown in Figure 4-11.

**Note:** We use the Extrude tool three times to produce the leg. We don't care about elbows, but we will need a knee later on!



Figure 4-11. Half body.

**Coincident vertices:** If you extrude, and in the process of moving you change your mind and press **ESC** to recover, the extruded vertices will still be there, in their original location! While you can move, scale, or rotate them by pressing **GKEY**, you probably don't want to extrude them again. To fully undo the extrusion, look for the Remove Doubles button, highlighted in Figure 4-12. This will eliminate coincident vertices.



Figure 4-12. The Edit Buttons window.

**Note:** The CD contains a .blend file with this example, saved at various modelling phases. The first file, <code>guickstart00.blend</code> contains what you should have obtained up to now.

Subsequent steps are numbered progressively, <code>Quickstart01.blend</code>, <code>Quickstart02.blend</code> and so on, while <code>Quickstart.blend</code> contains the final result. This standard applies to all other examples in the Book.

Now we'll create the other half of Gus:

1. Select all vertices (AKEY) and choose the 3D Cursor entry in the Rotation/Scaling Pivot Menu of the 3D Window header. (Figure 4-13).

2. Press **SHIFT-D** to duplicate all selected vertices, edges, and faces. The new objects are in Grab mode, press **ESC** to exit from this mode without moving the vertices.

3. Press **MKEY** to open the Mirror Axis Menu. Choose Global X. The result is shown in Figure 4-14.



Figure 4-13. Setting the reference center to the cursor.



Figure 4-14. Flip the copy of the half body to obtain a full body.

4. Deselect all then reselect all by pressing **AKEY** twice, then eliminate the coincident vertices by pressing the Remove doubles button (Figure 4-12). A box will appear, notifying you that eight vertices have been removed.

**Reference center:** In Blender, scaling, rotating and other mesh modifications occur either with respect to the cursor position, the object's center, or the barycenter (center of mass) of the selected items, depending upon which entry of the Rotation/Scaling Pivot Menu (Figure 4-13) is active. The crosshair button selects the cursor as reference.

**Moving the cursor:** To place the cursor at a specific grid point, position it next to where you want it and press **SHIFT-S** to bring up the Snap Menu. The entry Curs->Grid places the cursor exactly on a grid point. The Curs->Sel places it exactly on the selected object. The other entries move objects other than the cursor.

Gus Needs a head:

1. Move the cursor so that it is exactly one grid square above Gus' body (Figure 4-15, left). Add a new cube here (**SPACE**>>ADD>>Mesh>>Cube).

2. Press **GKEY** to switch to Grab Mode and move the newly created vertices down, constraining the movement with **MMB**, for about one third of a grid unit (Figure 4-15, right).



Figure 4-15. The sequence of adding the head.

3. This produces a rough figure at best. To make it smoother, locate the SubSurf Toggle Button (Figure 4-16) in the Mesh panel and switch it on. Be sure to set both the two NumButtons below to 2.

**Note:** SubSurfacing is an advanced modelling tool, it dynamically refines a given coarse mesh creating a much denser mesh and locating the vertices of the finer mesh so that they smoothly follow the original coarse mesh. The shape of the Object is still controlled by the location of the coarse mesh vertices, but the rendered shape is the smooth, fine mesh one.

4. Switch out of EditMode (**TAB**) and switch from the current default Wireframe mode to Solid mode with **ZKEY** to have a look at Gus. He should look like Figure 4-17 left.

<ul> <li>Link and Materials</li> </ul>			Mesh
ME:Cube     F     OB:Cube			Auto Smooth
Vertex Groups	Material		✓ Degr: 30 ▶
	< 1 Ma	at: 1 🕨 ?	SubSur Catmu 🗢 TexM
New Delete	New	Delete	≪Subdiv: 1 ► ≪1 ►
Assign Remove	Select	Deselect	
Select Desel.	Ass	sign	Sticky Make
AutoTexSpace	Set Smooth	Set Solid	TexFac Make Faste

Figure 4-16. The Edit Buttons window.



Figure 4-17. Setting Gus to smooth.

5. To make Gus look smooth, press the SetSmooth button in Figure 4-16. Gus will now appear smooth but with funny black lines in his middle (Figure 4-17, middle). These lines appear because the SubSurfed finer mesh is computed using information about the coarse mesh normal directions, which may not be self consistent, that is, some face normals might point outward, some inward, if extrusions and flippings have been made. To reset the normals, switch back to EditMode (**TAB**), select all vertices (**AKEY**), and press **CTRL-N**. Click with **LMB** on the Recalc normals outside box which appears. Now Gus should be nice and smooth, as shown in Figure 4-17, right.

Press **MMB** and drag the mouse around to view Gus from all angles. Oops, he is too thick! To fix that, switch to side view **NUM3**. Now, switch to EditMode (if you are not there already), then back to Wireframe mode (**ZKEY**), and select all vertices with **AKEY** (Figure 4-18, left).



Figure 4-18. Slimming Gus using constrained scaling.

Let's make Gus thinner:

1. Press **SKEY** and start to move the mouse horizontally. (Click **MMB** to constrain scaling to just one axis or press **YKEY** to obtain the same result). If you now move the mouse toward Gus he should become thinner but remain the same height.

2. The three numbers on the 3DWindow toolbar show the scaling factor. Once you constrained scaling, only one of these numbers will vary. Press and hold **CTRL**. The scale factor will now vary in discrete steps of value 0.1. Scale Gus down so that the factor is 0.2, then set this dimension by clicking **LMB**.

3. Return to Front view and to Solid mode (**ZKEY**), then rotate your view via **MMB**. Gus is much better now!

# Let's see what Gus looks like

Relevant to Blender v2.31

We're just about ready to see our first rendering, but first, we've got some work to do.

1. **SHIFT-LMB** on the top right small button of the layer visibility buttons in the 3DWindow toolbar (Figure 4-19) to make both Layer 1 (Gus's layer) and Layer 10 (the layer with the camera) visible.



Figure 4-19. Making both layer 1 and 10 visible.

**Note:** Remember that the *last* layer selected is the active layer, so all subsequent additions will automatically be on layer 10.

2. Select the camera (**RMB**) and move it to a location like (x=7, y=-10, z=7). Do this by pressing **GKEY** and dragging the camera while keeping **CTRL** pressed to move it in steps of 1 grid unit.

**Entering precise locations and rotations:** If you prefer to enter numerical values for an object's location you can do so by pressing **NKEY** and modifying the NumButtons in the Panel that appears (Figure 4-20). Remember to press or to confirm your input.

× ▼ Transform Pro	operties
OB: Camera	Par:
LocX: 8.000	▶
✓ LocY: -10.000	•
↓ LocZ: 7.000	•
<ul> <li>RotX: 90.000</li> </ul>	▶ 💽 🔹 SizeX: 1.000 →
<ul> <li>RotY: 0.000</li> </ul>	I ← SizeY: 1.000 →
<ul> <li>RotZ: 0.000</li> </ul>	▶ 🔹 SizeZ: 1.000 →

Figure 4-20. The Panel for numerical input of object position/rotation etc.

To make the camera point at Gus, keep your camera selected then select Gus via **SHIFT-RMB**. The camera should be magenta and Gus light pink. Now press **CTRL-T** and select the Old Track entry in the pop up. This will force the camera to track Gus and always point at him. This means that you can move the camera wherever you want and be sure that Gus will always be in the center of the camera's view.

**Tracking:** If the tracking object already has a rotation of its own, as is often the case, the result of the **CTRL-T** sequence might not be as expected. If it is not, select the tracking object (in our example the camera), and press **ALT-R** to remove any object rotation. Once you do this the camera will really track Gus.

Figure 4-21 shows top, front, side and camera view of Gus. To obtain a camera view press **NUM0**.



Figure 4-21. Camera position with respect to Gus.

Now we need to create the ground for Gus to stand on.

1. In top view (**NUM7**), and *out* of EditMode, add a plane (**SPACE**>>ADD>>Mesh>>Plane).

**Note:** It is important to be out of EditMode, otherwise the newly added object would be part of the object currently in EditMode, as it was for Gus' head when we added it. If the cursor is where Figure 4-21 shows, such a plane would be in the middle of Gus's body.

2. Switch to ObjectMode and Front view (**NUM1**) and move (**GKEY**) the plane down to Gus's feet, using **CTRL** to keep it aligned with Gus.

3. Switch the reference center from cursor (where we set it at the beginning) to object by pressing the highlighted button Figure 4-22.

4. Go to Camera view (**NUM0**) and, with the plane still selected, press **SKEY** to start scaling.



Figure 4-22. Set the reference center to Object center.

5. Enlarge the plane so that its edges extend beyond the camera viewing area, as indicated by the outer white dashed rectangle in Camera view.

Now, some light!

1. In Top view (**NUM7**), add a Lamp light (**SPACE**>>ADD>>Lamp) in front of Gus, but on the other side of the camera; for example in (x=-9, y=-10, z=7) (Figure 4-23).



Figure 4-23. Inserting a Lamp.

2. Switch to the Lamp Buttons in the Shading context via the button with a lamp in the Button Window toolbar (Figure 4-24) or F5.

<u> </u>	·	
	0 H O H O H	*02300

Figure 4-24. The Lamp Buttons window button.

3. In the Buttons Window, Preview Panel, press the Spot toggle button to make the lamp a Spotlight (Figure 4-25) of pale yellow (R=1, G=1, B=0.9). Adjust ClipSta: Num Button to 5, Samples: to 4, and Soft: to 8.

			<ul> <li>Texture and Input Map To</li> </ul>
Lanp Scot Sun Hemi	E LA:Lamp     Dist: 20:00     Diad     Energy 1:000 i     Dist: 20:00     Diad     Energy 1:000 i     Dist: 20:00     Dis	Spot8 45.00         Image: Control of Contro of Control of Control of Contro of Control of Control o	Add New           Glob         Mew           Glob         Mew           Glob         State           Glob         State           Glob         State           Glob         State           Glob         State           Glob         State

Figure 4-25. Spot light settings.

4. Make this spotlight track Gus just as you did for the camera by selecting Spot, **SHIFT**, then Gus, then by pressing **CTRL-T**. If you added the spot in Top View you should not need to clear its rotation via **ALT-R**.

5. Add a second lamp in the same location as the spot, and again in Top View, with (**SPACE**>>ADD>>Lamp). Make this lamp a Hemi lamp with energy of 0.6 (Figure 4-26).

			Texture and Input Map To
Lanp Spot Sun Hemi	E LA:Lamp.001     Dist: 20.00     Diad     Energy 0.500     Sphere     R 1.000     Diad     Dist: 20.00     Diad	Spot 84 0.0	Add New           Glob         View         Object           Glob         Glob         View           Glob         View         Object           Glob         Glob         View           Glob         Glob         View           Glob         Glob         Glob           Glob

Figure 4-26. The Hemi lamp settings

**Two lamps?:** Use two or more lamps to help produce soft, realistic lighting, because in reality natural light never comes from a single point. You will learn more about this in Chapter 12.

We're almost ready to render. As a first step, go to the Scene context and Render buttons by pressing the image-like icon in the Button window toolbar (Figure 4-27) or **F10**.



Figure 4-27. The Rendering buttons window buttons.

In the Render Buttons, Format Panel, set the image size to 640x480 with the Num buttons at the top right. In the Render Panel set the Shadows Toggle Button top center to On, and the OSA Toggle Button center-left to On as well (Figure 4-28). These latter controls will enable shadows and oversampling (OSA) which will prevent jagged edges.

▼ Output	<ul> <li>Render</li> </ul>
/tmp/	RENDER
🖉 //ftype	Blender Internal
Backbuf Edge Edge Settings	OSA MBLU 5 8 11 16 Bf: 0.5
DispView DispWin Extension	<ul> <li>✓ Xparts: 1 → &lt; Yparts: 1</li> <li>Sky Premul Key</li> </ul>

Figure 4-28. The Rendering Buttons window

Now press the RENDER button or **F12**. The result, shown in Figure 4-29, is actually quite poor. We still need materials, and lots of details, such as eyes, and so on.



Figure 4-29. Your first rendering. Congratulations!

**Saving:** If you have not done so already, now would be a good time to save your work, via the File>>Save menu shown in Figure 4-30, or **CTRL-W**. Blender will warn you if you try to overwrite an existing file.

Blender does automatic saves into your system's temporary directory. By default, this happens every four minutes and the file name is a number. Loading these saves is another way to undo unwanted changes.

$\bigtriangledown$	File	Add	Timeline	Game	Render
	Ne	ew			Ctrl X
	0	pen			F1
	Re	eopen	Last		Ctrl O
	Sa	ave			Ctrl W
	Sa	ave As			F2
	Sa	ave Ima	age		F3
	Sa	Ctrl U			
	A	Shift F1			
	Im	port			+
	E>	<port< th=""><td></td><td></td><td>+</td></port<>			+
	Pa	ack Da	ta		
	Ur				
	Q	uit Bler	nder		Q

Figure 4-30. The Save menu.

# **Materials and Textures**

Relevant to Blender v2.31

It's time to give Gus some nice cookie-like material:

1. Select Gus. Then, in the Button Window header, select the Shading Context by pressing the red dot button (Figure 4-31) or using the F5 key.

·		
G≣OŁ∏∎	▓❹ॼॡ⊕	4 1 →

Figure 4-31. The Material Buttons window Button.

2. The Button window will be almost empty because Gus has no materials yet. To add a material, click on the Menu Button in the Material Panel (the one with two triangles, pointing up and down) and select Add New (Figure 4-32).

	Material	
\$	Add New	
ME	:Cube	OB ME <

Figure 4-32. The Material Menu button.

3. The Buttons window will be populated by Panels and Buttons and a string holding the Material name, "Material" by default, will appear next to the white square button. Change this to something meaningful, like GingerBread.

4. Modify the default values as per Figure 4-33 to obtain a first rough material.

Image: Sector 2000         Image: Sector 2000         Image: Sector 2000           ME: Cube         0.0         M:0         Image: Sector 2000           VC41         Ug41         VC41         Sector 2000           0         0.0         10.074         Image: Sector 2000           0         0.00         10.074         Image: Sector 2000           MM         Sector 2000         Image: Sector 2000         Image: Sector 2000           FIG: H6V DVH         Sector 2000         Sector 2000         Image: Sector 2000	Lambert         Flar         0.800         Halo           CookTorr         Spec 0.050         Redo         2040/rs           Hard S         Vite         Vite         707/rs           Ziranop         Enviro         Vite         707/rs           Anto 0.5         Enviro 1.1         Cirry Shad         10           Adds 0.0         Zorffs: 0.000         Zirrero         Zirrero	Add Nev

Figure 4-33. The Material Buttons window and a first gingerbread material.

5. Press the Menu Button in the Textures Panel area (Figure 4-34) and select Add new. We're adding a texture in the first channel. Call it "GingerTex."

•	Texture	
		Add New

Figure 4-34. The Textures menu button in the Material Buttons

6. Select the Texture Buttons by clicking the button in Figure 4-35 or by pressing F6.



Figure 4-35. The Texture Buttons window Button.

7. From the columns of ToggleButtons which appear in the Texture panel select Stucci and set all parameters as in Figure 4-36.

	<ul> <li>Texture</li> </ul>	Colors	
Mat	<ul> <li>TE:GingerTex</li> </ul>	XQF	Plastic Wall In Wall Out
World	GingerTex	None	Soft noise Hard noise
Lamp	Grain	Image EnvMap	NoiseSize : 0.020
			✓ Turbulence: 5.00 ▶
		Clouds Marble	
		Magic Blend	
Default Var		Noise Plugin	

Figure 4-36. The Texture Buttons window with a stucci texture.

8. Return to the Material buttons (F5) and set the Map Input and Map To tabs of the Texture Panel as in Figure 4-37. Release the Col Toggle Button and set the Nor Toggle Button, then raise the Nor slider to 0.75. These changes will make our Stucci texture act as a "bumpmap" and make Gus look more biscuit-like.

🔻 Map Input	🔻 Мар То
UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
Flat         Cube         ofsX 0.000           Tube         Sphe         ofsY 0.000           ofsZ 0.000         ofsZ 0.000	Stencil Neg No RGB
X         Y         Z         ≤         sizeX 1.00 →           X         Y         Z         ≤         sizeY 1.00 →           X         Y         Z         ≤         sizeY 1.00 →	G 0.000         Col 1.000           B 1.000         Nor 0.750           DVar 1.00         Var 1.000

Figure 4-37. Settings for the Stucci texture in the Material Buttons window.

9. Now add a second texture, name it "Grain," and make it affect only the Ref property with a 0.4 Var (Figure 4-38). The texture itself is a plain Noise texture.



Figure 4-38. Settings for an additional Noise texture in channel 2.

Give the ground an appropriate material, such as the dark blue one shown in Figure 4-39.



Figure 4-39. A very simple material for the ground.

To give some finishing touches we'll add eyes and some other details.

1. First make Layer 1 the only one visible by clicking with **LMB** on the layer 1 button (Figure 4-40). This will hide the lamps, camera, and ground.



Figure 4-40. Layer visibility buttons on toolbar.

2. Place the cursor at the center of Gus's head. (Remember that you are in 3D so be sure to check at least two views to be sure!)

3. Add a sphere (**SPACE**>>ADD>>Mesh>>UVsphere). You will be asked for the number of Segments: (meridians) and Rings: (parallels) into which to divide the sphere. The default of 32 is more than we need here, so use a value of 16 for both. The sphere is in the first image at the top left of the sequence in Figure 4-41.

4. Scale the sphere down (**SKEY**) to a factor 0.1 in all dimensions, then switch to side view (**NUM3**) and scale it only in the horizontal direction (**YKEY**) a further 0.5 (see the second two images in Figure 4-41).



Figure 4-41. Sequence for creation of the eyes.

5. Zoom a little if necessary via NUM+, MW, or CTRL-MMB, and drag the sphere (GKEY) to the left so that it is halfway into the head (as shown in the first image in the second row of Figure 4-41).

6. Return to front view (**NUM1**) and move the sphere sideways, to the right. Place it where Gus should have an eye.

7. Flip a duplicate around the cursor by following the sequence you learned when flipping Gus's body. (Select the crosshair toolbar button, in EditMode **AKEY** to select all, **SHIFT-D**, **ESC MKEY**, Global X Menu entry). Now Gus has two eyes.

8. Exit EditMode (**TAB**), and place the cursor as close as you can to the center of Gus's face. Add a new sphere and scale and move it exactly as before, but make it smaller and place it lower than and to the right of the cursor, centered on the SubSurfed mesh vertex Figure 4-42).



Figure 4-42. Creating a mouth with Spinning tools.

9. Now, in the Edit Buttons (F9), locate the group of buttons at bottom in the Mesh Tools Panel (Figure 4-43). Set Degr: to 90, Steps: to 3, and verify that the Clock-wise: TogButton is on. Then, with all vertices still selected, press SpinDup. This will create three duplicates of the selected vertices on an arc of 90 degrees, centered around the cursor. The result should be Gus's mouth, like the last image of the sequence shown in Figure 4-42.

▼ Mesh Tools						
Beauty	Subdivide	Fract Subd				
Noise	Hash	Xsort				
To Sphere	Smooth	Split				
Flip Normals	Flip Normals Rem Double Limit: 0.001					
	Extrude					
Screw	Spin	Spin Dup				
l∢ Degr: 90⊩	▲ Degr: 90 ▲ Steps: 3 ▲ Turns: 1 ▲					
Keep Original Clockwise						
Extrude Dup 💽 🔍 Offset: 1.000 🕨						

Figure 4-43. The Spin Tools buttons in the Edit Buttons window.

Now that you have learned the trick, add three more of these ellipsoids to form Gus's buttons. Once you have made one button, you can simply exit EditMode, press **SHIFT-D** to create a duplicate, and move the duplicate into place, as shown in Figure 4-44.



Figure 4-44. The complete Gus!

Give the eyes a chocolate-like material, like the one shown at the top in Figure 4-45. Give the mouth a white sugar like material, like the second one shown in Figure 4-45, and give the buttons a red, white, and green sugar like material. These are shown from top to bottom in Figure 4-45 too.



Chapter 4. Your first animation in 30 + 30 minutes

Figure 4-45. Some other candy materials.

**Objects sharing a material:** To give one object the same material as another object, select that material in the Menu list which appears when you press the Menu Button Button Window Material Panel.



Figure 4-46. Selecting an existing material from the Material Menu.

Once you have finished assigning materials, make layer 10 visible again (you should know how), so that lights and the camera also appear, and do a new rendering (**F12**). The result should look more or less like Figure 4-47.



Figure 4-47. The complete Gus still rendering.

Save your image, if you so wish, by pressing **F3**. Enter the name of your image in the file window and save.

**Image types and extension:** You must choose the image format (JPEG, PNG, and so on) by setting it in the Rendering buttons *before* pressing **F3** (Figure 4-27) and using the Menu (Figure 4-48) in the Format Panel. Blender does *not* add an extension to the file name; you must enter one if you wish.

▼ Format						
Game framing :	setting	\$ >>		PAL		
SizeX: 640	A Siz	eV·480 ⊾		NTSC		
4 01267. 040 P	1 012	61.400 P		Default		
< AspX: 100 ▶	≛ As	pY:100 🕨		Preview		
				PC		
·				PAL 16:9		
PNG		Crop		PANO		
∢ Quality: 90 ►	≺Frs	/sec: 25 🕨		FULL		
BW R	GB	RGBA		Unified Rend		

Figure 4-48. File type selection menu in the Rendering Buttons window.

# Rigging

## Relevant to Blender v2.31

If we were going for a still picture, our work up to this point would be enough, but we want Gus to move! The next step is to give him a skeleton, or Armature, which will move him. This is called the fine art of rigging. Gus will have a very simple rigging: four limbs (two arms and two legs) and a few joints (no elbows, only knees), but no feet or hands. To add the rigging:

1. Set the cursor where the shoulder will be, press **SPACE**>>Add>>Armature. A rhomboidal object will appear, a bone of the armature system, stretching from cursor to mouse pointer. Place the other end of the armature in Gus's hand (Figure 4-49) with **LMB**. This will fix the bone and create a new one from the end point of the previous one, producing a bone chain. We don't need any other bones right now, so press **ESC** to exit.



Figure 4-49. Adding the first bone, an elbowless arm.

2. Stay in EditMode, then move the cursor to where the hip joint will be and add a new bone (**SPACE**>>ADD>>Armature) down to the knee. Press **LMB** and a new bone should automatically appear there. Stretch this bone down to the foot (Figure 4-50).



Figure 4-50. Adding the second and third bones, a leg bone chain.

**Bone position:** The bones we are adding will deform Gus's body mesh. To produce a neat result, try to place the bone joints as shown in the illustrations.

3. Now place the cursor in the center and select all bones with **AKEY**. Duplicate them with **SHIFT-D** and exit grab mode with **ESC** then flip them with **MKEY** relatively to the cursor and Global X axis as you did with meshes (Figure 4-51).



Figure 4-51. The complete armature after duplicating and flipping.

Once you've selected all of the bones (AKEY), the Edit Buttons window should show an Armature Bones Panel which contains the Armature buttons (Figure 4-52).

AR:Armature     F     OB:Armature	Rest Pos Drav Axes Draw Nanes X-Ray	Selected Bones Hide BO-Arm L child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-UpLeg L child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-Arm R child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-Arm R child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-UpLeg R child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-UpLeg R child of a Skinnable a Dist 1.00 Weight 1.00 Hide BO-UpLeg R child of UpLeg R a M

Figure 4-52. The Edit Buttons window for an armature.

Press the Draw Names button to see the names of the bones, then **SHIFT-LMB** on the names in the Edit Button window (Figure 4-52) to change them to something appropriate like Arm.R, Arm.L, UpLeg.R, LoLeg.R, UpLeg.L and LoLeg.L. Exit EditMode with (**TAB**).

**Naming Bones:** It is very important to name your bones with a trailing '.L' or '.R' to distinguish between left and right ones, so that the Action editor will be able to automatically flip your poses.

# Skinning

Relevant to Blender v2.31

Now we must make it so that a deformation in the armature causes a matching deformation in the body. We do this with Skinning, which assigns vertices to bones so that the former are subject to the latter's movements.

1. Select Gus's body, then **SHIFT** select the armature so that the body is magenta and the armature is light pink.

2. Press **CTRL-P** to parent the body to the armature. A pop up dialog will appear (Figure 4-53). Select the Use Armature entry.



Figure 4-53. The pop-up menu which appears when parenting an Object to an Armature.

3. A new menu appears, asking if you want Blender to do nothing, create empty vertex groups, or create and populate vertex groups (Figure 4-54).



Figure 4-54. Automatic Skinning options.

4. We'll use the automatic skinning option. Go ahead and select Create From Closest Bones.

Now select only Gus's body and go to EditMode (**TAB**). You will notice in the Edit Buttons (**F9**) Window and Mesh Tools 1 Panel, the presence of a Vertex Group menu

and buttons (Figure 4-55).

Image: ME: Dube         F         DB: Dube           Vertex: Croups         Ginger@read           Image: Merced         Imager@read           Image: Merced         Imager@read           New         Delete           Select         Desetect           Select         Desetect           AutoTex:Space         Set Smoot           Set Snoot         Set Solid	Auto Smooth Beg: 30 Subdy: 2 2 2 Optimal Sticky Make Fortron Make	Beauty Studivide Fract Sound Notes Hash Xsort To Schere Smooth Split Fig.Norma5 Fem.Double Limit: 0009 Extrude Screw Spin Spin Dup Degr: 90: Steps; 9: Turns: 1 Repp Original Extrude Dup Offset: 1,000 s	Centre           Hide         Reveal           Select Swap            Draw Kormals            Draw Fores            Draw Fores            All edges

Figure 4-55. The vertex groups buttons in the Edit Buttons window of a mesh.

By pressing the Menu Button a menu with all available vertex group pops up (six in our case, but a truly complex character, with hands and feet completely rigged, can have tens of them! Figure 4-56). The buttons Select and Deselect show you which vertices belong to which group.

# ⇒ マ	View S	Select	Mesh	🛕 Edit M	
LoLeg.R		GE	0 K		
Arm.B LoLeg.L	:l Material: be	F	OB:C	ube	
Arm.L	jups		GingerBre	ad	
Arm.B Weight: 1,000		JUL	✓ 1 Mat: 1 → ?		
New	Delete		lew	Delete	
Assign	Remove		elect	Deselect	
Select	Desel.		Assign		
AutoTe	Set	Smoot	Set Solid		

Figure 4-56. The menu with the vertex groups automatically created in the skinning process.

Select the Right arm (Arm.R) group and, with all vertices de-selected (AKEY, if needed) press Select. You should see something like Figure 4-57.



Figure 4-57. Gus in EditMode with all the vertices of group Arm.R selected.

The vertices marked with yellow circles in Figure 4-57 belong to the deformation group, but they should not. The autoskinning process found that they were very close to the bone so it added them to the deformation group. We don't want them in this group because, since some are in the head and some are in the chest, adding them to the deformation group would deform those body parts. To remove them from the group, deselect all the other vertices, those which should *remain* in the group using Box selection (**BKEY**), but use **MMB**, not **LMB**, to define the box, so that all vertices within the box become deselected.

Once only the 'undesired' vertices are selected, press the Remove button (Figure 4-55) to eliminate them from group Arm.R.

Deselect all (**AKEY**) then check another group. Check them all and be sure that they look like those in Figure 4-58.



Figure 4-58. The six vertex groups.

**Vertex groups:** Be very careful when assigning or removing vertices from vertex groups. If later on you see unexpected deformations, you might have forgotten some vertices, or placed too many in the group. You can modify your vertex groups at any time.

**Other details:** Our deformations will affect only Gus's body, not his eyes, mouth, or buttons, which are separate objects. While this is not an issue to consider in this simple animation, it's one that must be taken into account for more complex projects, for example by parenting or otherwise joining the various parts to the body to make a single mesh. (We'll describe all of these options in detail in later Chapters).
## Posing

Relevant to Blender v2.31

Once you have a rigged and skinned Gus you can start playing with him as if he were a doll, moving his bones and viewing the results.

1. Select the armature only, then select Pose Mode from the "Mode" Menu (Figure 4-59). This option only appears if an armature is selected.



Figure 4-59. The toggle button to switch to pose mode in the 3D Window toolbar.

2. The armature will turn blue. You are in Pose Mode. If you now select a bone it will turn cyan, not pink, and if you move it (**GKEY**), or rotate it (**RKEY**), the body will deform!



Figure 4-60. You are in pose mode now!

**Original position:** Blender remembers the original position of the bones. You can set your armature back by pressing the RestPos button in the Armature Edit Buttons (Figure 4-52).

**Forward and Inverse Kinematics:** While handling bones in pose mode you will notice that they act as rigid, inextensible bodies with spherical joints at the end. You can actually grab only the first bone of a chain and all the other will follow it. All subsequent bones in the chain cannot be grabbed and moved, you can only rotate them, so that the selected bone rotates with respect to the previous bone in the chain while all the subsequent bones of the chain follow its rotation.

This procedure, called *Forward Kinematics (FK)* is easy to follow, but it makes precise location of the last bone of the chain difficult. We can use another method, *Inverse Kinematics (IK)* where you actually define the position of the *last* bone in the chain, and all the other assume a position, automatically computed by Blender, to keep the chain without gaps. Hence precise positioning of hands and feet is much easier.

We'll make Gus walk by defining four different poses relative to four different stages of a stride. Blender will do the work of creating a fluid animation.

1. First, verify that you are at frame 1 of the timeline. The frame number appears in a NumButton on the right of the Buttons Window Toolbar (Figure 4-61). If it is not set to 1, set it to 1 now.



Figure 4-61. The current frame Num Button in the Buttons window Toolbar.

2. Now, by rotating only one bone at a time (**RKEY**), we'll raise UpLeg.L and bend LoLeg.L backwards while raising Arm.R a little and lowering Arm.L a little, as shown in Figure 4-62.



Figure 4-62. Our first pose.

3. Select all bones with **AKEY**. With the mouse pointer on the 3D Window, press **IKEY**. A menu pops up (Figure 4-63). Select LOCROT from this menu. This will get the position and orientation of all bones and store it in a pose at frame 1. This pose represents Gus in the middle of his stride, while moving his left leg forward and above the ground.

Insert Key
Loc
Rot
Size
LocRot
LocRotSize
Avail

Figure 4-63. Storing the pose to the frame.

4. Now move to frame 11 either by entering the number in the NumButton or by pressing **UPARROW**. Then move Gus to a different position, like Figure 4-64, with his left leg forward and right leg backward, both slightly bent. Gus is walking in place!



Figure 4-64. Our second pose.

5. Select all bones again and press **IKEY** to store this pose at frame 11.

6. We now need a third pose at frame 21, with the right leg up, because we are in the middle of the other half of the stride. This pose is the mirror of the one we defined at frame 1. Therefore, return to frame 1 and, in the Armature Menu in the 3D Window header select the Copy Pose entry. (Figure 4-65). You have copied the current pose to the buffer.



Figure 4-65. Copying the pose to the buffer

7. Go to frame 21 and paste the pose with the Paste Flipped Pose option in the Armature Menu (Figure 4-66). This button will paste the cut pose, exchanging the positions of bones with suffix .L with those of bones with suffix .R, effectively flipping it!

Insert Keyframe	1		
Paste Flipped Pose			
Paste Pose	1		
Copy Current Pose	1		
Transform	•		
Transform Properties N			
Armature 😢 Pose Mode 🗢			

Figure 4-66. Pasting the copy as a new, flipped, pose.

The pose is there but it has not been stored yet! You must press **IKEY** with all bones selected.

8. Now apply the same procedure to copy the pose at frame 11 to frame 31, also flipping it.

9. To complete the cycle, we need to copy the pose at frame 1 *without* flipping to frame 41. Do so by copying it as usual, and by using Paste Pose entry. End the sequence by storing the pose with **IKEY**.

**Checking the animation:** To preview your Animation, set the current frame to 1 and press **ALT-A** in the 3D window.

## Gus walks!

Relevant to Blender v2.31

The single step in-place is the core of a walk, and once you have defined one there are techniques to make a character walk along a complex path. But, for the purpose of our Quick Start, this single step in-place is enough.

Change to the Rendering Buttons (**F10**) and set the animation start and end to 1 and 40 respectively (Figure 4-67). Because frame 41 is identical to frame 1, we only need to render frames from 1 to 40 to produce the full cycle.

2 //render/	RENDER Shado EnvMa Pano Radio	ANIM	Game framing settings >>> PAL SizeX: 640+ SizeV: 480+
	OSA MBLUR 100% 5 8 11 16 f: 0.500 75 50% 25%	Do Sequence Render Daemon	
Backbuf Edge Edge Settings	<pre>« Xparts: 1 » «Vparts: 1» Fields Od x</pre>	PLAY TT: 25	AVI Raw + Crop PAL 16:9 PANO Quality: 95 + Frs/sec: 25 FULL
DispView DispWin Extensio	Sky Premul Key Border Gamma	< Sta: 1 → < End: 40 →	BW RGB RGBA Unified Ren

Figure 4-67. Setting the Rendering Buttons for an animation.

2. Select AVI Raw as the file type in Format Panel (Figure 4-67). While this is generally not the best choice, mainly for file size issues (as will be explained later on), it is fast and it will run on any machine, so it suits our needs. (You can also select AVI Jpeg to produce a more compact file, but using lossy Jpeg compression and obtaining a movie that some external render might not be able to play).

3. Finally, press ANIM button in Anim Panel. Remember that *all* the layers that you want to use in the animation must be shown! In our case, these are layers 1 and 10.

**Stopping a Rendering:** If you make a mistake, like forgetting to set layer 10 to on, you can stop the rendering process with the **ESC** key.

Our scene is pretty simple, and Blender will probably render each of the 40 images in a few seconds. Watch them as they appear.

**Stills:** Of course you can always render each of your animation frames as a still by selecting the frame you wish to render and pressing the RENDER button.

Once the rendering is complete you should have a file named 0001\_0040.avi in a render subdirectory of your current directory - the one containing your .blend file. You can play this file directly within Blender by pressing the Play button beneath the ANIM button (Figure 4-67). The animation will automatically cycle. To stop it press **ESC**.

We have produced only a very basic walk cycle. There is much more in Blender, as you'll soon discover!

## Chapter 5. ObjectMode

### By Martin Kleppmann

The geometry of a Blender scene is constructed from one or more Objects: Lamps, Curves, Surfaces, Cameras, Meshes, and the basic objects described in the Section called *Basic Objects* in Chapter 6. Each object can be moved, rotated and scaled in *ObjectMode*. For more detailed changes to the geometry, you can work on the mesh of an Object in *EditMode* (see the Section called *EditMode* in Chapter 6).

Once you've added a basic object via **SPACE**>>Add menu, Blender changes into EditMode by default if the Object is a Mesh, a Curve or a Surface. You can change to ObjectMode by pressing **TAB**. The object's wireframe, if any, should now appear pink, meaning that the object is now selected and active.

## Selecting objects

#### Relevant to Blender v2.31

To select an object, click it with the **RMB**. To select multiple objects, hold down **SHIFT** and click with the **RMB**. Generally, the last object to be selected becomes the *active* one: It appears in light pink, whereas the non-active selected objects appear purple. The definition of the active object is important for various reasons, including parenting.

To deselect the active object, click it again with **RMB**, if multiple Objects are selected hold **SHIFT** to maintain the other selected. Press **AKEY** to select all objects in the scene (if none are currently selected) or to deselect all (if one or more is selected).

**BKEY** activates *Border select*. Use Border select to select a group of objects by drawing a rectangle while holding down **LMB**. You will select all objects that lie within or touch this rectangle.

**Note:** Border select adds to the previous selection, so to select only the contents of the rectangle, deselect all with **AKEY** first. Use **MMB** while you draw the border to deselect all objects within the rectangle.

## Moving (translating) objects

### Relevant to Blender v2.31

To move groups of objects, press **GKEY** to activate *Grab mode* for all selected objects. The selected objects will be displayed as white wireframes and can be moved with the mouse (without pressing any mouse buttons). To confirm the new position, click **LMB** or press **ENTER**; to cancel Grab mode, click **RMB** or press **ESC**. The header of the 3D Window displays the distance you are moving.

To lock movement to an axis of the global coordinate system, enter Grab mode, move the object roughly along the desired axis, then press **MMB**. To deactivate locking press **MMB** again. As a new 2.3 feature you can constrain movement to a given axis by pressing **XKEY**, **YKEY** or **ZKEY**. A single key constrains movement to the corresponding *global* axis, as **MMB** does. A second keypress of the *same* key constrains movement to the corresponding Object *local* axis. A third keypress of the same key removes constraints. Lines are drawn to let you better visualize the constraint.

Once grabbing is activated you can enter the Object translation numerically by simply typing in a number. This will let you enter the first co-ordinate shown in the 3D Window header. You can change co-ordinate with **TAB** use **NKEY** to exit/re-start numeric input mode, **ENTER** to finalize and **ESC** to exit. **BACKSPACE** will return

#### Chapter 5. ObjectMode

to original values. Please note that you must use the keyboard .KEY not the NUM. for decimals.

If you keep **CTRL** pressed while moving the object you will activate *snap mode*, and the object will move by a whole number of Blender units (grid squares). Snap mode ends when you release **CTRL** so be sure to confirm the position before releasing it.

The location of selected objects can be reset to the default value by pressing ALT-G.

**Note:** If you are striving for very fine and precise positioning, keep **SHIFT** pressed as you move. This way a large mouse movement will translate to a small object movement, which allows for fine tuning.

**Blender Gesture System:** You can also enter Grab mode by drawing a straight line while holding down LMB.

## **Rotating objects**

#### Relevant to Blender v2.31

To rotate objects, activate Rotate mode by pressing **RKEY**. As in Grab mode, you can change the rotation by moving the mouse, confirm with **LMB**, or **ENTER** cancel with **RMB** or **ESC**.

Rotation in 3D space occurs around an axis, and there are various ways to define this axis. Blender defines an axis by direction and a point that it passes through. For example, by default, the direction of the axis is orthogonal to your screen.

If you are viewing the scene from the front, side, or top, the rotation axis will be parallel to one of the global coordinate system axes. If you are viewing the scene from an angle, the rotation axis is angled too, which can easily lead to a very odd rotation of your object. In this case, you may want to keep the rotation axis parallel to the coordinate system axes. Toggle this behaviour by pressing **MMB** during Rotate mode and watch the angle display in the window header.

Alternatively, once you are in rotate mode, you can press **XKEY**, **YKEY** or **ZKEY** to constrain rotation along that axis of the *global reference*. By pressing **XKEY-XKEY** (**XKEY** twice) you constrain rotation around the x axis of the Object *local reference*. The same is true for double **YKEY** and **ZKEY**. As for Grab, a third keypress removes constraints.

It is possible to have a numerical imput for rotation exactly as it was for translations.

Select the point for the rotation axis to pass through with the pertinent menu in the header of the 3D window, as discussed below. (Figure 5-1).



Figure 5-1. The rotation point selection buttons

- *Bounding Box Center* the axis passes through the center of the selection's bounding box. (If only one object is selected, the point used is the center point of the object, which might not necessarily be in the geometric center. In Figure 5-1 it is on the middle of the rightmost edge, marked by a purple dot. For more on this point, see the Section called *EditMode* in Chapter 6.)
- *Median Point* the axis passes through the median point of the selection. This difference is only relevant in EditMode, and the 'Median' point is the barycentrum of all vertices.
- *3D Cursor* the axis passes through the 3D cursor. The cursor can be placed anywhere you wish before rotating. You can use this option to easily perform certain translations the at the same time that you rotate an object.
- *Individual Object Centers* each selected object receives its own rotation axis, all mutually parallel and passing through the center point of each object, respectively. If you select only one object, you will get the same effect as with the first button.

If you're just getting started with rotation, don't worry too much about the foregoing details. Just play around with Blender's tools and you'll get a feeling for how to work with them.

Keeping **CTRL** pressed switches to snap mode. In snap mode rotations are constrained to  $5\ddot{\imath}_{\ell}^{1/2}$  steps. Keeping **SHIFT** pressed allows fine tuning here too. The rotation of selected objects can be reset to the default value by pressing **ALT-R**.

Blender Gesture System: You can also enter Rotate mode by drawing a circular line while holding down LMB.

## Scaling/mirroring objects

#### Relevant to Blender v2.31

To change the size of objects, press **SKEY**. As in grab mode and rotate mode, scale the objects by moving the mouse, confirm with **LMB** or **ENTER**, and cancel with **RMB** or **ESC**.

Scaling in 3D space requires a center point. This point is defined with the same buttons as the axis' supporting point for rotation (Figure 5-1). If you increase the size of

#### Chapter 5. ObjectMode

the object, all points are moved away from the selected center point; if you decrease it, all points move towards this point.

By default, the selected objects are scaled uniformly in all directions. To change the proportions (make the object longer, broader, and so on), you can lock the scaling process to one of the global coordinate axes, just as you would when moving objects. To do so, enter scale mode, move the mouse a bit in the direction of the axis you want to scale, then press **MMB**. To return to uniform scaling, press **MMB** again. You will see the scaling factors in the header of the 3D window.

Again all considerations on constraining to given axis made for Grabbing still holds, as well as those on numerical input.

Here again **CTRL** switches to snap mode, with discrete scaling at 0.1 steps. Press **SHIFT** for fine tuning. The scaling of selected objects can be reset to the default value by pressing **ALT-S**.

Mirroring objects is a different application of the scale tool. Mirroring is effectively nothing but scaling with a negative factor in one direction. To mirror in the direction of the X or Y axes, press **SKEY** to go to scaling mode, then **NKEY** to switch to numeric input. Select the desired coordinates and enter '-1' as scaling factor.

**Blender Gesture System:** You can also enter scale mode by drawing a V-shaped line while holding down **LMB**.

## **Transform Properties Panel**

### Relevant to Blender v2.31

Say you want to display the position/rotation/scaling of your object in numbers. Or, you want to enter the location, rotation, and scaling values for an object directly at once. To do so, select the object you want to edit and press **NKEY**. The Transform Properties Panel (Figure 5-2) is displayed. **SHIFT-LMB**-click a number to enter a value, then press OK to confirm the changes or move the mouse outside the window to cancel.



### Figure 5-2. The number dialog

The panel also displays the Object name in the OB: Button. You can edit it from here.

## Duplicate

Relevant to Blender v2.31

To duplicate an object, press **SHIFT-D** to create an identical copy of the selected objects. The copy is created at the same position, in Grab mode.

This is a new object except that it shares any Material, Texture, and IPO with the original. These attributes are linked to both copies and changing the material of one object also changes the material of the other. (You can make separate materials for each, as described in the Materials Chapter.)

You can create a *Linked Duplicate* rather than a real duplicate by pressing **ALT-D**. This will create a new object with *all* of its data linked to the original object. If you modify one of the linked objects in EditMode, all linked copies are also modified.

# Parenting (Grouping)

Relevant to Blender v2.31

To create a group of objects, you must first make one of them the *parent* of the others. To do so, select at least two objects, press **CTRL-P**, and confirm on the dialog Make Parent?. The *active* object will be made the parent of all the others. The center of all children is now linked to the center of the parent by a dashed line. At this point, grabbing, rotating, and scaling the parent will do the same to the children being grabbed, rotated and scaled likewise.

Parenting is a very important tool with many advanced applications, as we'll see in later chapters.

Press **SHIFT-G** with an active object to see the Group Selection menu (Figure 5-3). This contains:

- Children Selects all the active objects' children, and the children's children, up to the last generation.
- Immediate Children Selects all the active objects' children but not these latter's children.
- Parent Selects the parent of the active object.
- Objects on shared layers This actually has nothing to do with parents. It selects all objects on the same layer(s) of the active object.



**Figure 5-3. Group Select** 

Move the child to the parent by clearing its origin (select it and press **ALT-O**). Remove a parent relation via **ALT-P**. You can (Figure 5-4):

- Clear parent Frees the children, which return to their *original* location, rotation, and size.
- Clear parent...and keep transform Frees the children, and *keeps* the location, rotation, and size given to them by the parent.
- Clear parent inverse Places the children with respect to the parent as if they were placed in the Global reference. This effectively clears the parent's transformation from the children.



Figure 5-4. Freeing Children

## Tracking

### Relevant to Blender v2.31

To make an object rotate so that it faces another object, and keep this facing even if either object is moved, select at least two objects and press **CTRL-T**. A dialog appears asking if you want to use a Track *constraint* or the old (Pre-2.30) track system. The Track constraint will be analyzed in the Section called *Constraints* in Chapter 16 and is the preferred method.

Here we will briefly treat the old track system, so, let's assume you have selected old Track in the dialog. By default the inactive object(s) now track the active object so that their local y axis points to the tracked object. However, this may not happen if the tracking object already has a rotation of its own. You can produce correct tracking by canceling the rotation (**ALT-R**) of the tracking object.

The orientation of the tracking object is also set so that the z axis is upward. To change this, select the tracking object, change the Button Window to Object Context ( 2, or F7) and select the track axis from the first row of six radio buttons and the upward-pointing axis from the second in the Anim Setting panel. (Figure 5-5).



Figure 5-5. Setting track axis.

To clear a track constraint, select the tracking object and press **ALT-T**. As with clearing a parent constraint, you must choose whether to lose or save the rotation imposed by the tracking.

## **Other Actions**

Relevant to Blender v2.31

### Erase

Press **XKEY** or **DEL** to erase the selected objects. Using **XKEY** is more practical for most people, because it can easily be reached with the left hand on the keyboard.

### Join

Press **CTRL-J** to join all selected objects to one single object. (The objects must be of the same type.) The center point of the resulting object is obtained from the previously *active* object.

### Select Links

Press **SHIFT-L** to select all objects sharing a link with the active one. You can select objects sharing an IPO, data, material, or texture link (Figure 5-6).



Figure 5-6. Selecting links.

## **Boolean operations**

### Relevant to Blender v2.31

Boolean operations are particular actions which can be taken only on mesh type objects. While they will work for all Mesh objects, they are really intended for use with solid, closed objects with a well defined interior and exterior region. Thus, it is very important to define the normals in each object consistently, that is all each normal of each face should point outward. See Chapter 6 for further info on normals and on why you can end up with normals pointing partuially outward and partially inward.

In the case of open objects, the interior is defined mathematically by extending the boundary faces of the object to infinity. As such, you may find that you get unexpected results for these objects.

A boolean operation never affects the original operands, the result is always a new Blender object.

### Chapter 5. ObjectMode

Boolean operations are invoked by selecting *exactly* two meshes and pressing **WKEY**. There are three types of boolean operations to choose from in the popup menu, Intersect, Union and Difference.

The boolean operations also take materials and UV-Textures into account, producing objects with material indices or multi UV-mapped objects.



Figure 5-7. Options for boolean operations

Consider the object Figure 5-7.

- The Intersect operation creates a new object whose surface encloses the volume common to *both* original objects.
- The Union operation creates a new object whose surface encloses the volume of *both* original objects.
- The Difference operation is the only one in which the order of selection is important. The active object (light purple in wireframe view) is subtracted from the selected object. That is, the resulting object surface encloses a volume which is the volume belonging to the selected *and inactive* object, but *not* to the selected *and active* one.

Figure 5-8 shows the results of the three operations.



Figure 5-8. Resulting objects for: intersect, union, difference (top to bottom).

The number of polygons generated can be very large compared to the original meshes, especially when using complex concave objects. Furthermore, the polygons that are generated can be of generally poor quality: very long and thin and sometimes very small. Try using the Mesh Decimator (EditButtons **F9**) to fix this problem.

Vertices in the resulting mesh that fall on the boundary of the two original objects often do not match up, and boundary vertices are duplicated. This is good in some respects because it means that you can select parts of the original meshes by selecting one vertex in the result and pressing the select linked button (**LKEY**). This is handy if you want to assign materials and such to the result.

Note: Sometimes the boolean operation can fail with a message saying ("An internal error occurred -- sorry"). If this occurs, try to move or rotate the objects just a very small

## *Chapter 5. ObjectMode*

amount.

# **Chapter 6. Basic Mesh Modelling**

The principal Object of a 3D scene is usually a *Mesh*. In this chapter we will first enumerate the basic mesh objects, or *primitives*, then follow with the description of the most basic actions you can take on a Mesh Objects.

## **Basic Objects**

### Relevant to Blender v2.31

To create a basic Object press **SPACE** and select "ADD>>Mesh", or, access the 'Add'menu by pressing **SHIFT-A** or simply hold **LMB** on 3D Window, for more than half a second. Select the basic object you'd like to create from the menu. We describe every basic object or *primitive* you can create within Blender below. Figure 6-1 also shows the variety of basic objects that can be created.



Figure 6-1. Basic Objects

## Plane

A standard plane contains four vertices, four edges, and one face. It is like a piece of paper lying on a table; it is not a real three-dimensional object because it is flat and has no thickness. Objects that can be created with planes include floors, tabletops, or mirrors.

## Cube

A standard cube contains eight vertices, 12 edges, and six faces, and is a real threedimensional object. Objects that can be created out of cubes include dice, boxes, or crates.

## Circle

A standard circle is comprised of *n* vertices. The number of vertices can be specified in the popup window which appears when the circle is created. The more vertices the circle contains, the smoother its contour will be. Examples of circle objects are disks, plates, or any kind of flat and round object.

## **UVSphere**

A standard UVsphere is made out of n segments and m rings. The level of detail can be specified in the popup window which appears when the UVsphere is created. Increasing the number of segments and rings makes the surface of the UVsphere smoother. Segments are like Earth meridians, going pole to pole, rings are like Earth parallels. Example objects that can be created out of UVspheres are balls, heads or pearls for a necklace.

**Note:** If you specify a six segment, six ring UVsphere you'll get something which, in top view, is a hexagon (six segments), with five rings plus two points at the poles. Thus, one ring fewer than expected, or two more, if you count the poles as rings of radius 0.

#### **Icosphere**

An Icosphere is made up of triangles. The number of subdivisions can be specified in the window that pops up when the Icosphere is created; increasing the number of subdivisions makes the surface of the Icosphere smoother. At level 1 the Icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Any increasing level of subdivision splits each triangular face into four triangles, resulting in a more spherical appearance. Icosphere's are normally used to achieve a more isotropical and economical layout of vertices than a UVsphere.

#### Cylinder

A standard cylinder is made out of *n* vertices. The number of vertices in the circular cross-section can be specified in the popup window that appears when the object is created; the higher the number of vertices, the smoother the circular cross-section becomes. Objects that can be created out of cylinders include handles or rods.

#### Tube

A standard tube is made out of *n* vertices. The number of vertices in the hollow circular cross-section can be specified in the popup window that appears when the object is created; the higher the number of vertices, the smoother the hollow circular cross-section becomes. Objects that can be created out of tubes include pipes or drinking glasses. (The basic difference between a cylinder and a tube is that the former has closed ends.)

#### Cone

A standard cone is made out of *n* vertices. The number of vertices in the circular base can be specified in the popup window that appears when the object is created; the higher the number of vertices, the smoother the circular base becomes. Objects that can be created out of cones include spikes or pointed hats.

#### Grid

A standard grid is made out of n by m vertices. The resolution of the x-axis and yaxis can be specified in the popup window which appears when the object is created; the higher the resolution, the more vertices are created. Example objects that can be created out of grids include landscapes (with the proportional editing tool) and other organic surfaces.

#### Monkey

This is a gift from old NaN to the community and is seen as a programmer's joke or "Easter Egg". It creates a monkey's head once you press the Oooh Oooh Oooh button. The Monkey's name is *Suzanne* and is Blender's mascot.

## EditMode

#### Relevant to Blender v2.31

When working with geometric objects in Blender, you can work in two modes: ObjectMode and EditMode. Basically, as seen in the previous section, operations in ObjectMode affect whole objects, and operations in EditMode affect only the geometry of an object, but not its global properties such as the location or rotation.

In Blender you switch between these two modes with the **TAB** key. EditMode only works on one object at a time: the active object. An object outside EditMode is drawn in purple in the 3D Windows (in wireframe mode) when selected; it is black otherwise. The active object is drawn black in EditMode, but each vertex is highlighted in purple (Figure 6-2). Selected vertices are drawn in yellow (Figure 6-3) and, if ap-

propriate buttons in the Editing (F9) Context Mesh Tools 1 Panel are pressed (Draw Faces and Draw Edges) also selected edges and faces are highlighted.



Figure 6-2. Two pyramids, one in EditMode (left) and one in ObjectMode (right).



Figure 6-3. Cube with selected vertices in yellow.

## **Structures: Vertices, Edges and Faces**

In basic meshes, everything is built from three basic structures: *Vertices, Edges* and *Faces*. (We're not talking about Curves, NURBS, and so forth here.) But there is no need to be disappointed: This simplicity still provides us with a wealth of possibilities that will be the foundation for all our models.

#### Vertices

A vertex is primarily a single point or position in 3D space. It is usually invisible in rendering and in ObjectMode. (Don't mistake the center point of an object for a vertex. It looks similar, but its bigger and you can't select it.)

To create a new vertex, change to EditMode, hold down **CTRL**, and click with the **LMB**. Of course, as a computer screen is two-dimensional, Blender can't determine all three vertex coordinates from one mouse click, so the new vertex is placed at

the depth of the 3D cursor 'into' the screen. Any vertices selected previously are automatically connected to the new one with an edge.

### Edges

An edge always connects two vertices with a straight line. The edges are the 'wires' you see when you look at a mesh in wireframe view. They are usually invisible on the rendered image. They are used to construct faces. Create an edge by selecting two vertices and pressing **FKEY**.

### Faces

A Face is the most high level structure in a mesh. Faces are used to build the actual surface of the object. They are what you see when you render the mesh. A Face is defined as the area between either three or four vertices, with an Edge on every side. Triangles always work well, because they are always flat and easy to calculate.

Take care when using four-sided faces, because internally they are simply divided into two triangles each. Four-sided faces only work well if the Face is pretty much flat (all points lie within one imaginary plane) and convex (the angle at no corner is greater than or equal to 180 degrees). This is the case with the faces of a cube, for example. (That's why you can't see any diagonals in its wireframe model, because they would divide each square face into two triangles. While you could build a cube with triangular faces, it would just look more confusing in EditMode.)

An area between three or four vertices, outlined by Edges, doesn't have to be a face. If this area does not contain a face, it will simply be transparent or non-existent in the rendered image. To create a face, select three or four suitable vertices and press **FKEY**.

## **Basic Editing**

Most simple operations from ObjectMode (like selecting, moving, rotating, and scaling) work identically on vertices as they do on objects. Thus, you can learn how to handle basic EditMode operations very quickly. The only notable difference is a new scaling option, **ALT-S** which scales the selected vertices along the direction of the normals (shrinks-fattens). The truncated pyramid in Figure 6-4, for example, was created with the following steps:

- 1. Add a cube to an empty scene. Enter EditMode.
- 2. Make sure all vertices are deselected (purple). Use border select (**BKEY**) to select the upper four vertices.
- 3. Check that the scaling center is set to *anything but* the 3D cursor (see Figure 5-1), then switch to scale mode (**SKEY**), reduce the size, and confirm with **LMB**.
- 4. Exit EditMode by pressing TAB.



Figure 6-4. Chopped-off pyramid

One Extra feature for Edit Mode is the Mirroring tool. If you have some vertices selected and you press **MKEY** you will be presented with a Menu containing nine options. You can select from these to mirror the selected vertice with respect to any of the X,Y or Z axis of the Global, Local, or Viewing reference.

One additional feature of EditMode is the CircleSelect mode. It is invoked by pressing **BKEY** twice instead of only once, as you would for BorderSelect. A light grey circle is drawn around the cursor and any **LMB** click selects all vertices within. **NUM+** and **NUM-** or the **MW**, if any, enlarge or shrink the circle.

All operations in EditMode are ultimately performed on the vertices; the connected edges and faces automatically adapt, as they depend on the vertices' positions. To select an edge, you must select the two endpoints or either place the mouse on the edge and press **CTRL-ALT-RMB**. To select a face, each corner must be selected.

EditMode operations are many, and most are summarized in the Editing Context Buttons window, accessed via the ( 🛄 ) header button or via **F9** (Figure 6-5). Note the group of buttons in the Mesh Tools 1 Panel:

<ul> <li>Link and Materials</li> </ul>		Mesh Tools	Mesh Tools 1
ME:Cube.004 F 0B:Cube.003 Vertex Groups     0 Met: 0 -> ?     New Delete     Select Desel.     AutorexSpace     Set Smoot Set Sold	Auto Smooth Begr:S0 SubSurf SubSurf Centre Dptmal Sticky Make Slower Da Centre New Centre New Centre New Centre New Centre New Centre Centre Centre New Centre C	Beauty Usedivide Fract Sud Noise Heah Xort To Sphere Smooth Spit The Normals Tem Double Limit. 0001 Extrude Screw Spin Spin Dup Degr: 90 Steps 9 Turns 1 Keep Original Colovate Extrude Dup Offset 1.000	Centre Hide Reveal Select Swap Nize: 0.100 Draw Normals Draw Pices Draw Edges All edges

Figure 6-5. Edit Context.

- NSize: Determines the length, in Blender Units, of the normals to the faces, if they are drawn.
- Draw Normals Toggle drawing of Normals. If this is ON, face normals are drawn as cyan segments.
- Draw Faces If this is ON, faces are drawn as semi-transparent blue, or as semi-transparent purple if they are selected. If this is OFF, faces are invisible.
- Draw Edges Edges are always drawn black, but if this button is ON, selected edges are drawn in yellow. Edges joining a selected node and an un-selected one have a yellow-black gradient.
- All Edges Only those edges strictly necessary to show an object's shape are shown in Object mode. You can force Blender to draw all edges with this button.

Note: Of course, all these colors are customizable in the Theme editor.

With **WKEY** you can call up the "Specials" menu in EditMode (Figure 6-6). With this menu you can quickly access functions which are frequently required for polygon-modelling.

**Tip:** You can access the entries in a PopupMenu by using the corresponding numberkey. For example, pressing **WKEY** and then **1KEY** will subdivide the selected edges without you having to touch the mouse at all.

pecials		
ubdivide		
ubdivide Fractal		
ubdivide Smooth		
lerge		
emove Doubles		
ide		
eveal		
elect swap		
Flip Normals		
mooth		

Figure 6-6. Specials Menu

- Subdivide Each selected edge is split in two, new vertices are created at middle points, and faces are split too, if necessary.
- Subdivide Fractal As above, but new vertices are randomly displaced within a user-defined range.
- Subdivide Smooth As above, but new vertices are displaced towards the barycenter (centre of mass) of the connected vertices.
- Merge Merges selected vertices into a single one, at the barycenter position or at the cursor position.
- Remove Doubles Merges all of the selected vertices whose relative distance is below a given threshold (0.001 by default).
- Hide Hides selected vertices.
- Reveal Shows hidden vertices.
- Select Swap All selected vertices become unselected and vice-versa.
- Flip Normals Change the Normal directions in the selected faces.
- Smooth Smooths out a mesh by moving each vertex towards the barycenter of the linked vertices.
- Mirror Same as MKEY described above.

Many of these actions have a button of their own in the Mesh Tools Panel of the Edit Buttons Window (Figure 6-5). The Remove doubles threshold can be adjusted here, too.

## Mesh Undo

As of Blender 2.3 we finally have a true Undo. It works only for Meshes and only in EditMode.

Mesh undo works in the background saving copies of your mesh in memory as you make changes. Pressing the **UKEY** in mesh EditMode reverts to the previously saved mesh, undoing the last edit operation (Figure 6-7).

Undo operations are only stored for one mesh at a time. You can leave and re-enter EditMode for the same mesh without losing any undo information, but once another mesh is edited, the undo information for the first is lost.



Figure 6-7. Undo and Redo

Pressing **SHIFT-U** re-does the last undo operation (Figure 6-7). Pressing **ALT-U** brings up the Undo Menu (Figure 6-8). This lists all the undo steps by name so you can quickly find your way back to a known good point in your work. The **ALT-U** menu also contains the option All Changes. This option is more powerful than merely pressing **UKEY** repeatedly, and will reload the mesh data as it was at the beginning of your edit session, even if you have used up all your undo steps.



Figure 6-8. Undo Menu

Edit undo can be memory intensive. A mesh of 64,000 faces and verts can use over 3Mb of RAM per undo step. If you are on a machine that is strapped for RAM, in the User Preference Window, under Edit Methods, there is a NumButton for determining the maximum number of undo steps saved. The allowable range is between 1 and 64. The default is 32.

## Smoothing

*Relevant to Blender v2.31* 

As seen in the previous sections, polygons are central to Blender. Most objects in Blender are represented by polygons and truly curved objects are often approximated by polygon meshes.

When rendering images, you may notice that these polygons appear as a series of small, flat faces. (Figure 6-9). Sometimes this is a desirable effect, but usually we want our objects to look nice and smooth. This section shows you how to smooth an object, and how to apply the AutoSmooth filter to quickly and easily combine smooth and faceted polygons in the same object.



Figure 6-9. Simple un-smoothed test object

There are two ways to activate Blender's face smoothing features. The easiest way is to set an entire object as smooth or faceted by selecting a mesh object, in Object-Mode, switching to the Editing Context (F9), and clicking the Set Smooth button in the Link and Materials Panel (Figure 6-10). The button does not stay pressed, but forces Blender to assign the "smoothing" attribute to each face in the mesh. Now, rendering the image with F12 should produce the image shown in Figure 6-11. Notice that the outline of the object is still strongly faceted. Activating the smoothing features doesn't actually modify the object's geometry; it changes the way the shading is calculated across the surfaces, giving the illusion of a smooth surface.

Click the Set Solid button in the same Panel to revert the shading to that shown in Figure 6-9.

🔻 Link an	d Materials			
ME:Cube.004     F     OB:Cube.003				
Vertex Groups				
< 0 Mat: 0 ▶ ?				
New	Delete	New	Delete	
Assign	Remove	Select	Deselect	
Select	Desel.	Assign		
AutoTe	xSpace	Set Smoot	Set Solid	

Figure 6-10. Set Smooth and Set Solid buttons of EditButtons window



Figure 6-11. Same object as above, but completely smoothed by 'Set Smooth'

Alternatively, you can choose which faces to smooth by entering EditMode for the object with **TAB**, then selecting the faces and clicking the Set Smooth button (Figure 6-12). When the mesh is in editmode, only the selected faces will receive the "smoothing" attribute. You can set solid faces (removing the "smoothing" attribute) in the same way: by selecting faces and clicking the Set Solid button.



Figure 6-12. Object in editmode with some faces selected.

It can be difficult to create certain combinations of smooth and solid faces using the above techniques alone. Though there are workarounds (such as splitting off sets of faces by selecting them and pressing **YKEY**), there is an easier way to combine smooth and solid faces, by using AutoSmooth.

Press the AutoSmooth button in the Mesh Panel of the Edit Buttons (Figure 6-13) to tell Blender to decide which faces should be smoothed on the basys of the angle between faces (Figure 6-14). Angles on the model that are sharper than the angle specified in the Degr NumBut will not be smoothed. Higher values will produce more smoothed faces, while the lowest setting will look identical to a mesh that has been set completely solid.

Only faces that have been set as smooth will be affected by the AutoSmooth feature. A mesh, or any faces that have been set as solid will not change their shading when AutoSmooth is activated. This allows you extra control over which faces will be smoothed and which ones won't by overriding the decisions made by the AutoSmooth algorithm.

🔽 Mesh		
Auto Smooth Degr: 30		
SubSurf	TexMesh:	
Subdiv: 1   <1		Centre
Uptimaj		Centre New
Sticky Make		Centre Cursor
VertCol Make	SlowerDra	Double Sided
TexFac Make	FasterDra	No V.Normal Fli

Figure 6-13. AutoSmooth button group in the EditButtons window.



Figure 6-14. Same test object with AutoSmooth enabled

## Extrude

### Relevant to Blender v2.31

One tool of paramount importance for working with Meshes is the "Extrude" command (**EKEY**). This command allows you to create cubes from rectangles and cylinders from circles, as well as to very easily create such things as tree limbs. Although the process is quite intuitive, the principles behind Extrude are fairly elaborate as discussed below.

- First, the algorithm determines the outside edge-loop of the Extrude; that is, which among the selected edges will be changed into faces. By default, the algorithm considers edges belonging to two or more selected faces as internal, and hence not part of the loop.
- The edges in the edge-loop are then changed into faces.
- If the edges in the edge-loop belong to only one face in the complete mesh, then all of the selected faces are duplicated and linked to the newly created faces. For example, rectangles will result in cubes during this stage.
- In other cases, the selected faces are linked to the newly created faces but not duplicated. This prevents undesired faces from being retained 'inside' the resulting mesh. This distinction is extremely important since it ensures the construction of consistently coherent, closed volumes at all times when using Extrude.
- Edges not belonging to selected faces, which form an 'open' edge-loop, are duplicated and a new face is created between the new edge and the original one.
- Single selected vertices which do not belong to selected edges are duplicated and a new edge is created between the two.

Grab mode is automatically started when the Extrude algorithm terminates, so newly created faces, edges, and vertices can be moved around with the mouse.

Extrude is one of the most frequently used modelling tools in Blender. It's simple, straightforward, and easy to use, yet very powerful. The following short lesson describes how to build a sword using Extrude.

## The Blade

1. Start Blender and delete the default plane. In top view add a mesh circle with eight vertices. Move the vertices so they match the configuration shown in Figure 6-15.



Figure 6-15. Deformed circle, to become the blade cross section.

2. Select all the vertices and scale them down with the **SKEY** so the shape fits in two grid units. Switch to front view with **NUM1**.

3. The shape we've created is the base of the blade. Using Extrude we'll create the blade in a few simple steps. With all vertices selected press **EKEY**, or click the Extrude button in the Mesh Tools Panel of the Editing Context (**F9** - Figure 6-16). A box will pop up asking Ok? Extrude (Figure 6-17).

Click this text or press **ENTER** to confirm, otherwise move the mouse outside or press **ESC** to exit. If you now move the mouse you'll see that Blender has duplicated the vertices, connected them to the original ones with edges and faces, and has entered grab mode.

<ul> <li>Mesh Tools</li> </ul>			
Beauty	Subdivide	Fract Subd	
Noise	Hash	Xsort	
To Sphere	Smooth	Split	
Flip Normals	Rem Double	Limit: 0.001	
	Extrude		
Screw	Screw Spin Spin Dup		
I Degr: 90 ⊦	∢ Steps: 9 »	< Turns: 1 🕨	
Кеер С	Keep Original		
Extrude Dup Offset: 1.000			

Figure 6-16. Extrude button in EditButtons context.



Figure 6-17. Extrude confirmation box.

4. Move the new vertices up 30 units, constraining the movement with **CTRL**, then click **LMB** to confirm their new position and scale them down a little bit with the **SKEY** (Figure 6-18).



Figure 6-18. The Blade

5. Press **EKEY** again to extrude the tip of the blade, then move the vertices five units up. To make the blade end in one vertex, scale the top vertices down to 0.000 (hold **CTRL** for this) and press **WKEY**>Remove Doubles (Figure 6-19) or click the Rem Doubles button in the EditButtons (**F9**). Blender will inform you that it has removed seven of the eight vertices and only one vertex remains. The blade is complete! (Figure 6-20)

Chapter 6. Basic Mesh Modelling



Figure 6-19. Mesh Edit Menu



Figure 6-20. The completed blade

## The Handle

6. Leave edit mode and move the blade to the side. Add a UVsphere with 16 segments and rings and deselect all the vertices with the **AKEY**.

7. Borderselect the top three rings of vertices with **BKEY** and delete them with **XKEY**>>Vertices (Figure 6-21).



Figure 6-21. UV sphere for the handle: vertices to be removed



Figure 6-22. First extrusion for the handle

8. Select the top ring of vertices and extrude them. Move the ring up four units and scale them up a bit (Figure 6-22), then extrude and move four units again twice and scale the last ring down a bit (Figure 6-23).

9. Leave EditMode and scale the entire handle down so that it's in proportion with the blade. Place it just under the blade.

### Chapter 6. Basic Mesh Modelling



Figure 6-23. Complete handle

## The Hilt

By now you should be used to the 'extrude>move>scale' sequence, so try to model a nice hilt with it. Start out with a cube and extrude different sides a few times, scaling them where needed. You should be able to get something like that shown in Figure 6-24.



Figure 6-24. Complete Hilt

After texturing, the sword looks like Figure 6-25



Figure 6-25. Finished sword, with textures and materials

As you can see, extrude is a very powerful tool that allows you to model relatively complex objects very quickly (the entire sword was created in less than one half hour). Getting the hang of extrude>move>scale will make your life as a Blender modeler a lot easier.

## Spin and SpinDup

### Relevant to Blender v2.31

Spin and spin dup are two other very powerful modelling tools allowing you to easily create bodies of revolution or axially periodic structures.

## Spin

Use the Spin tool to create the sort of objects that you would produce on a lathe. (This tool is often called a "lathe"-tool or a "sweep"-tool in the literature, for this reason.)

First, create a mesh representing the profile of your object. If you are modeling a hollow object, it is a good idea to thicken the outline. Figure 6-26 shows the profile for a wine glass we will model as a demonstration.



Figure 6-26. Glass profile

In EditMode, with all the vertices selected, access the Editing Context (F9). The Degr button in the Mesh Tools Panel indicates the number of degrees to spin the object (in this case we want a full 360° sweep). The Steps button specifies how many profiles there will be in the sweep (Figure 6-27).

<ul> <li>Mesh Tools</li> </ul>			
Beauty	Subdivide	Fract Subd	
Noise	Hash	Xsort	
To Sphere	Smooth	Split	
Flip Normals	Rem Double	Limit: 0.001	
	Extrude		
Screw	Screw Spin Spin Dup		
OCIEW	opin	opin Dup	
■ Degr: 90	<ul> <li>Steps: 9</li> </ul>	Turns: 1	
Degr: 90 Keep 0	Steps: 9 Driginal	Turns: 1 Clockwise	

**Figure 6-27. Spin Buttons** 

Like Spin Duplicate (discussed in the next section), the effects of Spin depend on the placement of the cursor and which window (view) is active. We will be rotating the object around the cursor in the top view. Switch to the top view with **NUM7**.

1. Place the cursor along the center of the profile by selecting one of the vertices along the center, and snapping the cursor to that location with **SHIFT-S**>>Curs->Sel.

Figure 6-28 shows the wine glass profile from top view, with the cursor correctly positioned.



Figure 6-28. Glass profile, top view in edit mode, just before spinning.

Before continuing, note the number of vertices in the profile. You'll find this information in the Info bar at the top of the Blender interface (Figure 6-29).

ò www.blender.org 231 Ve:38-38 | Fa:0-0 | Mem:1.301

### Figure 6-29. Mesh data - Vertex and face numbers.

2. Click the "Spin" button. If you have more than one window open, the cursor will change to an arrow with a question mark and you will have to click in the window containing the top view before continuing. If you have only one window open, the spin will happen immediately.

Figure 6-30 shows the result of a successful spin.



Figure 6-30. Spinned profile

3. The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select (**BKEY**) (Figure 6-31) and do a Remove Doubles operation.



Figure 6-31. Seam vertex selection

Notice the selected vertex count before and after the Remove Doubles operation (Figure 6-32). If all goes well, the final vertex count (38 in this example) should match the number of the original profile noted in Figure 6-29. If not, some vertices were missed and you will need to weld them manually. Or, worse, too many vertices will have been merged.

⇒ SCE:1	X	💩 www.blender.org 231	Ve:146-1406   Fa:0-1332   Me
\$ SCE:1	×	💩 www.blender.org 231	Ve:38-1298   Fa:0-1332   Mer

Figure 6-32. Vertex count after removing doubles.

**Merging two vertices in one:** To merge (weld) two vertices together, select both of them by holding **SHIFT** and **RMB** on them. Press **SKEY** to start scaling and hold down **CON-TROL** while scaling to scale the points down to 0 units in the X,Y and Z axis. **LMB** to complete the scaling operation and click the Remove Doubles button in the EditButtons window.

Alternatively, you can press **WKEY** and select Merge from the appearing Menu (Figure 6-33). Then, in a new menu, choose whether the merged node will have to be at the center of the selected nodes or at the cursor. The first choice is better in our case.


Figure 6-33. Merge menu

All that remains now is to recalculate the normals by selecting all vertices and pressing **CTRL-N**>>Recalc Normals Outside. At this point you can leave EditMode and apply materials or smoothing, set up some lights, a camera and make a rendering. Figure 6-34 shows our wine glass in a finished state.



Figure 6-34. Final render of the glasses.

### SpinDup

The Spin Dup tool is a great way to quickly make a series of copies of an object along a circle. For example, if you have modeled a clock, and you now want to add hour marks.



Figure 6-35. Hour mark indicated by the arrow

Model just one mark, in the 12 o'clock position (Figure 6-35). Select the mark and switch to the Editing Context with **F9**. Set the number of degrees in the Degr: Num Button in the Mesh Tools Panel to 360. We want to make 12 copies of our object, so set the Steps to 12 (Figure 6-36).

<ul> <li>Mesh Tools</li> </ul>			
Beauty	Subdivide	Fract Subd	
Noise	Hash	Xsort	
To Sphere	Smooth	Split	
Flip Normals	Rem Double	Limit: 0.001	
	Extrude		
Screw	Screw Spin		
I Degr: 90⊩	∢ Steps: 9≽	🔹 Turns: 1 🕨	
	Keep Original		
Keep C	Driginal	Clockwise	

Figure 6-36. Spin Dup buttons

- Switch the view to the one in which you wish to rotate the object by using the keypad. Note that the result of the Spin Dup command depends on the view you are using when you press the button.
- Position the cursor at the center of rotation. The objects will be rotated around this point.
- Select the object you wish to duplicate and enter EditMode with TAB.
- In EditMode, select the vertices you want to duplicate (note that you can select all vertices with **AKEY** or all of the vertices linked to the point under the cursor with **LKEY**). See Figure 6-37.

**Cursor Placement:** To place the cursor at the precise location of an existing object or vertex, select the object or vertex, and press **SHIFT-S**>>CURS>>SEL.



Figure 6-37. Mesh selected and ready to be SpinDuped

• Press the Spin Dup button. If you have more than one 3DWindow open, you will notice the mouse cursor change to an arrow with a question mark. Click in the window in which you want to do your rotation. In this case, we want to use the front window (Figure 6-38).

If the view you want is not visible, you can dismiss the arrow/question mark with **ESC** until you can switch a window to the appropriate view with the keypad.



Figure 6-38. View selection for Spin Dup.

When spin-duplicating an object 360 degrees, a duplicate object is placed at the same location of the first object, producing duplicate geometry. You will notice that after clicking the Spin Dup button, the original geometry remains selected. To delete it,

simply press **XKEY**>>Vertices. The source object is deleted, but the duplicated version beneath it remains (Figure 6-39).



Figure 6-39. Removal of duplicated object

**Avoiding duplicates:** If you like a little math you needn't bother with duplicates because you can avoid them at the start. Just make 11 duplicates, not 12, and not around the whole  $360^{\circ}$ , but just through  $330^{\circ}$  (that is  $360^{*}11/12$ ). This way no duplicate is placed over the original object.

In general, to make *n* duplicates over 360 degrees without overlapping, just spin one less object over  $360^{*}(n-1)/n$  degrees.

Figure 6-40 shows the final rendering of the clock.



Figure 6-40. Final Clock Render.

## Screw

#### Relevant to Blender v2.31

The "Screw" tool combines a repetitive "Spin" with a translation, to generate a screwlike, or spiral-shaped, object. Use this tool to create screws, springs, or shell-shaped structures.



Figure 6-41. How to make a spring: before (left) and after (right) the Screw tool.

The method for using the "Screw" function is strict:

- Set the 3DWindow to front view (NUM1).
- Place the 3DCursor at the position through which the rotation axis must pass. Such an axis will be vertical.
- Ensure that an *open poly line* is available. This can be a single edge, as shown in the figure, or a half circle, or whatever. You need only to ensure that there are two 'free' ends; two vertices belonging to a single edge linking then to another vertex. The "Screw" function localizes these two points and uses them to calculate the translation vector that is added to the "Spin" per each full rotation (Figure 6-41). If these two vertices are at the same location, this creates a normal "Spin". Otherwise, interesting things happen!
- Select all vertices that will participate in the "Screw".
- Assign the Num Buttons Steps: and Turns: in the Mesh Tools Panel the desired values. Steps: determines how many times the profile is repeated within each 360° rotation, while Turns: sets the number of complete 360° rotations to be performed.
- Press Screw!

If there are multiple 3DWindows, the mouse cursor changes to a question mark. Click on the 3DWindow in which the "Screw" is to be executed.

If the two "free" ends are aligned vertically the result is the one seen above. If they are not, the translation vector remains vertical, equal to the vertical component of the vector joining the two 'free' vertices, while the horizontal component generates an enlargement (or reduction) of the screw as shown in Figure 6-42.



Figure 6-42. Enlarging screw (right) obtained with the profile on the left.

## Warp Tool

#### Relevant to Blender v2.31

The warp tool is a little-known tool in Blender, partly because it is not found in the EditButtons window, and partly because it is only useful in very specific cases. At any rate, it is not something that the average Blender-user needs to use every day.

A piece of text wrapped into a ring shape is useful when creating flying logos, but it would be difficult to model without the use of the warp tool. For our example, we'll warp the phrase "Amazingly Warped Text" around a sphere.

1. First add the sphere.

2. Then add the text in front view, in the Editing Context and Curve and Surface Panel set Ext1 to 0.1 - making the text 3D, and set Ext2 to 0.01, adding a nice bevel to the edge. Make the BevResol 1 or 2 to have a smooth bevel and lower the resolution so that the vertex count will not be too high when you subdivide the object later on (Figure 6-43 - and see the Section called *Text* in Chapter 9). Convert the object to curves, then to a mesh, (**ALT-C** twice) because the warp tool does not work on text or on curves. Subdivide the mesh twice, so that the geometry will change shape cleanly, without artifacts.

🔻 Curve and Surface		
UV Orco	✓ DefResolU: 4 → Set	 suiltin>
Centre Centre New Centre Cursor	Back         Front         3D           Width:         1.000	Load Font         ToUpper           Loft         Right         Middle         Plush           TextOnCurve:         Ob Family:              Ob Family:          Size: 1.000 ·          Linedist: 1.000 ·            Spacing: 1.000 ·         Y offset: 0.00 ·          X offset: 0.00 ·

Figure 6-43. Text settings

Switch to top view and move the mesh away from the 3D cursor. This distance defines the radius of the warp. (See Figure 6-44.)



Figure 6-44. Top view of text and sphere

Place the mesh in Edit Mode (**TAB**) and press **AKEY** to select all vertices. Press **SHIFT-W** to activate the warp tool. Move the mouse up or down to interactively define the amount of warp. (Figure 6-45). Holding down **CTRL** makes warp change in steps of five degrees.



Figure 6-45. Warped text

Now you can switch to camera view, add materials, lights and render (Figure 6-46).



Figure 6-46. Final rendering

## **Object Hooks**

by Kenneth Styrberg

Relevant to Blender v2.35

Hooks give access at object level to the underlying geometry of meshes, curves, surfaces or lattices. A hook is an object feature and it is like a parent to an object, but for vertices. You can create as many hooks to an object as you like, and assign for each hook vertices that will be affected. Overlapping hooks is also possible, here a weighting factor per hook is provided that determines the amount each hook will affect the overlapping vertices. Note: When you completely remodel something, you most likely have to reassign existing hooks as well.

### Adding hooks

Since hooks relate to vertices or control points, most editing options are available in edit mode for meshes, curves, surfaces and lattices. Select any number of vertices, and press **CTRL-H** to access the hooks menu.



Figure 6-47. Hooks menu

Add, New Empty Adds a new hook and create a new empty object, that will be a parent to the selection, at the center of the selection.

Add, To Selected Object When another object is selected (you can do that in edit mode with CTRL-RMB) the new hook is created and parented to that object.

### Using hooks

Inside of edit mode, hooks are disabled, to enable modeling better. Only in object mode you can actually use the hooks. All object level options and transformations are possible now, including using hierarchies, constraints, ipo and path animations.

You can also make the hook-parent a child of the original object if you don't want object transformations to deform the hooks.

### EditMode options

Once hooks are available in an object, the CTRL-H menu will give additional options:

Hooks
Add Hook, To New Empty
Add Hook, To Selected Object
Remove
Reassign
Select
Clear Offset

Figure 6-48. Hooks extended menu

Remove... This will give a new menu with a list of hooks to remove.

Reassign... Use this if you want to assign new vertices to a hook.

**Select...** To select the vertices of a specific hook.

Clear Offset... This will neutralize the current transformation of a hook parent.

#### **Hooks** panel

You can find buttons for hooks in the object context (F7) in the Hooks tab. Here you can give a hook a new name, the default name is the parent name, give it a new parent by typing the new parents name or assign it a Force weighting factor.

▼ Draw	Hooks
Empty	Name: Empty
	Eviloff: 0.00
Parent:Empty	Force: 1.0
Delete	Clear offset

Figure 6-49. Hooks panel

**Force** Since multiple hooks can work on the same vertices, you can weight the influence of a hook this way. Weighting rules are:

- If the total of all forces is smaller than 1.0, the remainder, 1.0-forces, will be the factor the original position have as force.
- If the total of all 'forces' is larger than 1.0, it only uses the hook transformations, averaged by their weights.

**Falloff** If not zero, the falloff is the distance where the influence of a hook goes to zero. It currently uses a smooth interpolation, comparable to the Proportional Editing Tools. (See the Section called *Proportional Editing Tool* in Chapter 7)

**Delete** Delete the hook from the object.

Clear offset Neutralize the current transformation of a hook.

## Chapter 7. Advanced Mesh Modelling

Blender provides several advanced Mesh Modeling features, mostly aimed at handling easily complex meshes or rather enabling economical, lov-vertex number modeling of complex smooth surfaces.

## **Catmull-Clark Subdivision Surfaces (-)**

*Relevant to Blender v2.31 MISSING SIMPLE SUBSURF and CREASES (so let's wait 2.234 to update)* 

With any regular Mesh as a starting point, Blender can calculate a smooth subdivision on the fly, while modelling or while rendering, using Catmull-Clark Subdivision Surfaces or, in short *SubSurf*. SubSurf is a mathematical algorithm to compute a "smooth" subdivision of a mesh. This allows high resolution Mesh modelling without the need to save and maintain huge amounts of data. This also allows for a smooth 'organic' look to the models.

Actually a SubSurfed Mesh and a NURBS surface have many points in common inasmuch as both rely on a "coarse" low-poly "mesh" to define a smooth "high definition" surface. But there are also notable differences:

- NURBS allow for finer control on the surface, since you can set "weights" independently on each control point of the control mesh. On a SubSurfed mesh you cannot act on weights.
- SubSurfs have a more flexible modelling approach. Since a SubSurf is a mathematical operation occurring on a mesh, you can use all the modelling techniques described in this chapter on the mesh. There are more techniques, which are far more flexible, than those available for NURBS control polygons.

SubSurf is a Mesh option, activated in the Editing Context Mesh Panel (**F9** - Figure 7-1). The Num Buttons immediately below it define, on the left, the resolution (or level) of subdivision for 3D visualization purposes; the one on the right, the resolution for rendering purposes. You can also use **SHIFT-O** if you are in ObjectMode. This switches SubSurf On/Off. The SubSurf level can also be controlled via **CTRL-1** to **CTRL-4**, but this only affects the visualization sub-division level.

Since SubSurf computations are performed both real-time, while you model, and at render time, and they are CPU intensive, it is usually good practice to keep the Sub-Surf level low (but non-zero) while modelling; higher while rendering.

▼ Mesh		
Auto Smooth	d Decir	mator: 2 🔹 🕨
✓ Degr: 30 →	Apply	Cancel
SubSurf	TexMesh:	Centre
	ļ	Centre New
Sticky Make	l	Centre Cursor
VertCol Make	SlowerDra	Double Sided
TexFace Make	FasterDraw	No V.Normal Fli

**Figure 7-1. SubSurf buttons** 

From version 2.3 Blender has a new SubSurfed-related button: Optimal. This changes the way SubSurf meshes are drawn and can be of great help in modelling. Figure 7-

### Chapter 7. Advanced Mesh Modelling

2 shows a series of pictures showing various different combinations on Suzanne's Mesh.



Figure 7-2. SubSurfed Suzanne.

Figure 7-3 shows a 0,1,2,3 level of SubSurf on a single square face or on a single triangular face. Such a subdivision is performed, on a generic mesh, for *each* square or rectangular face.

It is evident how each single quadrilateral face produces  $4^n$  faces in the SubSurfed mesh. *n* is the SubSurf level, or resolution, while each triangular face produces  $3^*4^{(n-1)}$  new faces (Figure 7-3). This dramatic increase of face (and vertex) number results in a slow-down of all editing, and rendering, actions and calls for lower SubSurf level in the editing process than in the rendering one.



Figure 7-3. SubSurf of simple square and rectangular faces.

Blender's subdivision system is based on the Catmull-Clarke algorithm. This produces nice smooth SubSurf meshes but any 'SubSurfed' face, that is, any small face created by the algorithm from a single face of the original mesh, shares the normal orientation of that original face.

This is not an issue for the shape itself, as Figure 7-4 shows, but it is an issue in the rendering phase and in solid mode, where abrupt normal changes can produce ugly black lines (Figure 7-5).



Figure 7-4. Side view of subsurfed meshes. With random normals (top) and with coherent normals (bottom)

Use the **CTRL-N** command in EditMode, with all vertices selected, to make Blender recalculate the normals.



Figure 7-5. Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).

In these images the face normals are drawn cyan. You can enable drawing normals in the EditButtons (**F9**) menu.

#### Chapter 7. Advanced Mesh Modelling

Note that Blender cannot recalculate normals correcty if the mesh is not "Manifold". A "Non-Manifold" mesh is a mesh for which an 'out' cannot unequivocally be computed. Basically, from the Blender point of view, it is a mesh where there are edges belonging to *more* than two faces.

Figure 7-6 shows a very simple example of a "Non-Manifold" mesh. In general a "Non-Manifold" mesh occurs when you have internal faces and the like.



Figure 7-6. A "Non-Manifold" mesh

A "Non-Manifold" mesh is not a problem for conventional meshes, but can give rise to ugly artifacts in SubSurfed meshes. Also, it does not allow decimation, so it is better to avoid them as much as possible.

Use these two hints to tell whether a mesh is "Non Manifold":

- The Recalculation of normals leaves black lines somewhere
- The "Decimator" tool in the Mesh Panel refuses to work stating that the mesh is "No Manifold"

The SubSurf tool allows you to create very good "organic" models, but remember that a regular Mesh with square faces, rather than triangular ones, gives the best results.

Figure 7-7 and Figure 7-8 show an example of what can be done with Blender Sub-Surfs.



Figure 7-7. A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).



Figure 7-8. Solid view (left) and final rendering (right) of the Gargoyle.

## Weighted creases for subdivision surfaces

by Kenneth Styrberg

Relevant to Blender v2.34

Weighted creases for subdivision surfaces allows for tuning of the edge sharpness.

Creases are a property of mesh edges, and can be edited in mesh *Edit Mode* when the mesh is a subsurf. Select the edges you want to have creased, and press **SHIFT-E** to change the amount of the edge sharpness.

You can enable an indication of your edge sharpness by enabling *Draw Creases*. See Figure 7-9.

▼ Mesh Tools 1					
Centre					
Hide Reveal					
Select Swap					
NSize: 0.100					
Draw Normals					
Draw Faces					
Draw Edges					
Draw Creases					
Draw Seams					
All Edges					

Figure 7-9. Mesh Tools 1 panel

The sharpness value on the edge is indicated as a thicker part on the edge. If the edge has a sharpness value of 1.0, the edge will have a thicker look, and if sharpness value is 0.0, the edge will be thin. If sharpness value is between 0.0 and 1.0, only part of the edge will be thicker. See Figure 7-10.



Figure 7-10. Edge sharpness around 0.5

To use creases we need to activate subsurfaces. Select the default cube, if you do not have one, add one! Go to the Edit panel (**F9**) and press the *SubSurf* button. Make sure the subsurf type is *Catmull-Clark* in the drop-down list, now move up the subdivision level to 3 for both display and render values. See Figure 7-11. The cube will get the look of a sphere.

🔻 Mesh			
Auto Smooth		✓ Decimator: 12 →	
✓ Degr: 30 →		Apply	Cancel
SubSurf < Subdiv	Catmul ≎ /: 3 ► < 3 ►	TexMesh:	
Op	timal	[	Centre
Edges	Delete		Centre New
VertCol	Make	l	Centre Cursor
TexFac	Make	SlowerDra	Double Sided
Sticky	Make	FasterDra	No V.Normal Fli

Figure 7-11. Mesh panel

Enter *Edit Mode* (**TAB**), with cube selected. By default you are in *Vertices Select* mode, now press **CTRL-TAB** to get a *Select Mode* menu, Figure 7-12, select *Edges*.



Figure 7-12. Select Mode menu

Now select all edges by pressing **AKEY**. All edges should get a yellowish color, Figure 7-13. If all edges are black, then something was already previously selected. Press **AKEY** again to select all edges.



Figure 7-13. All edges selected

Now press **SHIFT-E** to edit the edge sharpness value. The sharpness value will be seen in realtime in the tool bar, Figure 7-14. Move mouse pointer closer to or away from the edge/s to alter the sharpness value. Set the value for all edges to 1.0. The cube will get back it's normal look as a cube.

(1) Cube	
Edge sharpness: 0.376	

Figure 7-14. Edge sharpness

Now select two opposite edges on top of cube. Press **SHIFT-E** to edit the edge sharpness value. Set the value for the edges to 0.0, Figure 7-15.



Figure 7-15. Two edges selected

If you render you will get a nice cube with a rounded top, Figure 7-16. A very good improvement in workflow to achieve this kind of mesh!



Figure 7-16. Result

## **Edge Tools**

Relevant to Blender v2.33

In Blender 2.30 some brand new modelling tools were added. These focused on edges and faces, as opposed to vertices.

A key issue in Modelling is often the necessity to add vertices in certain zones of the mesh, and this is often regarded as splitting, or adding, edges in a given region. Blender now offers two tools for this, a *Knife Tool* able to split edges in desired locations, and a *Face Loop* tool, able to select face paths and split them consistently.

Many Edge Tools are grouped in a menu which is linked to **KKEY** Hotkey, but each individual tool has its own hotkey as well.

### **Edge/Face select**

In EditMode, by pressing **ALT-B** one activates the edge/face select tool. If **ALT-B** is pressed once, then Blender is in edge select mode. The edge under the cursor is highlighted cyan. For each end point in the edge the following operations are performed:

1. It checks to see if it connects to only 3 other edges.

2. If the edge in question has already been added to the list, the selection ends.

3. Of the 3 edges that connect to the current edge, the ones that share a face with the current edge are eliminated and the remaining edge is added to the list and is made the current edge.

This way a loop of edges is highlighted (Figure 7-17). By pressing **LMB** such a highlighted loop is converted into a set of selected vertices. Any previously selected vertices become unselected. If you wish to add the highlighted loop to the current selection use **SHIFT-LMB**, while if you want to subtract the highlighted loop from the current selection use **ALT-LMB**.



Figure 7-17. One open (left) and one closed (right) Edgeloop.

If **ALT-B** is pressed twice a Face Loop, rather than an Edge Loop, is highlighted. A face loop is made by two neighbouring edge loops and extends only to quadrilateral faces, ending when a triangular face is met (and the two bounding edgeloops merge into one). The same mouse actions apply as for the edge loops (Figure 7-18).

Face loop selection is also invoked with SHIFT-R in EditMode.



Figure 7-18. One open (left) and two closed (center and right) Faceloops.

### **Face Loop Splitting**

The Loop tool allows you, eventually, to split, a *loop* of faces. This loop is defined as described in the previous section.

In EditMode press **CTRL-R** rather than **SHIFT-R**. The edge under the cursor is aquamarine, the median line of the corresponding face loop is highlighted yellow (Figure 7-19, left). Once the face loop selection is performed via **LMB** a cyan line is highlighted between the two edgeloops defining the faceloop.

One of the two vertices pertaining to the edge under the mouse pointer defining the edgeloop is highlighted via a big magenta dot (Figure 7-19, center left). Now by moving the mouse the cyan edge loop moves towards or away from the magenta dot. In the 3D Window header the distance of the edge loop from the reference magenta point is given as a percentage of the edge length.

#### Chapter 7. Advanced Mesh Modelling

You can force the edge to move in 10% steps by keeping **CTRL** pressed. You can flip the reference vertex of the reference edge (the magenta point) with **FKEY** (Figure 7-19, center right).

By clicking **LMB** the edge loop is created, all faces and internal edges of the face loop are split in half at the points highlighted by the cyan edge loop. (Figure 7-19, right).



Figure 7-19. Splitting a Faceloop.

This is a really useful way to refine a mesh in a SubSurface-friendly way.

By default the new, cyan, edge loop is created so that each edge is divided into two parts which are proportional one to the other and the proportionality ratio is the percentage given on the header (Figure 7-20, left). You can force the new edge loop to stay at a given, fixed, distance from the edge loop to which the reference vertex belongs by switching proportional mode off with **PKEY**. This turns the highlighted edgeloop blue too (Figure 7-20, center). **PKEY** acts as an on/off switch.



Figure 7-20. Proportional and Smooth face cuts.

Furthermore, by default, new vertices for the new edge loop are placed exactly on the pre-existing edges. This keeps subdivided faces flat. If a smoother result is desired **SKEY** can be used, prior to finalazing the split, to set smooth mode on/off. If smooth mode is on then new vertices are not on the previous edge any more but displaced in the direction of the normal to the edge by a given percentage. A pop up asks for the percentage after **LMB** is pressed to finalize the split (Figure 7-20, right).

Note: Both Face Loop tools are present in the KKEY menu too.

### **Knife Tool**

The Knife Tool works by subdividing edges if both their verts are selected and the edge is intersected by a user-drawn 'knife' line. For example, if you wish to cut a hole only in the front of a sphere, you can select only the front vertices, and then draw the line with the mouse.

To test the tool add a Grid Mesh. You will be in EditMode and all vertices are selected. Press **SHIFT-K** to activate the Knife Tool. You are prompted to choose the type of cut: Exact will divide the edges exactly where the knife line crosses them, Centers divides an intersected edge at its midpoint. For this cut, we chose Centers.

Now you can click **LMB** and start drawing. If you move and click **LMB** you draw straight segments from clicked point to clicked point; if you hold **LMB** pressed while dragging you draw freehand lines. The polylines can be drawn with an arbitrary number of segments, but the intersection routines only detect one crossing per edge. Crossing back over an edge multiple times does not perform additional cuts on it. **MMB** constrains drawing to an axis as expected. Snap to grid is not currently implemented, but is being looked at for future releases. When you have finished drawing your line, hit **ENTER** to confirm the cut. **ESC** at any time cancels the operation. Figure 7-21 shows some examples.



Figure 7-21. Center knife with polyline (top); Exact Knife with single segment (middle) and Exact freehand knife (bottom).

**Note:** With a large mesh, it will be quicker to select a smaller number of vertices, those defining only the edges you plan to split since the Knife will save time in testing selected vertices for knife trail crossings.

## **Bevelling Tools**

Relevant to Blender 2.33

Blender has, since version 2.32, a Bevel tool. A Bevel is something that smooths out a sharp edge or corner. True world edges are very seldom exactly sharp. Not even a knife blade edge can be considered perfectly sharp, if you really go for accuracy, and most edges are intentionally bevelled for mechanical and practical reasons.

Blender's Bevel tool is still under heavy development and the current implementation is rather crude since all edges in a given mesh are bevelled. There's no control over edges you want to keep sharp, or edges on nearly flat surfaces, which you don't need to bevel at all.

The Bevel tool can be used in EditMode, and can be accessed via the **WKEY** menu, where an entry reads Bevel (Figure 7-22, left). Once selected, a popup asks for the number of recursions in the bevel (Figure 7-22, center left). If it is one, then each face is reduced in size and each edge becomes a single new face. Tri and quad faces are created as necessary at vertices. If the Recursion number is greater than one, then the aforementioned procedure is applied that number of times, hence for Recur: 2 each edge is transformed into 4 edges, three new faces appear at the edge, smoothing the original one. In general the number of new edges is 2 elevated to the Recur power.

**Vertex Number:** Remember that for each new edge two new vertices are created, and some more vertices are created at an intersection between edges, so your vertex number can quickly become enormous if you bevel with a high recursion!



Figure 7-22. Bevelling a cube.

Once the Recur number is set each face of the mesh receives a yellow highlight (Figure 7-22, center right). By moving the mouse pointer, the yellow highlights shrink or grow, and their current shrinking factor is reported on the windows header. By pressing **CTRL** shrinkage occurs in 0.1 steps, by pressing **SHIFT** fine tuning is possible. By pressing **SPACE** a popup appears, asking you to type in the bevel value.

**LMB** finalizes the operation, **RMB** or **ESC** aborts it. The final result can be seen in (Figure 7-22, right).

## Symmetrical Modelling

#### Relevant to Blender v2.34

You often need to model objects which exhibit some sort of symmetry. For radial, rotational or multiple symmetry the best approach is to carefully model one base structure and then, as a last step, duplicate the base cell via SpinDup or whichever command is most appropriate.

For objects with bilateral symmetry, those with one plane of symmetry, such as most animals (humans included) and many machines, the above method implies modelling one half of the object, and then mirroring a duplicate of the first half to get the whole object.

Since it is usually difficult to attain correct proportions by only modelling a half, it is possible to duplicate the half before it is completely modelled, and act on one half and automatically update the other.



Figure 7-23. A plane.

In Front View add a plane or whatever (Figure 7-23). Consider it as a starting point for one half of the object. Let's say the object's right half, which for us in frontal view is on the left of the screen. The plane of symmetry is the yz plane. Move the mesh, in EditMode, so that it is completely on the left of center. Delete some nodes, and add some others, to give it its general shape, as in Figure 7-24.



Figure 7-24. Right half.

Now switch to ObjectMode and, with the half selected, make a *linked* duplicate with **ALT-D**. Press **ESC** to exit from Grab Mode and press **NKEY**. In the Numeric input panel which appears, set Sizex to -1 (Figure 7-25). This effectively mirrors the linked duplicate with respect to the Object's center, hence the importance of keeping the center on the plane of symmetry.

🗙 🔻 Transform Prope		
OB: Plane.001	Par:	
<ul> <li>LocX: 0.000 →</li> </ul>		
<ul> <li>LocY: 0.000 -&gt;</li> </ul>	0	
<ul> <li>LocZ: 0.000 →</li> </ul>	$\sim$	
BotX: 90.000 →	SizeX: -1.000 → 1	
RotY: 0.000	✓ SizeY: 1.000 →	
RotZ: 0.000	<ul> <li>SizeZ: 1.000 →</li> </ul>	

Figure 7-25. Mirroring the linked duplicate.

Having duplicated the Object as a linked duplicate implies that the two objects share the *same* mesh data, which is implicitly mirrored by the unitary negative scaling along the x axis, which is normal to the symmetry plane.

Now you can edit either of the two halves. Since they share mesh data any change, be it an extrude, delete, face loop cut etc. immediately reflects on the other side (Figure 7-26).



Figure 7-26. Editing one half.

By carefully editing one half, and possibly by using a blueprint as a background to provide guidelines, very interesting results can be achieved (Figure 7-27, left).



Figure 7-27. A head. Left: EditMode; Center: ObjectMode; Right: Joined.

As a final step, when symmetrical modelling is complete, the two halves must be selected and joined into a single Object (**CTRL-J**). This makes the seam (very visible in Figure 7-27, center) disappear. Once you have a single object (Figure 7-27, right), you can start modelling the subtle asymmetries which every being has.

**Note:** In Blender 2.33 and earlier versions the OpenGL implementation causes mirrored linked duplicates to have wrong normals, so that one of the two halves is black. This is fixed in Blender 2.34, but older versions can use this technique anyway by setting the mesh to single sided while symmetrical modelling is used.

## **Proportional Editing Tool**

#### *Relevant to Blender v2.31*

When working with dense meshes, it can become difficult to make subtle adjustments to the vertices without causing nasty lumps and creases in the model's surface. When you face situations like these, use the proportional editing tool. It acts like a magnet to smoothly deform the surface of the model, without creating lumps and creases.

In a top-down view, add a plane mesh to the scene with SPACE>>Add>>Mesh>>Plane. Subdivide it a few times with WKEY>>Subdivide (or by clicking on the Subdivide button in the Editing Context Mesh Tools Panel) to get a relatively dense mesh (Figure 7-28). Or, add a grid directly via SPACE>>Add>>Mesh>>Grid, specifying the number of vertices in each direction. When you are finished, deselect all vertices with AKEY.

**Vertex limit:** In Blender, up to 2.31, a single mesh can have no more than 65,000 vertices. From 2.32 onwards meshes can have 2 billion vertices.



Figure 7-28. A planar dense mesh.

Select a single vertex in the mesh by clicking it with **RMB** (Figure 7-29).



Figure 7-29. A planar dense mesh with just one selected vertex.

While still in EditMode, activate the proportional editing tool by pressing **OKEY** or by using the Mesh>>Proportional Editing Menu entry (Figure 7-30 top).



Figure 7-30. Proportional Editing icon and schemes

Switch to a front view (**NUM 1**) and activate the grab tool with **GKEY**. As you drag the point upwards, notice how nearby vertices are dragged along with it (Figure 7-31).

Change the curve profile used by either using the Mesh>>Proportional Faloff submenu or by pressing **SHIFT-O** to toggle between the two options Sharp and Smooth. Note that you cannot do this while you are in the middle of a proportional editing operation; you will have to press **ESC** to cancel the editing operation before you can change the curve.

When you are satisfied with the placement of the vertex, press **LMB** to fix its position. If you are not satisfied, nullify the operation and revert your mesh to the way it looked before you started dragging the point with **ESC** key.



Figure 7-31. Different 'Magnets' for proportional Editing.

While you are editing you can increase or decrease the radius of influence (shown by the dotted circle in Figure 7-31) by pressing **NUM+** and **NUM-** respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly. You can also use **MW** to enlarge and shrink the circle.

You can use the proportional editing tool to produce great effects with the scaling (SKEY) and rotation (RKEY) tools, as Figure 7-32 shows.



Figure 7-32. A landscape obtained via Proportional Editing

Combine these techniques with vertex painting to create fantastic landscapes. Figure 7-33 shows the results of proportional editing after the application of textures and lighting.



Figure 7-33. Final rendered landscape

Chapter 7. Advanced Mesh Modelling

### Noise

*Relevant to Blender v2.31* 

The Noise function allows you to displace vertices in meshes based on the greyvalues of a texture applied to it. That way you can generate great landscapes or carve text into meshes.

Specials		
Subdivide		
Subdivide Fractal		
Subdivide Smooth		
Merge		
Remove Doubles		
Hide		
Reveal		
Select swap		
Flip Normals		
Smooth		

#### Figure 7-34. Subdivide tool

Add a plane and subdivide it at least five times with the special menu **WKEY**>>Subdivide (Figure 7-34). Now add a material and assign a Clouds texture to it. Adjust the NoiseSize: to 0.500. Choose white as the color for the material and black as the texture color, to give us good contrast for the noise operation.

🔻 Mesh Tools				
Beauty	Subdivide	Fract Subd		
Noise	Hash	Xsort		
To Sphere	Smooth	Split		
Flip Normals	Rem Double	Limit: 0.001		
	Extrude			
Screw	Screw Spin Spin Dup			
I Degr: 90 ⊧	∢ Steps: 9 ≽	🔹 Turns: 1 🕨		
Кеер С	Keep Original Clockw			
Extrude Dup Offset: 1.000 >				

Figure 7-35. Noise button in EditButtons

Ensure that you are in EditMode and that all vertices are selected, then switch to the Editing Context **F9**. Press the Noise button in the Mesh Tools Panel (Figure 7-35) several times until the landscape looks nice. Figure 7-36 shows the original - textured - plane as well as what happens as you press Noise. Remove the texture from the landscape now because it will disturb the look. Then add some lights, some water, set smooth and SubSurf the terrain, and so on. (Figure 7-37).



Figure 7-36. Noise application process. From top left to bottom right: Plane with texture, sub-divided plane, "Noise" button hit 2, 4, 6 and 8 times.



Figure 7-37. Noise generated landscape

**Note:** The noise displacement always occurs along the mesh's z coordinate, which is along the direction of the z axis of the Object local reference.

# **Decimator Tool**

### Relevant to Blender v2.33

The Decimator tool is an often overlooked feature which allows you to reduce the vertex/face count of a mesh with minimal shape changes.

#### Chapter 7. Advanced Mesh Modelling

This is not applicable to meshes which have been created by modelling carefully and economically, where all vertices and faces are necessary to correctly define the shape, but if the mesh is the result of complex modelling, with proportional editing, successive refinements, possibly some conversions from SubSurfed to non-SubSurfed meshes, you might very well end up with meshes where lots of vertices are not really necessary.

A simple example is a plane, and a 4x4 undeformed Grid object. Both render exactly the same, but the plane has 1 face and 4 vertices, while the grid has 9 faces and 16 vertices, hence lots of unneeded vertices and faces.

The Decimator Tool (Figure 7-38) allows you to eliminate these unneeded faces. Its NumButton reports the number of faces of the selected mesh in ObjectMode. The decimator tool only handles triangles, so each quadrilateral face is implicitly split into two triangles for decimation.

▼ Mesh				
Auto Smooth		<ul> <li>Decir</li> </ul>	🔹 Decimator: 188 🛛 🕨	
< Deg	gr: 30 🛛 🕨	Apply Cancel		
SubSurf	Catmul ≎	TexMesh:		
VibduZ	/: 1 ▶   < 1 ▶ time!		Centre	
L Ob	umai	Centre New		
Sticky	Make		Centre Cursor	
VertCol	Make	SlowerDra	Double Sided	
TexFac	Make	FasterDra	No V.Normal Fli	

Figure 7-38. Decimator buttons.

Let's consider the example we used in the Bevel section. As you might notice there is a tiny triangular face on each cube vertex which might very well be unnecessary (Figure 7-39, top left). The header says the cube has 98 faces and 96 vertices. The Decimator button says the cube has 188 triangular face, namely 90 quads (which are 180 tris) and 8 tris.



Figure 7-39. Decimator at work.

By changing the number in the decimator NumBut, by either clicking or typing it in, the mesh immediately changes to triangles only. As the number gets lower, faces disappear one after the other. Blender causes coplanar faces and vertices on aligned edges to disappear first. This tends to keep the shape of the mesh. As more and more faces are asked to be removed faces less and less coplanar and vertices less and less colinear are merged, hence sensible shape change might occur (Figure 7-39, top center).

In this particular case, if we just want the central tri face of each cube vertex to disappear we expect the final mesh to be 2x6=12 faces for each cube face, 2x3x12=72 faces for each bevelled edge, and 9x8=72 faces for each bevelled vert, totalling 156 faces. It is very uncommon to know beforehand how many faces the final mesh can have, usually you must look carefully at the mesh in a 3D window to check that the shape is still good.

The two buttons below the Decimator finalize or cancel the decimation. Once it is finalized triangles are not shown any more (Figure 7-39, top right) but the mesh is nevertheless made only of triangles (Figure 7-39, bottom left). You can revert to quads if you so wish, by selecting all vertices and hitting **ALT-J** (Figure 7-39, bottom center). This way we reduce the vertex count to 80 and face count to 82 without any noticeable shape loss. It might look a small gain, but if this cube is going to be dupliverted on a particle system with 1000 particles it might be worth it.



Figure 7-40. Decimated landscape, top: original; middle: lightly decimated; bottom: heavily decimated.

Figure 7-40 shows a landscape generated via a careful application of the Noise technique described earlier, on a quite vast grid. On top, the result for the original mesh and below, two different levels of decimation. To the eye the difference is indeed almost unnoticeable, but as the vertex count goes down there is a huge gain.

# **Chapter 8. Meta Objects**

#### Relevant to Blender v2.31

Meta Objects consist of spherical, tubular and cuboidal elements that can affect each other's shape. You can only create rounded and fluid 'mercurial', or 'clay-like', forms that exist *procedurally*, that is are computed dynamically. Use Meta Objects for special effects or as a basis for modelling.

Meta Objects are also called *implicit surfaces*, again to point out that they are not *explicitly* defined by vertices (as meshes are) or control points (as surfaces are).

Meta Objects are defined by a *directing structure* which can be seen as the source of a static field. The field can be either positive or negative and hence the field generated by neighbouring directing structures can attract or repel.

The implicit surface is defined as the surface where the 3D field generated by all the directing structures assumes a given value. For example a Meta Ball, whose directing structure is a point, generates an isotropic field around it and the surfaces at constant field value are spheres centered at the directing point. Two neighbouring Meta balls interact and, if they are close enough, the two implicit surfaces merge into a single surface (Figure 8-1).



Figure 8-1. Two Metaballs

In fact, Meta Objects are nothing more than mathematical formulas that perform logical operations on one another (AND, OR), and that can be added and subtracted from each other. This method is also called CSG, Constructive Solid Geometry. Because of its mathematical nature, CSG uses little memory, but requires lots of CPU to compute. To optimize this the implicit surfaces are *polygonized*. The complete CSG area is divided into a 3D grid, and for each *edge* in the grid a calculation is made, and if (and more importantly where) the formula has a turning point, a 'vertex' for the *polygonize* is created.

To create a Meta Object press **SPACE** and select Add>>MBall. You can select the base shapes: Ball, Tube, Plane, Ellipsoid and Cube.

MetaBalls have a point directing structure, MetaTubes have a segment as a directing structure, MetaPlanes a plane, and MetaCubes a cube. The underlying structure becomes more evident as you lower the Wiresize and raise the Threshold values in the Meta Ball Panel.

#### Chapter 8. Meta Objects

When in EditMode, you can move and scale the Meta Objects as you wish. This is the best way to construct static - as opposed to animated - forms. Meta Objects can also influence each other *outside* EditMode. When outside EditMode you have much more freedom; the balls can rotate or move and they get every *transformation* of the Parent Objects. This method requires more calculation time and is thus somewhat slow.

The following rules describe the relation between Meta Objects:

- All Meta Objects with the same 'family' name (the name without the number) influence each other. For example "MBall", "MBall.001", "MBall.002", "MBall.135". Note here that we are not talking about the name of the MetaBall ObData block.
- The Object with the family name *without* a number determines the basis, the resolution, *and* the transformation of the polygonize. It also has the Material and texture area and will be referred to as *base* Meta Object.

Only one Material can be used for a Meta Object set. In addition, Meta Objects save a separate texture area; this normalises the coordinates of the vertices. Normally the texture area is identical to the *boundbox* of all vertices. The user can force a texture area with the **TKEY** command (outside of EditMode).

The fact that the base Object dictates the polygonalization implies that, if we have two Meta Objects and we move one of them we will see the polygonalization of the *non-base* Object change during motion, regardless of which of the two object is actually moving.

The Meta Ball Panel in Editing context offers few settings. If in Object Mode, only this Panel is present. You can define the polygonalization average dimension both in the 3D Viewport via the Wiresize Num Button, and at rendering time via the Rendersize Num Button. The lower these are, the smoother the Meta Object is, and the slower its computation.

The Threshold Num Button is an important setting for MetaObjects. It controls the 'field level' at which the surface is computed. To have finer control, when in EditMode, the Stiffness Num Button of the Meta Ball Tools Panel allows you to enlarge or reduce the MetaObject's field of influence.

In this latter Panel you can also change the Meta Object type and set it negative (that is subtractive, rather than additive) with other Meta Objects of the same set.
# **Chapter 9. Curves and Surfaces**

Curves and surfaces are objects like meshes, but differ in that they are expressed in terms of mathematical functions, rather than as a series of points.

Blender implements Bézier and Non Uniform Rational B-Splines (NURBS) curves and surfaces. Both, though following different mathematical laws, are defined in terms of a set of "control vertices" which define a "control polygon." The way the curve and the surface are interpolated (Bézier) or attracted (NURBS) by these might seem similar, at first glance, to Catmull-Clark subdivision surfaces.

When compared to meshes, curves and surfaces have both advantages and disadvantages. Because curves are defined by less data, they produce nice results using less memory at modelling time, whereas the demands increase at rendering time.

Some modelling techniques, such as extruding a profile along a path, are only possible with curves. But the very fine control available on a per-vertex basis on a mesh, is not possible with curves.

There are times when curves and surfaces are more advantageous than meshes, and times when meshes are more useful. If you have read the previous Chapter, and if you read this you will be able to choose whether to use meshes or curves.

## Curves

Relevant to Blender v2.31

This section describes both Bézier and NURBS curves, and shows a working example of the former.

#### **Béziers**

Bézier curves are the most commonly used type for designing letters or logos. They are also widely used in animation, both as paths for objects to move along and as IPO curves to change the properties of objects as a function of time.

A control point (vertex) of a Bézier curve consists of a point and two handles. The point, in the middle, is used to move the entire control point; selecting it also selects the other two handles, and allows you to move the complete vertex. Selecting one or two of the other handles allows you to change the shape of the curve by dragging the handles.

A Bézier curve is tangent to the line segment which goes from the point to the handle. The 'steepness' of the curve is controlled by the handle's length.

There are four types of handles (Figure 9-1):

- Free Handle (black). This can be used in any way you wish. Hotkey: **HKEY** (toggles between Free and Aligned);
- Aligned Handle (purple). These handles always lie in a straight line. Hotkey: **HKEY** (toggles between Free and Aligned);
- Vector Handle (green). Both parts of a handle always point to the previous handle or the next handle. Hotkey: **VKEY**;
- Auto Handle (yellow). This handle has a completely automatic length and direction, set by Blender to ensure the smoothest result. Hotkey: **SHIFT-H**.



Figure 9-1. Types of Handles for Bézier curves

Handles can be *grabbed*, *rotated* and *scaled* exactly as ordinary vertices in a mesh would.

As soon as the handles are moved, the type is modified automatically:

- Auto Handle becomes Aligned;
- Vector Handle becomes Free.

Although the Bézier curve is a continuous mathematical object it must nevertheless be represented in discrete form from a rendering point of view.

This is done by setting a *resolution* property, which defines the number of points which are computed between every pair of control points. A separate *resolution* can be set for each Bézier curve (Figure 9-2).

e D	efR	esolL	l: 6	Set
Bad	:k	Fro	nt 📘	3D
4	Wi	dth: 1	1.00	0 🕨
4	E	kt1:0	).000	) 🛛 🕨
4	E	<t2: 0<="" td=""><td>000.</td><td>) 🛛 🕨</td></t2:>	000.	) 🛛 🕨
4	Be	vRes	ol: C	) - F
Bevt	Db:			

Figure 9-2. Setting Bézier resolution.

## NURBS

NURBS curves are defined as rational polynomials, and are more general, strictly speaking, than conventional B-Splines and Bézier curves inasmuch they are able to exactly follow any contour. For example a Bézier circle is a polynomial *approximation* of a circle, and this approximation is noticeable, whereas a NURBS circle is *exactly* a circle. NURBS curves have a large set of variables, which allow you to create mathematically pure forms (Figure 9-3). However, working with them requires a little more theory:

Convert	Make Knots	
Poly	Uniform U	
Bezier	Endpoint U	
Bspline	Bezier U	
Cardinal	✓ Order U: 2 →	<v:2►< td=""></v:2►<>
Nurb		V:12
Set Weight	Weight: 1.000	1.0 sqrt(0.5

Figure 9-3. Nurbs Control Buttons.

- *Knots.* Nurbs curves have a *knot* vector, a row of numbers that specifies the parametric definition of the curve. Two pre-sets are important for this. Uniform produces a uniform division for closed curves, but when used with open ones you will get "free" ends, which are difficult to locate precisely. Endpoint sets the knots in such a way that the first and last vertices are always part of the curve, which makes them much easier to place;
- *Order.* The *order* is the 'depth' of the curve calculation. Order '1' is a point, order '2' is linear, order '3' is quadratic, and so on. Always use order '5' for Curve paths because it behaves fluidly under all circumstances, without producing irritating discontinuities in the movement. Mathematically speaking this is the order of both the Numerator and the Denominator of the rational polynomial defining the NURBS;
- *Weight.* Nurbs curves have a 'weight' per vertex the extent to which a vertex participates in the "pulling" of the curve.



Figure 9-4. Setting Nurbs Control polygon and weights.

Figure 9-4 shows the Knot vector settings as well as the effect of varying a single knot weight. As with Béziers, the resolution can be set on a per curve basis.

## Working example

Blender's curve tools provide a quick and easy way to build great looking extruded text and logos. We will use these tools to turn a rough sketch of a logo into a finished 3D object.

Figure 9-5 shows the design of the logo we will be building.



Figure 9-5. The sketched logo

First, we will import our original sketch so that we can use it as a template. Blender supports TGA, PNG, and JPG format images. To load the image, select

the View>>Background Image... menu entry of the 3D Window you are using. A transparent panel will appear, allowing you to select a picture to use as a background. Activate the BackGroundPic button and use the LOAD button to locate the image you want to use as a template (Figure 9-6). You can set the "strength" of the background pic with the Blend slider.

× 🔻 Background
BackGroundPic Size: 5.00
chapter_07_curves_surfaces/logo.png ×
LOAD Blend:0.500
Center X: 0.00

Figure 9-6. 3D window settings.

Get rid of the Panel with **ESC** or by pressing the x button in the panel header (Figure 9-7). You can hide the background image when you are finished using it by returning to the Panel and deselecting the BackGroundPic button.



Figure 9-7. Logo sketch loaded as background

Add a new curve by pressing **SPACE**>>Curve>>Bezier Curve. A curved segment will appear and Blender will enter EditMode. We will move and add points to make a closed shape that describes the logo you are trying to trace.

You can add points to the curve by selecting one of the two endpoints, then holding **CTRL** and clicking **LMB**. Note that the new point will be connected to the previously selected point. Once a point has been added, it can be moved by selecting the control vertex and pressing **GKEY**. You can change the angle of the curve by grabbing and moving the handles associated with each vertex (Figure 9-8).



Figure 9-8. Bézier handles

You can add a new point between two existing points by selecting the two points and pressing **WKEY**>>Subdivide (Figure 9-9).



Figure 9-9. Adding a Control Point.

Points can be removed by selecting them and pressing **XKEY**>>Selected. To cut a curve in two, select two adjacent control vertices and press **XKEY**>>Segment.

To make sharp corners, select a control vertex and press **VKEY**. You will notice that the color of the handles changes from purple to green (Figure 9-10). At this point, you can move the handles to adjust the way the curve enters and leaves the control vertex (Figure 9-11).



Figure 9-10. Vector (green) handles.



Figure 9-11. Free (black) handles.

To close the curve and turn it into a single continuous loop, select at least one of the control vertices on the curve and press **CKEY**. This will connect the last point in the curve with the first one (Figure 9-12). You may need to manipulate additional handles to get the shape you want.



Figure 9-12. The finished outline.

Leaving EditMode with **TAB** and entering shaded mode with **ZKEY** should reveal that the curve renders as a solid shape (Figure 9-13). We want to cut some holes into this shape to represent the eyes and wing details of the dragon.

**Surfaces and Holes:** When working with curves, Blender automatically detects holes in the surface and handles them accordingly to the following rules. A closed curve is always considered the boundary of a surface and hence rendered as a flat surface. If a closed curve is completely included within another one, the inner one is subtracted from the outer one, effectively defining a hole.



Figure 9-13. Shaded logo.

Return to wireframe mode with **ZKEY** and enter EditMode again with **TAB**. While still in EditMode, add a circle curve with **SPACE**>>Curve>>Bezier Circle (Figure 9-14). Scale the circle down to an appropriate size with **SKEY** and move it with **GKEY**.



Figure 9-14. Adding a circle.

Shape the circle using the techniques we have learned (Figure 9-15). Remember to add vertices to the circle with **WKEY**>>Subdivide.



Figure 9-15. Defining the eye.

Create a wing cutout by adding a Bézier circle, converting all of the points to sharp corners, and then adjusting as necessary. You can duplicate this outline to save time when creating the second wing cutout. To do so, make sure no points are selected, then move the cursor over one of the vertices in the first wing cutout and select all linked points with **LKEY** (Figure 9-16). Duplicate the selection with **SHIFT-D** and move the new points into position.



Figure 9-16. Defining the wings.

To add more geometry that is not connected to the main body (placing an orb in the dragon's curved tail for example), use the **SHIFT-A** menu to add more curves as shown in Figure 9-17.



Figure 9-17. Orb placement within the tail.

Now that we have the curve, we need to set its thickness and beveling options. With the curve selected, go to the EditButtons (F9) and locate the Curves and Surface panel. The Extl parameter sets the thickness of the extrusion while Extl sets the size of the bevel. BevResol sets how sharp or curved the bevel will be.

Figure 9-18 shows the settings used to extrude this curve.

✓ DefResolU: 6 → Set		
Back Front 3D		
<ul> <li>Width: 1.000 →</li> </ul>		
✓ Ext1:0.010 →		
BevOb:		

Figure 9-18. Bevel settings

**From Curves to Meshes:** To perform more complex modelling operations, convert the curve to a mesh with ALT-C>>Mesh. Note that this is a one-way operation: you cannot convert a mesh back into a curve.

When your logo model is complete, you can add materials and lights and make a nice rendering (Figure 9-19).



Figure 9-19. Final rendering.

## **Surfaces**

Relevant to Blender v2.31

Surfaces are actually an extension of NURBS curves. In Blender they are a separate ObData type.

Whereas a curve produces only one-dimensional interpolation, Surfaces have a second extra dimension. The first dimension is U, as for curves, and the second is V. A two-dimensional grid of control points defines the form for these NURBS surfaces.

Use Surfaces to create and revise fluid curved surfaces. Surfaces can be cyclical in both directions, allowing you to easily create a 'donut' shape, and they can be drawn as 'solids' in EditMode (zbuffered, with OpenGL lighting). This makes working with surfaces quite easy.

**Note:** Currently Blender has a basic tool set for Surfaces, with limited Ability to create holes and melt surfaces. Future versions will contain increased functionality in these areas.

You can take one of the various surface 'primitives' from the ADD menu as a starting point (Figure 9-20). Note that you can choose 'Curve' and 'Circle' from the 'surface' menu! This is possible because NURBS curves are intrinsically NURBS Surfaces, simply having one dimension neglected.

**Note:** A NURBS 'true' curve and a NURBS 'surface' curve are not interchangeable, as you'll see as you follow the extruding process below and the skinning section further on.

	Lattice		
	Armatu	re	
	Lamp		
	Camera	a	
	Empty		NURBS Curve
	Text		NURBS Circle
	MBall	<b>&gt;</b>	NURBS Surface
	Surface	9 ▶	NURBS Tube
	Curve	•	NURBS Sphere
	Mesh	•	NURBS Donut
Object	Add	Select	
Edit	Transform	View	

Figure 9-20. Add surface menu.

When you add a 'surface' curve you can create a true surface simply by extruding the entire curve (**EKEY**). Each edge of a surface can then be extruded as you wish to form the model. Use **CKEY** to make the U or V direction cyclic. Be sure to set the 'knots' to Uniform or Endpoint with one of the pre-sets from the EditButtons Curve Tools panel.

When working with surfaces, it is handy to always work on a complete column or row of vertices. Blender provides a selection tool for this: **SHIFT-R**, "Select Row". Starting from the last selected vertex, a complete row of vertices is *extend* selected in the 'U' or 'V' direction. Choose Select Row again with the same vertex to toggle between the 'U' of 'V' selection.



Figure 9-21. A sphere surface

NURBS can create pure shapes such as circles, cylinders, and spheres (but note that a Bézier circle is not a pure circle.) To create pure circles, globes, or cylinders, you must set the weights of the vertices. This is not intuitive, and you should read more on NURBS before trying this.

Basically, to produce a circular arc from a curve with three control points, the end points must have a unitary weight, while the weight of the central control point must be equal to one-half the cosine of half the angle between the segments joining the points. Figure 9-21 shows this for a globe. Three standard numbers are included as presets in the EditButtons Curve Tools panel (Figure 9-22).

Note: To read the weight of a selected vertex, press the NKEY.

Convert	Make Knots	
Poly	Uniform U	
Bezier	Endpoint U	
Bspline	Bezier U	
Cardinal	🔹 Order U: 2 🕨	<v:2►< td=""></v:2►<>
Nurb		V:12
Set Weight	Weight: 1.000	1.0
	sqrt(2)/ 0.25	sqrt(0.5

Figure 9-22. Pre-set weights

## Text

Relevant to Blender v2.31



Figure 9-23. Text Examples.

Text is a special curve type for Blender. Blender has its own built-in font but can use external fonts too, including both PostScript Type 1 fonts and True Type fonts (Figure 9-23).

Open Blender or revert to a fresh scene by pressing **CTRL-X**. Add a TextObject with the Toolbox (**SPACE**>>Add>>Text). You can edit the text with the keyboard in Edit-Mode; a text cursor shows your position in the text. When you leave EditMode with **TAB**, Blender fills the text-curve, producing a flat filled object that is renderable at once.

Now go to the EditButtons **F9** (Figure 9-24).

🔻 Curve and Surface		▼ Font
UV Orco	DefResolU: 6 Set Back Front 3D	 
Centre Centre New Centre Cursor	Width: 1.000 Ext1: 0.000 Ext2: 0.000	Left Right Middle Flush TextOnCurve: Ob Family:
	BevResol: 0 BevOb:	Size: 1.000         Linedist: 1.000           Spacing: 1.000         Y offset: 0.00           Shear: 0.000         X offset: 0.00

Figure 9-24. Text edit buttons

As you can see in the Font panel MenuButton, Blender uses its own <builtin> font by default when creating a new TextObject. Now click Load Font. Browse in the FileWindow to a directory containing PostScript Type 1 or True Type fonts and load a new font. (You can download several free PostScript fonts from the web, and Microsoft Windows includes many True Type fonts of its own - though in the latter case be aware that some of them are copyrighted!).

Try out some fonts. Once you've loaded a font, you can use the MenuButton to switch the font for a TextObject.

For now we have only a flat object. To add some depth, we can use the Ext1: and Ext2: buttons in the Curve and Surface panel just as we did with curves.

Use the TextOnCurve: option to make the text follow a 2D-curve. Use the alignment buttons above the TextOnCurve: text field in the Font panel to align the text on the curve.

One particularly powerful Blender function is that a TextObject can be converted with **ALT-C** to a Bézier curve, which allows you to edit the shape of every single character on the curve. This is especially handy for creating logos or when producing custom lettering. The transformation from text to curve is irreversible and, of course, a further transformation from curve to mesh is possible too.

## **Special Characters**

Normally, a Font Object begins with the word "Text", which can be deleted simply with **SHIFT-BACKSPACE**. In EditMode, the Text Object only reacts to text input. Nearly all of the hotkeys are disabled. The cursor can be moved with the arrow keys. Use **SHIFT-ARROWLEFT** and **SHIFT-ARROWRIGHT** to move the cursor to the end of the lines or to the beginning or end of the text.

Nearly all 'special' characters are available. A summary of these characters follows:

• ALT-c: copyright

- ALT-f: Dutch Florin
- ALT-g: degrees
- ALT-1: British Pound
- ALT-r: Registered trademark
- ALT-s: German S
- **ALT-x**: Multiply symbol
- ALT-y: Japanese Yen
- ALT-1: a small 1
- ALT-2: a small 2
- ALT-3: a small 3
- ALT-?: Spanish question mark
- ALT-!: Spanish exclamation mark
- ALT->: a double >>
- ALT-<: a double <<

All the characters on your keyboard should work, including stressed vowels and so on. If you need special characters (such as accented letters, which are not there on a US keyboard) you can produce many of them using a combination of two other characters. To do so, press **ALT-BACKSPACE** within the desired combination, and then press the desired combination to produce the special character. Some examples are given below.

- AKEY, ALT-BACKSPACE, TILDE: ã
- AKEY, ALT-BACKSPACE, COMMA: à
- AKEY, ALT-BACKSPACE, ACCENT: á
- AKEY, ALT-BACKSPACE, OKEY: å
- EKEY, ALT-BACKSPACE, QUOTE: ë
- OKEY, ALT-BACKSPACE, SLASH: Ø

You can also add complete ASCII files to a Text Object. Save the file as **/tmp/.cutbuffer** and press **ALT-V**.

Otherwise you can write your text in a Blender Text Window, load text into such a window, or paste it into the window from the clipboard and press **ALT-M**. This creates a new Text Object from the content of the text buffer (Up to 1000 characters).

## **Extrude Along Path**

Relevant to Blender v2.31

The "Extrude along path" technique is a very powerful modelling tool. It consists of creating a surface by sweeping a given profile along a given path.

Both the profile and the path can be a Bézier or a NURBS curve.

Let's assume you have added a Bézier curve and a Bézier circle as separate objects to your scene (Figure 9-25).



Figure 9-25. Profile (left) and path (right).

Play a bit with both to obtain a nice 'wing-like' profile and a fancy path (Figure 9-26). By default, Béziers exist only on a plane, and are 2D objects. To make the path span in all three directions of space, as in the example shown above, press the 3D button in the Curve EditButtons (**F9**) Curve and Surface panel (Figure 9-27).



Figure 9-26. Modified profile (left) and path (right).

<ul> <li>Curve and Surface</li> </ul>	
UV Orco	✓ DefResolU: 6 → Set
	Back Front 3D
Centre	
Centre New	✓ Width: 1.000 →
Centre Cursor	✓ Ext1: 0.000 →
🔹 PathLen: 100 🔹	✓ Ext2: 0.000 →
CurvePath CurveFoll	🔹 BevResol: 0 🕞
PrintLen 0.0000	BevOb:

Figure 9-27. 3D Curve button.

Now look at the name of the profile object. By default it is "CurveCircle" and it is shown on the **NKEY** panel when it is selected. You can change it by **SHIFT-LMB** on

the name, if you like (Figure 9-28).

×v	Transform Pr	ope	rties		
OB:	CurveCircle		Par	:	
		_			
-	LOCX: -1.18	- (F			
4	LocY: 0.00	÷.			
-	LocZ: 0.00	÷.			
4	RotX: 0.00	÷	4	SizeX: 0.52	- (F
-	RotY: 0.00	÷.	-	SizeV: 0.52	ŀ
-	RotZ: 0.00	÷.	4	SizeZ: 0.52	- P

Figure 9-28. Profile name.

Now select the path. In its EditButtons locate the BevOb: Text Button in the Curve and Surface panel and write in there the name of the profile object. In our case "CurveCircle" (Figure 9-29).

▼ Curve and Surface	
UV Orco	✓ DefResolU: 6 ▶ Set
Cantra	Back Front 3D
Centre	
Centre New	♦ Width: 1.000 ▶
Centre Cursor	← Ext1: 0.000 →
🔹 PathLen: 100 🕨	
CurvePath CurveFoll	🔹 BevResol: 0 🕞
PrintLen 0.0000	BevOb:CurveCircle

Figure 9-29. Specify the Profile on the path.

The result is a surface defined by the Profile, sweeping along the path (Figure 9-30).



Figure 9-30. Extrusion result.

To understand the results, and hence obtain the desired effects it is important to understand the following points:

- The profile is oriented so that its z-axis is tangent (*i.e.* directed along) the path and that its x-axis is on the plane of the path; consequently the y-axis is orthogonal to the plane of the path;
- If the path is 3D the "plane of the path" is defined locally rather than globally and is visually rendered, in EditMode, by several short segments perpendicular to the path (Figure 9-31);
- The y-axis of the profile always points upwards. This is often a source of unexpected results and problems, as we'll explain later on.



Figure 9-31. Path local plane.

**Tilting:** To modify the orientation of the local path plane select a control point and press **TKEY**. Then move the mouse to change the orientation of the short segments smoothly in

the neighborhood of the control point. **LMB** fixes the position, and **ESC** reverts to previous state.

With the y-axis constrained upwards, unexpected results can occur when the path is 3D and the profile being extruded comes to a point where the path is exactly vertical. Indeed if the path goes vertical and then continues to bend there is a point where the y-axis of the profile should begin to point downwards. If this occurs, since the y-axis is constrained to point upwards there is an abrupt 180° rotation of the profile, so that the y-axis points upwards again.

Figure 9-32 shows the problem. On the left there is a path whose steepness is such that the normal to the local path plane is always upward. On the right we see a path where, at the point circled in yellow, such a normal begins to point down. The result of the extrusion presents an abrupt turn there.



Figure 9-32. Extrusion problems due to y-axis constraint.

The only solutions to this problems are: To use multiple - matching - paths, or to carefully tilt the path to ensure that a normal always points upwards.

**Changing profile orientation:** If the orientation of the profile along the curve is not as you expected, and you want to rotate it for the entire path length, there is a better way to do so than tilting all path control points.

You can simply rotate the profile in EditMode on its plane. This way the profile will change but its local reference will not.

# **Curve Taper**

by Kenneth Styrberg Relevant to Blender v2.35 Taper is a tool for curve objects. In the Edit panel (F9) you have a TaperOb field where you put the name of the curve that will define the width of the curve object.

<ul> <li>Curve and Surface</li> </ul>	
UV Orco	✓ DefResolU: 6 > Set
	Back Front 3D
Centre	
Centre New	✓ Width: 1.000 →
Centre Cursor	< Ext1: 0.000 →
✓ PathLen: 100 →	✓ Ext2: 0.000 →
CurvePath CurveFollo	< BevResol: 0 >
CurveStretch	BevOb:
PrintLen 0.0000	TaperOb:

Figure 9-33. Curve and Surface panel

Note: Important rules

- Only the first curve in a TaperOb is evaluated.
- Starting width is from left to right.
- Negative widths are possible too, but rendering can cause artifacts.
- It scales the width of normal extrusions based on evaluating the taper curve, which means sharp corners in taper curve won't be easily visible.



Figure 9-34. Taper example 1

In Figure 9-34 you can clearly see the effect the left taper curve has on the right curve object. Here the left taper curve is closer to the object center and that results in a smaller curve object to the right.



Figure 9-35. Taper example 2

In Figure 9-35 a control point in the taper curve to the left is moved away from the center and that gives a wider result to the curve object on the right.

**Note:** The curve object is extruded with a curve circle. (See the Section called *Extrude Along Path* for more on curve extruding).

In Figure 9-36, we see the use of a more irregular taper curve added to a curve circle.



Figure 9-36. Taper example 3

# **Curve Deform**

by Kenneth Styrberg Relevant to Blender v2.35

### Introduction

Curve Deform provide a simple but efficient method to define a deformation on a mesh. By parenting a mesh object to a curve, you can deform the mesh along the curve by moving the mesh along, or orthogonal to, the dominant axis.

The Curve Deform work on a dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction, the mesh will traverse along the curve. Moving the mesh in an orthogonal direction will move the mesh object closer or further away from the curve. The default settings in Blender map the Y axis to the dominant axis. When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

**Tip:** Try to position your object over the curve while moving it around. This gives the best control over how the deformation work.

### Interface

When parenting a mesh to a curve (**CTRL-P**), you will be presented with a menu, Figure 9-37. By selecting **Curve Deform** you have enable the Curve Deform function on the mesh object.



Figure 9-37. Make Parent menu.

The dominant axis setting is set on the mesh object. By default the dominant axis in Blender is Y. This can be changed by selecting one of the Track X,Y or Z buttons in Anim Panel, Figure 9-38, in Object Context (F7).

▼ Anim settings				
TrackX Y Z - X - Y - Z UpX Y Z				
Draw Key Draw Key S Powertrack SlowPar				
DupliFrames DupliVerts Rot No Speed				
■ DupSta: 1 → ■ DupOn: 1 →				
✓ DupEnd 100 → ✓ DupOff 0 →				
Offs Ob Offs Par Offs Particle 0.0000				
TimeOffset: 0.00 Automatic Time PrSpeed				

Figure 9-38. Anim settings panel.

Cyclic curves work as expected, where the object deformation traverse along the path in cycles.

**CurveStretch** gives an option to let the mesh object stretch , or squeeze, over the entire curve. This option is in Edit Context (F9) for the curve. See Figure 9-39.

▼ Curve and Surface	
UV Orco	✓ DefResolU: 6 > Set
	Back Front 3D
Centre	
Centre New	✓ Width: 1.000 →
Centre Cursor	✓ Ext1: 0.000 →
✓ PathLen: 100 →	✓ Ext2: 0.000 →
CurvePath CurveFollo	BevResol: 0 >
CurveStretch	BevOb:
PrintLen 0.0000	TaperOb:

Figure 9-39. Curve and Surface panel.

## Example

Lets make a simple example.

**1.** Remove default cube object from scene and add a Monkey! (SHIFT-A -> Add -> Mesh -> Monkey, Figure 9-40)

#### Chapter 9. Curves and Surfaces

	Curve Mesh	ł	Cylinder
Object Edit	Add	Select View	Tube Cone
			Grid
			Monkey

Figure 9-40. Add a Monkey!.

2. Now press TAB to exit Edit Mode. Now add a curve. (SHIFT-A -> Add -> Curve -> Bezier Curve, Figure 9-41)

	Text		Bezier Curve
Æ	Meta	•	Bezier Circle
	Surface		NURBS Curve
E	Mash		NURBS Circle
-J-h	Add	Select	Path
	Transform	View	

Figure 9-41. Add a Curve.

**3.** While in editmode, move the control points of the curve as Figure 9-42, then exit Edit Mode, (TAB).



Figure 9-42. Edit Curve.

**4.** Select the Monkey, **(RMB)**, and then shift select the curve, **(SHIFT-RMB)**. Press **CTRL-P** to open up the Make Parent menu. Select **Curve Deform**. Figure 9-37. The Monkey should be positioned on the curve as Figure 9-43.



Figure 9-43. Monkey on a Curve.

**5.** Now if you select the Monkey, (**RMB**), and move it, (**G**), in the Y-direction, (the dominant axis by default), the Monkey will deform nicely along the curve.

Tip: If you press **MMB** while moving the Monkey you will constrain the movement to one axis only.

**6.** In Figure 9-44, you can see the Monkey at different positions along the curve. To get a cleaner view over the deformation I have activated **SubSurf** with **Subdiv 2** and **Set Smooth** on the Monkey mesh. (**F9** to get Edit options).

**Tip:** Moving the Monkey in other directions than the dominant, will create some odd deformations. Sometime this is what you want to achieve so you'll need to experiment and try it out!



Figure 9-44. Monkey deformations.

# Skinning

Relevant to Blender v2.31

Skinning is the fine art of defining a surface using two or more profiles. In Blender you do so by preparing as many curves of the the desired shape and then converting them to a single NURBS surface.

As an example we will create a sailboat. The first thing to do, in side view (**NUM3**), is to add a Surface Curve. Be sure to add a *Surface* curve and not a curve of Bézier or NURBS flavour, or the trick won't work (Figure 9-45).



Figure 9-45. A Surface curve for skinning.

Give the curve the shape of the middle cross section of the boat, by adding vertices as needed with the Split button and, possibly, by setting the NURBS to 'Endpoint' both on 'U' and 'V' (Figure 9-46) as needed.



Figure 9-46. Profile of the ship.

Now duplicate (SHIFT-D) the curve as many times as necessary, to the left and to the right (Figure 9-47). Adjust the curves to match the various sections of the ship at different points along its length. To this end, blueprints help a lot. You can load a blueprint on the background (as we did for the logo design in this chapter) to prepare all the cross section profiles (Figure 9-48).

Note that the surface we'll produce will transition smoothly from one profile to the next. To create abrupt changes you would need to place profiles quite close to each other, as is the case for the profile selected in Figure 9-48.



Figure 9-47. Multiple profiles along ship's axis.



Figure 9-48. Multiple profiles of the correct shapes.

Now select all curves (with **AKEY** or **BKEY**), and join them by pressing **CTRL-J** and by answering Yes to the question 'Join selected NURBS?'. The profiles are all highlighted in Figure 9-49.



Figure 9-49. Joined profiles.

Now switch to EditMode (**TAB**) and select all control points with **AKEY**; then press **FKEY**. The profiles should be 'skinned' and converted to a surface (Figure 9-50).

**Note:** As should be evident from the first and last profiles in this example, the crosssections need not be defined on a family of mutually orthogonal planes.



Figure 9-50. Skinned surface in edit mode.

Tweak the surface, if necessary, by moving the control points. Figure 9-51 shows a shaded view. You will very probably need to increase Resolu and Relolv to obtain a better shape.



Figure 9-51. Final hull.

**Profile setup:** The only limitation to this otherwise very powerful technique is that all profiles must exhibit the same number of control points. This is why it is a good idea to model the most complex cross section first and then duplicate it, moving control points as needed, without adding or removing them, as we've shown in this example.

## Chapter 10. Materials and Textures

Before you can understand how to design effectively with materials, you must understand how simulated light and surfaces interact in Blender's rendering engine and how material settings control those interactions. A deep understanding of the engine will help you to get the most from it.

The rendered image you create with Blender is a projection of the scene onto an imaginary surface called the *viewing plane*. The viewing plane is analogous to the film in a traditional camera, or the rods and cones in the human eye, except that it receives simulated light, not real light.

To render an image of a scene we must first determine what light from the scene is arriving at each point on the viewing plane. The best way to answer this question is to follow a straight line (the simulated light ray) backwards through that point on the viewing plane and the focal point (the location of the camera) until it hits a renderable surface in the scene, at which point we can determine what light would strike that point. The surface properties and incident light angle tell us how much of that light would be reflected back along the incident viewing angle (Figure 10-1).



Figure 10-1. Rendering engine basic principle.

Two basic types of phenomena take place at any point on a surface when a light ray strikes it: diffusion and specular reflection. Diffusion and specular reflection are distinguished from each other mainly by the relationship between the incident light angle and the reflected light angle.

## Diffusion

#### Relevant to Blender v2.31

Light striking a surface and then re-irradiated via a Diffusion phenomenon will be scattered, i.e., re-irradiated in all directions isotropically. This means that the camera will see the same amount of light from that surface point no matter what the *incident viewing angle* is.

It is this quality that makes diffuse light *viewpoint independent*. Of course the amount of light that strikes the surface depends on the incident light angle. If most of the light striking a surface is reflected diffusely, the surface will have a matte appearance (Figure 10-2).



Figure 10-2. Light re-irradiated in the diffusion phenomenon.

Since version 2.28, Blender has implemented three different mathematical formulae to compute diffusion. And, even more notably, the diffusion and specular phenomena, which are usually bound in a single type of material, have been separated so that it is possible to select diffusion and specular reflection implementation separately.

The three Diffusion implementations, or *shaders*, use two or more parameters each. The first two parameters are shared by all Diffuse Shaders and are the *Diffuse color*, or simply *color*, of the material, and the amount of incident light energy that is actually diffused. This latter quantity, given in a [0,1] range, is actually called Refl in the interface.

The implemented shaders are:

- *Lambert* This was Blender's default diffuse shader up to version 2.27. As such, all old tutorials refer to this, and all pre-2.28 images were created with this. This shader has only the default parameters.
- Oren-Nayar This shader was first introduced in Blender 2.28. It takes a somewhat
  more 'physical' approach to the diffusion phenomena because, besides the two
  default parameters, it has a third one which is used to determine the amount of
  microscopical roughness of the surface.
- *Toon* This shader was first introduced in Blender 2.28. It is a very 'un-physical' shader in that it is not meant to fake reality but to produce 'toonish' rendering, with clear light-shadow boundaries and uniformly lit/shadowed regions. Even though it is relatively simple, it still requires two more parameters which define the size of the lit area and the sharpness of the shadow boundaries.

A subsequent section, devoted to the actual implementation of the material, will analyze all these and their relative settings.

## **Specular Reflection**

#### Relevant to Blender v2.31

Unlike Diffusion, Specular reflection is *viewpoint dependent*. According to Snell's Law, light striking a specular surface will be reflected at an angle which mirrors the incident light angle, which makes the viewing angle very important. Specular reflection forms tight, bright highlights, making the surface appear glossy (Figure 10-3).



Figure 10-3. Specular Reflection.

In reality, Diffusion and Specular reflection are generated by exactly the same process of light scattering. Diffusion is dominant from a surface which has so much smallscale roughness in the surface, with respect to wavelength, that light is reflected in many different directions from each tiny bit of the surface, with tiny changes in surface angle.

Specular reflection, on the other hand, dominates on a surface which is smooth, with respect to wavelength. This implies that the scattered rays from each point of the surface are directed almost in the same direction, rather than being diffusely scattered. It's just a matter of the scale of the detail. If the surface roughness is much smaller than the wavelength of the incident light it appears flat and acts as a mirror.

**Note:** It is important to stress that the Specular reflection phenomenon discussed here is not the reflection we would see in a mirror, but rather the light highlights we would see on a glossy surface. To obtain true mirror-like reflections you would need to use a raytracer. Blender is not a raytracer as such, but it can produce convincing mirror-like surfaces via careful application of textures, as will be shown later on.

Like Diffusion, Specular reflection has a number of different implementations, or *specular shaders*. Again, each of these implementations shares two common parameters: the *Specular colour* and the energy of the specularity, in the [0-2] range. This effectively allows more energy to be shed as specular reflection as there is incident energy. As a result, a material has at least two different colors, a diffuse, and a specular one. The specular color is normally set to pure white, but it can be set to different values to obtain interesting effects.

The four specular shaders are:

- *CookTorr* This was Blender's only Specular Shader up to version 2.27. Indeed, up to that version it was not possible to separately set diffuse and specular shaders and there was just one plain material implementation. Besides the two standard parameters this shader uses a third, *hardness*, which regulates the width of the specular highlights. The lower the hardness, the wider the highlights.
- *Phong* This is a different mathematical algorithm, used to compute specular highlights. It is not very different from CookTorr, and it is governed by the same three parameters.
- *Blinn* This is a more 'physical' specular shader, thought to match the Oren-Nayar diffuse one. It is more physical because it adds a fourth parameter, an *index of re-fraction (IOR)*, to the aforementioned three. This parameter is not actually used to compute refraction of rays (a ray-tracer is needed for that), but to correctly compute

specular reflection intensity and extension via Snell's Law. Hardness and Specular parameters give additional degrees of freedom.

• *Toon* - This specular shader matches the Toon diffuse one. It is designed to produce the sharp, uniform highlights of toons. It has no hardness but rather a Size and Smooth pair of parameters which dictate the extension and sharpness of the specular highlights.

Thanks to this flexible implementation, which keeps separate the diffuse and specular reflection phenomena, Blender allows us to easily control how much of the incident light striking a point on a surface is diffusely scattered, how much is reflected as specularity, and how much is absorbed. This, in turn, determines in what directions (and in what amounts) the light is reflected from a given light source; that is, from what sources (and in what amounts) the light is reflected toward a given point on the viewing plane.

It is very important to remember that the material color is just one element in the rendering process. The color is actually the product of the light color and the material color.

## Materials in practice

Relevant to Blender v2.31

In this section we look at how to set up the various material parameters in Blender, and what you should expect as a result.

	DD NEW	
\$	Add New	•
ME	E:Cube	OB

Figure 10-4. Add new material.

Once an Object is selected, by pressing the **F5** key or **O**, you switch to Shading context and the Material Buttons window appears. This window will appear terribly empty, unless the Object already has a material linked to it. If there is no linked material, add a new one with the menu button (Figure 10-4).

Once you have added a material the buttons will appear as shown in Figure 10-5. Four panels are present, left to right: a Preview panel, a Material panel, a Shader panel and a Texture panel. We will concentrate on the first three, for now.

▼ Preview		<ul> <li>Material</li> </ul>	Shaders
		⇒ MA:Material X 🔂 F	Lambert = Ref 0.80
	<b></b>	ME:Cube OB ME 1 Mat 1	
		VCol Light VCol Pair TexFac Shadeless	CookTor + Spec 0.50
		Col R 0.756	Hard:50   .
		Spe B 0.756	
		Alpha 1.000	Amb 0.5
	4	RGB HS DYN SpecTra 0.0	Add 0.0

Figure 10-5. Material Buttons.

The Preview panel shows the material preview. By default it shows a plane seen from the top, but it can be set to a sphere or a cube with the buttons on the right of the panel (Figure 10-6).



Figure 10-6. Material Preview, plane (left) sphere (middle) and cube (right).

### **Material Colors**

The Material Panel (Figure 10-7) allows, among other things, setting of the material colors.

VCol Light	VCol	Pain TexFac Shadeless
	Col	R 0.756
	Sno	G 0.756
	ohel	B 0.756
	Mir	
		Alpha 1.000
RGBHS	DYN	SpecTra 0.0

Figure 10-7. Material colors buttons.

Each material can exhibit up to three colors:

- *The basic material color*, or the Diffuse color, or, briefly the *Color* tout court (Col button in the interface) which is the color used by the diffuse shader.
- *The Specular color,* indicated by the Spe button in the interface, is the color used by the specular shader.
- *The Mirror color*, indicated by the Mir button in the interface, is the color used by special textures to fake mirror reflections. (You'll find more information on this in the Environment Mapping section).

The aforementioned buttons select the pertinent color, which is shown in preview immediately to the left of each button. The three sliders at the right allow you to change the values for the *active* color in both a RGB scheme and in a HSV scheme. You can select these schemes via the RGB and HSV buttons at the bottom.

The DYN button is used to set the Dynamic properties of the Object in the RealTime engine (which is outside the scope of this book), while the four buttons above relate to advanced *Vertex Paint* and *UV Texture*.

#### The Shaders

The Shader panel (Figure 10-8) displays two MenuButtons allowing you to select one diffuse shader (Figure 10-9) and one specular shader (Figure 10-10).

Lambert 🗢	Ref	0.80	
CookTor 🗢	Spec Hard:	0.50 50	

Figure 10-8. Material Shader buttons.

a	
<u> </u>	Diffuse Shader
	Toon
	Oren-Nayar
	Lambert
	Lambert 🗧 Ref 0.80 💻 🛏
l	CookTor + Spec 0.50
	Hard:50 💷 📖 🛶

Figure 10-9. Material Diffuse shaders.

Specular Shader	
Toon	
Blinn	10 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
Phong	
CookTorr	
CookTor = Spec 0.	50 =
Hard:50	

Figure 10-10. Material Specular shaders.

The two sliders on the side, valid for all shaders, determine the intensity of the Diffusion and Specular phenomena. The Ref slider has a 0 to 1 range whereas the Spec has a 0 to 2 range. Speaking in strictly physical terms, if *A* is the light energy impinging on the object, *Ref* times *A* is the energy diffused and *Spec* times *A* is the energy specularly reflected. To be physically correct this must be *Ref* + *Spec* < 1 or the object would radiate more energy than it receives. But this is CG, so don't be too strict on physics.

Depending on the chosen shader other sliders may be present, allowing you to set the various parameters discussed in the introduction.

For the sake of completeness, Figure 10-11 shows all possible combinations. Of course, since there are so many parameters, these are just a small sample.



Figure 10-11. Shader overview.

## **Tweaking Materials**

The remaining material buttons both in the Material and Shaders panels perform some interesting effects.

⇒ MA:Material X 😡 F	Lambert + Ref 0.800 Halo
ME:Cube OB ME < 1 Mat 1	Traceabl
VCol Light VCol Pain TexFace Shadeless	CookTor + Spec 0.50 Radio
Col R 0.756	Hard:50 Wire
Spe G 0.756	Env
Mir Alpha 1 000	Amb 0.5
RGB_HS_DYN_SpecTra 0.0	Add 0.0 Zoffs: 0.000 Zinvert

Figure 10-12. Additional material sliders.

Figure 10-12 shows some interesting sliders. Alpha governs the opacity of the material; 1 is fully opaque and 0 is fully transparent. SpecTra forces specularity highlights on transparent bodies to be opaque. Shadeless makes the material insensitive to its shading, giving it a uniformly diffuse color.

In the Shaders panel, the Emit slider gives, if non zero, an emitting property to the material. This property makes material visible even without lights and can be itself a source of light if the Radiosity engine is used. (Figure 10-13).



Figure 10-13. Regular material (left), material with Alpha < 1 (center) and material with Emit > 0 (right).

The remaining column of buttons (Figure 10-14) activates some special features. Top Halo Button makes the material an 'Halo' material, which will be described later on.

#### Chapter 10. Materials and Textures

By default the Traceable, Shadows and Radio are activated. The first allows the material to *cast* shadows, while the second allows the material to *receive* shadows; the third allows the material to be taken into account if a Radiosity rendering is performed.

▼ Shaders	
Lambert + Ref 0.800	Halo
	Traceabl
	Shadow
CookTor = Spec 0.50	Radio
Hard:50	Wire
	ZTransp
	Env
Amb 0.5	OnlySha
	No Mist
Add 0.0	Zinvert

Figure 10-14. Material special buttons.

Wire renders the Object as a wireframe. ZTransp is necessary to activate the Alpha transparency effect.

The other buttons are not used that often and are described in the reference section at the end of the book.

# Ramp Shaders (-)

TO BE WRITTEN

# **Raytracing Reflections (-)**

TO BE WRITTEN

# **Raytraced Transparencies (-)**

TO BE WRITTEN

## **Multiple Materials**

#### *Relevant to Blender v2.31*

Most objects are assembled so that they can be modelled in parts, with each part composed of a different material. But on some occasions it may be useful to have an object modelled as a single Mesh, yet exhibiting different materials.

Consider the mushroom image shown in Figure 10-15. This object is a single mesh to which we need to assign two materials: one for the stem and one for the cap. Here's how to do it.



Figure 10-15. Mushroom Mesh



Figure 10-16. Mushroom with one material

1. Create a creamy stem material of your choice, and assign it to the entire mushroom. (Figure 10-16).



Figure 10-17. Mushroom with cap vertices selected

2. In a 3DWindow enter EditMode for the mushroom and select all the vertices belonging to the cap (Figure 10-17).

3. Go to the Link and Material Panel in the Mesh Edit Buttons (F9) and press New (Figure 10-18).

Stem		
4 2 Mat: 2 → ?		
New Delete		
Select Deselect		
Assign		
Set Smooth	Set Solid	

Figure 10-18. Adding a new material to the mesh

4. The mesh should now have two materials. The label should now read 2 Mat: 2 meaning that material number 2 out of 2 is active. The selected faces are assigned to this new material once you press the Assign button; the unselected faces keep any previous material assignment.

To see which faces belong to which material use the Select and Deselect buttons. Switch among materials with the Mat: NumButton. You can have up to 16 materials per mesh.

5. At any rate, both mesh materials are instances of the same material! So, keeping the material you want to change active, switch to the Material Buttons (F5) where you will find a similar "2 Mat 2" button. The material now has two users, as indicated by the blue color in the name of the material, and the number button showing "2" (Figure 10-19).

⇒ MA:Stem		2 🗙	∯ F	**
ME:Cube	OB	ME 🛛	2 Ma	t 2 💿 🕨

Figure 10-19. Multiple user material
Click on the "2" and confirm the OK? Single user question. You have now duplicated the material. The original material is still called "Stem" and the duplicate is "Stem.001". Rename the duplicate to "Cap". You can now edit the material as needed to obtain a nice looking cap. (Figure 10-20).



Figure 10-20. Mushroom with two materials.

**Textures:** If your material uses textures they remain linked even after you make the material single-user. To unlink textures, so that you can edit the two material textures separately, go to the TextureButtons for that material, and make the texture single-user as well.

# **Special Materials**

Relevant to Blender v2.31

Blender provides a set of materials which do not obey the shader paradigm and which are applied on a per-vertex rather than on a per-face basis.

## **Halo Materials**

Press F5 to display the Material buttons, and then press the Halo button on the Shaders Panel. The Panels change as shown in Figure 10-21.

Halo	🔻 Material			
Traceable		2 🗙 🔂 F 📪 🐨		Halo
Radio	ME:Cube	OB ME 2 Mat 2	Halo Size: 0.50       Halo Size: 0.50	Flare
Wire			Add 0 000	Rings
2 Iransp	Halo	R 0.756		Lines
Env		G 0 756	Rings: 4 > Kines: 12 >	Star
OnlyShad	Line	D 0 750	🔹 Star: 4 🔺 🔍 Seed: 0 🔺	HaloTex
No Mist	Bing	B 0.756		HaloPuno
Zinvert		Alpha 1.000		X Alpha
	RGB HSV DVN	SpecTra 0.000		Shaded

Figure 10-21. Halo buttons

As you can see, the Mesh faces are no longer rendered; instead a 'halo' is rendered at each vertex. This is most useful for particle systems because they generate free vertices, but it can also come in very handy when creating certain special effects, when making an object glow, or when creating a viewable light source.

As you can see in the three colors which, in standard material were Diffuse, Specular and Mirror colors are now relative to three different characteristics: the color of the halo itself, the color of any possible ring and the color of any possible line you might want to add with the relevant toggle buttons in Figure 10-21.



Figure 10-22. Halo results

Figure 10-22 shows the result of applying a halo material to a single vertex mesh. The halo size, alpha, and hardness can be adjusted with the pertinent sliders in Figure 10-21. The Add sliders determine how much the halo colors are 'added to', rather than mixed with, the colors of the objects behind and together with other halos.

To set the number of rings, lines, and star points independently, once they are enabled with the relative Toggle Button, use the Num Buttons Rings:, Lines: and Star:. Rings and lines are randomly placed and oriented, to change their pattern you can change the Seed: Num Button which sets the random numbers generator seed.

Let's use a halomaterial to create a dotmatrix display.

1. To begin, add a grid with the dimensions 32x16. Then add a camera and adjust your scene so that you have a nice view of the billboard.

2. Use a 2D image program to create some red text on a black background, using a simple and bold font. Figure 10-23 shows an image 512 pixels wide by 64 pixels high, with some black space at both sides.



#### Figure 10-23. Dot matrix image texture.

3. Add a material for the billboard, and set it to the type Halo. Set the HaloSize to 0.06 and when you render the scene you should see a grid of white spots.

4. Add a Texture, then change to the Texture Buttons and make it an image texture. When you load your picture and render again you should see some red tinted dots in the grid.

5. Return to the Material Buttons and adjust the sizeX parameter to about 0.5 then render again; the text should now be centered on the Billboard.

6. To remove the white dots, adjust the material color to a dark red and render. You should now have only red dots, but the billboard is still too dark. To fix this enter EditMode for the board and copy all vertices using the **SHIFT-D** shortcut. Then adjust the brightness with the Add value in the MaterialButtons.



Figure 10-24. Dot Matrix display

You can now animate the texture to move over the billboard, using the ofsX value in the Texture panel of the MaterialButtons. (You could use a higher resolution for the grid, but if you do you will have to adjust the size of the halos by shrinking them, or they will overlap. (Figure 10-24).

**Halo Texturing:** By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo.

To have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies press the HaloTex button; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic rain effect using particle systems, or similar.

#### Lens Flares

Our eyes have been trained to believe that an image is real if it shows artifacts that result from the mechanical process of photography. *Motion blur, Depth of Field,* and *lens flares* are just three examples of these artifacts. The first two are discussed in the Chapter 17; the latter can be produced with special halos.

A simulated lens flare tells the viewer that the image was created with a camera, which makes the viewer think that it is authentic. We create lens flares in Blender from a mesh object using first the Halo button and then the Flare options in the Shaders Panel of the material settings. Try turning on Rings and Lines, but keep the colors for these settings fairly subtle. Play with the Flares: number and Fl.seed: settings until you arrive at something that is pleasing to the eye. You might need to play with FlareBoost: for a stronger effect (Figure 10-25). (This tool does not simulate the physics of photons travelling through a glass lens; it's just a eye candy.)

## Chapter 10. Materials and Textures

<ul> <li>Shaders</li> </ul>	
	Halo
✓ HaloSize: 0.50 →	Flare
Hard 50	
Add 0.000	Rings
	Lines
🔹 Rings: 4 🔺 🔍 Lines: 12 🔺	Star
🔹 Star: 4 🔺 🔍 Seed: 0 🔺	HaloTex
lareSize: 1.00 Sub Size: 1.00	HaloPun
Boost: 1.000	X Alpha
Fl.seed: 6 Flares: 1	Shaded

Figure 10-25. Lens Flare settings

Blender's lens flare looks nice in motion, and disappears when another object occludes the flare mesh. (Figure 10-26).



Figure 10-26. Lens Flare

# **Chapter 11. Textures**

Textures are...

# **Textures**

Relevant to Blender v2.31 MISSING AL NEW STUFF!

The material settings that we've seen up to now produce nice, smooth, *uniform* objects. Of course, such objects are never true to reality, where disuniformities are most common.

Blender accounts for these disuniformities, whether in color, reflective or specular power, roughness, and so on, via *textures*.

# **Textures from the Material Point of View**

In Blender, the Materials and Textures form separate blocks in order to keep the interface simple and to allow universal integration between Textures, Lamps, and World blocks.

The relationship between a Material and a Texture, called the 'mapping,' is two-sided. First, the information that is passed on to the Texture must be specified. Then the effect of the Texture on the Material is specified. The Texture panel on the right-hand side (and similar panels exist for the the Lamp and World buttons) defines all these calculations.

For an untextured material the panel shows a column of eight empty texture *channels* (Figure 11-1), by selecting one and pressing Add New or by selecting an existing texture with the MenuButton right below (Figure 11-2) you add a texture and the Panel shows two more tabs: Map Input and Map To. The Tabs buttons are organized in the sequence in which the 'texture pipeline' is performed.

<b>~</b>	Clouds

Figure 11-1. Texture Channels

Each channel has its own individual mapping. By default, textures are executed one after another and then superimposed. As a result, an added second Texture channel can completely replace the first one! Next to each non-empty texture channel a check button allows you to select or de-select a given channel. De-selected channels are simply removed from the pipeline.

TE:	:Clouds		
\$	Clear	1	<b>.</b>

#### **Figure 11-2. Texture selection block**

The Texture itself is designated by its name, which you can edit in the Text Button above the Texture selection MenuButton.

UV	Objec	t			
Glob	Orco	Stick	Win	Nor	Refl

Figure 11-3. Material Coordinate input

Figure 11-3 shows the Map Input panel. Each Texture has a 3D coordinate (the texture coordinate) as input. The values, passed to the texture as coordinates for each pixel of the rendered image belonging to a given material, are computed according to these buttons:

- UV Uses a special kind of mapping called 'UV' mapping. This is especially useful when using images as textures, as we'll see in the Section called *UV Editor and FaceSelect*.
- Object Uses an Object as source of coordinates; usually an Empty. The Object name must be specified in the text button on the right. This is the preferred way to place a small image as a logo or whatever at a given point on the object.
- Glob Uses Blender Global 3D coordinates.
- Orco Uses the Object local, original, coordinates.
- Stick Uses the Object local, sticky, coordinates.
- Win Uses the rendered image window coordinates.
- Nor Uses the direction of the normal vector as coordinates.
- Refl Uses the direction of the reflection vector as coordinates.

Flat	Cube
Tube	Sphe

Figure 11-4. Texture mapping

If the texture is an image it is 2D, and we must map the 3D space onto it. The most flexible way to do so is with UV mapping, otherwise four possible pre-set mappings are provided (Figure 11-4).

Х	Υ	Ζ
Χ	Υ	Ζ
X	Υ	Ζ

Figure 11-5. Coordinate transformation

The X, Y and Z coordinates passed to the texture can be shuffled about to obtain special effects. The buttons in Figure 11-5 allow you to switch X into Y or Z and so on, or to turn them off.

-	ofsX 0.000	- (b)
۰	ofsY 0.000	
-	ofsZ 0.000	
۰	sizeX 1.00	- )+-
-	sizeY 1.00	
4	size7 1 00	b.

Figure 11-6. Texture coordinate Offsets and Scaling factors

Coordinates can be scaled and translated by assigning an offset (Figure 11-6).

Stenci Neg No RGB			
R 0.780 🔳			
G 0.197 ■F			
B 0.144 ■⊨			
DVar 1.00	]		

**Figure 11-7. Texture Inputs** 

Moving to the Map To tab, Figure 11-7 shows the texture input settings. The three buttons determine whether the texture should be used as a Stencil (a Mask for subsequent texture channels); a Negative texture (assigning negative, rather than positive, values); or as a black and white (No RGB), intensity only, texture. The three sliders below these buttons define the texture base color, which can be overridden by color specifications inside the texture definition. Finally the last slider defines the intensity of the texture effect.

Col	Nor	C:	sp	Cmir	Ref
Spec	Har	d	A	lpha	Emit

Figure 11-8. Texture Outputs

Figure 11-8 shows the toggle buttons which determine which characteristic of the material will be affected by the texture. Some of these button are three state buttons, meaning that the texture can be applied as positive or negative. All of these buttons are independent.

- Col (on/off) Uses the texture to alter the Material color.
- Nor (off/positive/negative) Uses the texture to alter the direction of the local normal. This is used to fake surface imperfections or unevenness via bump mapping.
- Csp (on/off) Uses the texture to alter the Specular color.
- Cmir (on/off) Uses the texture to alter the Mirror color.
- Ref, Spec, Hard, Alpha, Emit (off/positive/negative) Uses the texture to alter the Corresponding Material value.

#### Chapter 11. Textures

Mix Mul	Add Sub
Col 1.000	
Nor 0.500	
Var 1.000	

Figure 11-9. Output settings

The output settings (Figure 11-9) determine the strength of the effect of the Texture output. Mixing is possible with a standard value, including addition, subtraction, or multiplication. Textures give three types of output:

- RGB textures: return three values, which always affect color.
- Bump textures: return three values, which always affect the normal vector. Only the "Stucci" and "Image" texture can give normals.
- Intensity textures: return a single value. This intensity can control "Alpha," for example, or determine the strength of a color specified using the *mapping* buttons.

You can adjust the intensity of these settings separately using the pertinent sliders (Figure 11-9).

## **Textures themselves**

Once a new texture has been added to a material, it can be defined by switching to the Texture Buttons (**F6**) or sub-context of the Shading context to obtain Figure 11-10.

Preview	Texture	Colors	
Mat	TE:Clouds	X 🔂 F	
World	Clouds	None	
Lamp			
		Image	EnvMap
		Clouds	Marble
		Stucci	Wood
		Magic	Blend
Default Vars		Noise	Plugin

**Figure 11-10. Texture buttons** 

A new, empty texture Button Window presents two panels: a Texture Preview and a Texture panel, the latter with two tabs.

In the Preview panel toggle buttons define if this is a Material, Lamp or World texture, and a button Default Var allows to return texture parameters to default values.

The Texture tab replicates the texture channels and the Texture Menu Button of the linked Material. The two columns of Toggle Buttons selects the Texture type. The button Image allows an image to be loaded and used as a texture (the first button simply is "no texture"). The third button allows for the use of a very special kind of texture, the Environment Map (EnvMap). The last button (Plugin) allows for loading an external piece of code to define the texture. (These three buttons are rather unique and will be treated separately later on.) As soon as a texture type is chosen a new Panel appears, with a name matching the texture type, where texture parameters can be set.

The remaining buttons define 3D *procedural* textures, which are textures that are defined mathematically. They are generally simpler to use, and will give outstanding results once mastered. We will describe just one of these, the Wood button, leaving you to investigate further. (The reference chapter in this book contains a full details on each.)

wood is a *procedural*, which means that each 3D coordinate can be translated directly into a color or a value. These types of textures are 'real' 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two.

Procedural textures are *not* filtered or anti-aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

Procedural texture can either produce colored textures or intensity only textures. If intensity only ones are used the result is a black and white texture, which can be greately enhanced by the use of colorbands. The colorband is an often-neglected tool in the Colors tab in the Texture Panel that gives you an impressive level of control over how procedural textures are rendered. Instead of simply rendering each texture as a linear progression from 0.0 to 1.0, you can use the colorband to create a gradient which progresses through as many variations of color and transparency (alpha) as you like (Figure 11-11).

Colorband	Add	4	Cur: 1	•	Del
≪Pos 0.500≻ E	LS		A 1.000	-	1
R 0.500	G 1.00		I B O.	500	

#### Figure 11-11. Texture Colorband.

Skilled use of colorbands leads to really cool marble and cloud textures. To use it, select a procedural texture, such as Wood. Click the Colorband button.

The Colorband is Blender's gradient editor. Each point on the band can be placed at any location and can be assigned any color and transparency. Blender will interpolate the values from one point to the next. To use it, select the point you want to edit with the Cur: number button, then add and delete points with the Add and Del buttons. The RGB and Alpha values of the current point are displayed, along with the point's location on the band. Drag with the left mouse to change the location of the current point.

We can use two wood textures to make ring patterns in two different scales, each of which will have different effects on the appearance of the wood. The wood textures are identical except for the way in which they are mapped in the material buttons window, and in the different color bands used.

We will also also use a Clouds texture to make a grain pattern. To see the result of just one texture, isolated from the others, remember the Check buttons in the Texture Panel in Material Buttons.

<b>*</b>	¥		
TE:CI	louds		
\$	Clear	1	₩

Figure 11-12. Copying and Pasting Textures

**Copying texture settings:** By adding an existing texture you link that texture, but all the Material mapping parameters remains as they are. To copy all texture settings, inclusive of mappings, you must copy a given texture channel and paste it into another by using the two arrow buttons in Figure 11-12.

Figure 11-13, Figure 11-14 and Figure 11-15 show the three individual textures which, when combined in a single material and mapped to various material parameters, create a nice wood texture (Figure 11-16).



Figure 11-13. First Wood ring texture



Figure 11-14. Second Wood ring texture



Figure 11-15. Clouds texture



Figure 11-16. Final result

# ImageTexture

The Image texture is the only true 2D texture, and is the most frequently used and most advanced of Blender's textures. The standard, built-in bump mapping and perspective-corrected mip-mapping, filtering, and anti-aliasing guarantee outstanding images (set DisplayButtons OSA to ON for this). Because pictures are two-dimensional, the way in which the 3D texture coordinate is translated to 2D must be specified in the *mapping* buttons (Figure 11-4).

The four standard mappings are: Flat, Cube, Tube and Sphere. Depending on the overall shape of the object, one of these types is more useful.



## Figure 11-17. Flat Mapping.

The Flat mapping (Figure 11-17) gives the best results on single planar faces. It does produce interesting effects on the sphere, but compared to a sphere-mapped sphere the result looks flat. On faces not in the mapping plane the last pixel of the texture is repeated, which produces stripes on the cube and cylinder.



## Figure 11-18. Cube Mapping.

The cube-mapping (Figure 11-18) often gives the most useful results when the objects are not too curvy and organic (notice the seams on the sphere).



## Figure 11-19. Tube Mapping.

The tube-mapping (Figure 11-19) maps the texture around an object like a label on a bottle. The texture is therefore more stretched on the cylinder. This mapping is of course very good for making the label on a bottle or assigning stickers to rounded objects. However, this is not a cylindrical mapping so the ends of the cylinder are undefined.



#### Figure 11-20. Sphere Mapping.

The sphere-mapping (Figure 11-20) is the best type for mapping a sphere, and it is perfect for making a planet and similar stuff. It is often very useful for creating organic objects. It too produces interesting effects on a cylinder.

**Moving a texture:** As described in the previous section you can manipulate the texture in the texture part of the MaterialButtons. There is one more important feature to manipulate the textures.

When you select an object and press **TKEY**, you get the option to visually scale and move the texture space, but not to rotate the texture. The <code>Object</code> coordinate mapping is anyway much more flexible.

# **Environment Maps**

Relevant to Blender v2.31

The shiny surfaces that Blender generates show specular highlights. The ironic thing about these specular shaders, though, is that they are sensitive only to lamps. Specifically, specular shaders surfaces show you a bright spot as a mirror-like reflection of a lamp.

This all makes sense except that if you turn the camera directly toward the lamp you won't see it! The camera sees this light only if it is being reflected by a specular shader, not directly. On the other hand, objects that appear very bright in your scene (that reflect a lot of light to the camera) but are not lamps don't show up in these highlights.

It is easy enough to make a lamp which is directly visible to the camera by placing some renderable object in the scene which looks like some appropriate sort of lamp fixture, flame, sun, and so on. However, there is no immediate fix for the fact that surrounding objects do not show up on specular highlights.

In a word, we lack *reflections*. This is the sort of problem we will address using the technique of environment mapping.

Just as we render the light that reaches the viewing plane using the camera to define a viewpoint, we can render the light that reaches the surface of an object (and hence, the light that might ultimately be reflected to the camera).

Blender's environment mapping renders a cubic image map of the scene in the six cardinal directions from any point. When the six tiles of the image are mapped onto an object using the Refl input coordinates, they create the visual complexity that the eye expects to see from shiny reflections.

**Note:** It's useful to remember here that the true goal of this technique is *believability*, not *accuracy*. The eye doesn't need a physically accurate simulation of the light's travel; it just needs to be lulled into believing that the scene is real by seeing the complexity it expects. The most unbelievable thing about most rendered images is the sterility, not the inaccuracy.

The first step to follow when creating an environment map is to define the viewpoint for the map. To begin, add an empty to the scene and place it *in the specular position of the camera with respect to the reflecting surface*. (This is possible, strictly speaking, only for planar reflecting surfaces.)

Ideally, the location of the empty would mirror the location of the camera across the plane of the polygon onto which it is being mapped. It would be ridiculously difficult to create a separate environment map for every polygon of a detailed mesh, so we take advantage of the fact that the human eye is very gullible.

In particular, for relatively small and complex objects, we can get away with simply placing the empty near the center. We name the empty env so that we can refer to it by name in the environment map settings.

We will create a reflective sphere over a reflective plane, using the set up depicted in Figure 11-21.



Figure 11-21. Environment Map utilization example

Note the 'env' Empty is placed exactly below the camera, at a distance from the reflecting plane equal to 3 blender units, which is equal to the height of the camera over the same plane.

Now, let's place some lights, leave the sphere without a given material, and move the plane to a *different layer*. For example, say that everything is on layer 1, except for the plane which is in layer 2.

Give the plane a low Ref and Spec material and add a texture on channel two with the parameters in Figure 11-22.

Material     MA-Material     MA-Material     ME/Plane     OB     ME     1 Mat     Sel     Col light VCOPail TexFac     Shadeless     Col light 000     Indiana     Sel     Go 800     Sel     Go 800     Sel     Sel	Shaders     Identified Ref 0.20     Halo     Traceabl     Shadow     CookTor s     Spec 0.20     Hard:50     Wire     ZTransp     Fny	Texture Map Input Map To     env     env     Clear 1
B 0.800	Amb 0.5 Emit 0.1 OnlySh No Mist Add 0.0 Zoffs: 0.000 Zinvert V Texture Map Input Map To UV Object Glob Orco Stick Win Nor Refl	Texture Map Input Map To     UV Object     Glob Orco Stick Win Nor Ref
	Flat         Cube         ofsX 0.000         ofsY 0.000           Tube         Sphe         ofsY 0.000         ofsY 0.000           ofsY 0.000         ofsZ 0.000         ofsZ 0.000         ofsZ 0.000           X         Y         Z         sizeX 1.00         ofsZ 0.000           X         Y         Z         sizeY 1.00         ofsZ 0.000           X         Y         Z         sizeZ 1.00         ofsZ 0.000	Flat         Cube         ofSX 0.000 b           Tube         Sphe         6 ofSY 0.000 b           K         Y         Z           K         Y         Z           K         Y         Z           K         Y         Z           K         Y         Z           K         Y         Z           K         Y         Z           SizeY 1.00 b         S           X         Y           K         SizeY 1.00 b

Figure 11-22. Reflecting plane material.

Note both the Refl mapping and the Cmir effect. We use channel 2 and not 1 because we will need channel 1 later on in this example.

Material Shaders			
SMA:Material X G F      CU:SurfSphere OB CU ← 1 Mat 1 →	V Tex Y Y	UV Object Glob Orco Stick Win Nor Ref	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
Col R 0.000	V Tex V Creat & Get	Tube Sphe ofsZ 0.000	R 1.000
Alpha 1.000		X         Y         Z         sizeX 1.00         sizeY 1.00           X         Y         Z         sizeY 1.00         sizeY 1.00           X         Y         Z         sizeY 1.00         sizeY 1.00	G 1.000 B 1.000 DVar 1.0 Var 0.500 Var 0.500

Figure 11-23. Reflecting plane EnvMap settings.

#### Chapter 11. Textures

Now define the newly assigned texture as an EnvMap in the Texture Buttons (F6) (Figure 11-23). In the Envmap Panel, note the OD: field containing the name of the Empty with respect to which we compute the EnvMap. Note also the resolution of the cube on which the EnvMap will be computed and, most important, the Don't render layer: buttons.

Because the EnvMap is computed from the Empty location it must have an unobstructed view of the scene. Since the reflecting plane would completely hide the sphere, it must be on its own layer which must be marked as 'Not renderable' for the EnvMap calculation.

Pressing **F12** starts the rendering process. First, six different square images comprising the EnvMap are computed, after which the final image is produced, of the sphere reflected over the plane.



Figure 11-24. Sphere on a reflecting surface.

To add more visual appeal to the scene, add a big sphere encompassing the whole scene and map a sky image onto it to fake a real, cloudy world. Then add a new Empty in the center of the Sphere and move the Sphere to Layer 3. Next, give the sphere an EnvMap exactly as you did for the plane (but this time layer 3 must not be rendered!)

Now add some cylinders, to make the environment even more interesting, and, before pressing F12 return to the plane's texture and press the Free Data button. This will force Blender to recalculate the EnvMap for the new, different, environment.

This time in the rendering process twelve images, six for each EnvMap, will be computed. The result is in Figure 11-25. The sphere is shiner than the plane due to slightly different settings in the materials.



Figure 11-25. Reflecting sphere on a reflecting surface.

But wait, there is a problem! The Sphere reflects the Plane, but the Plane reflects a dull grey Sphere! This is because the Plane EnvMap is computed *before* the sphere EnvMap. As such, when it is computed the sphere is still dull grey, while when the Sphere EnvMap is computed the plane already has its Reflection.

To fix this locate the Depth Num Button in the Envmap panel of the Texture buttons and set it to 1 both for the plane and the sphere EnvMap texture. This force recursive computation of EnvMaps. Each EnvMap is computed, then they are recomputed as many times as 'Depth' is set to, always one after the other. The result is in to fix this Figure 11-26.



Figure 11-26. Reflecting sphere on a reflecting surface with multiple reflections.

Now, if you are still wondering why the first texture channel of the Plane material was kept empty... Add a new texture to the first channel of the plane material. Make it Glob, affecting the Nor with a 0.25 intensity (Figure 11-27).

	▼ Map Input	🔻 Мар То
Waves env Env Clear 1 &	UV         Object           Glob         Orco         Stick         Win         Nor         Refl           Flat         Cube         ofsX 0.000 +         ofsX 0.000 +         ofsX 0.000 +           Tube         Sphe         ofsZ 0.000 +         ofsZ 0.000 +         ofsZ 0.000 +         ofsZ 0.000 +           X         Y         Z         sizeX 1.00 +         sizeX 1.00 +         sizeX 1.00 +           X         Y         Z         sizeX 1.00 +         sizeX 1.00 +         sizeX 1.00 +	Col         Nor         Csp         Cmir         Ref           Spec         Hard         Alpha         Emit           Stenc[Neg] No RGB         Mix         Mul] (Add] Sub           R 1.000         I         I           B 1.000         I         Nor 0.250           Dvar 1.00         I         Var 1.000

Figure 11-27. Additional texture set-up for BumpMapping.

This new texture should be of Stucci type; tune the Noise Size down to 0.15 or so. If you now render the image the plane will look like rippled water (Figure 11-28).



Figure 11-28. Reflecting sphere on a reflecting water with multiple reflections.

You must have the BumpMap on a channel preceding the EnvMap because textures are applied in sequence. If you were to do this the other way around the reflection would appear to be broken by waves.

You can save EnvMaps for later use and load them with the pertinent buttons in the Texture Buttons. You can also build your own envmap. The standard is to place the six images mapped on the cube on two rows of three images each, as in Figure 11-29.



Figure 11-29. The EnvMap as it is stored.

Blender allows three types of environment maps, as you can see in Figure 11-23:

- Static The map is only calculated once during an animation or after loading a file.
- Anim The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.
- Load When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps.

**Note:** You can animate the water of the previous example by setting an IPO for the ofsx and ofsy values of the texture placement in the Material Buttons. Rendering the animation would then show moving ripples on the surface, with reflections changing accordingly!

**Note:** The EnvMap of the Plane needs to be computed only once at the beginning if nothing else moves! Hence it can be static. The Envmap on the sphere is another matter, since it won't reflect the changes in the reflections in the water unless it is computed at each frame of the animation. Hence it should be of type Anim.

If the camera is the only moving object and you have a reflecting plane, the Empty must move too and you must use Anim EnvMaps. If the object is small and the Empty is in its center, the EnvMap can be Static, even if the object itself rotates since the Empty does not move. If, on the other hand, the Object translates the Empty should follow it and the EnvMap be of Anim type.

Other settings are:

- Filter: With this value you can adjust the sharpness or blurriness of the reflection.
- Clipsta, ClipEnd These values define the clipping boundaries when rendering the environment map images.

**Note:** EnvMap calculation can be disabled at a global level by the EnvMap Tog Button in the Render Panel of the Rendering Buttons.

## **Displacement Maps**

by Kenneth Styrberg

#### Relevant to Blender v2.35

Displacement mapping is a powerful technique that allows a texture input, either procedural or image, to manipulate the position of rendered faces. The displacement is controlled like a NOR map, a brighter texture will have a higher displacement. Unlike Normal or Bump mapping, where the normals are skewed to give an illusion of a bump, this creates real bumps. They cast shadows, occlude other objects, and do everything real geometry can do.

Displacement mapping is set up to behave as a texture channel, with one very important difference, in order to manipulate the positions of renderfaces smoothly, the faces have to be very small and this eats memory and CPU time.

For distant/non-critical items, NOR mapping should still be used. NOR maps, compared to Displacement maps, imposes very little additional CPU cost per render-

#### Chapter 11. Textures

face, and you can NOR map independantly of renderface count. Use of Displacement maps quickly leads to million face scenes.

Here we see the difference using NOR maps versus Displacement Maps, Figure 11-30. The left object has 240 faces and the right has 60717 faces at rendertime!



Figure 11-30. Difference between NOR and Displacement Maps.

Note: Use Displacement Maps when you need your geometry to be more accurate.

#### **Displacement Maps on Objects**

Here is a list, from best to worst, that shows how different object types work with Displacement Maps.

- Subsurf Meshes (Catmull-Clark) size is controlled with the render subsurf level. Displacement will work great!
- **Simple Subsurf Meshes** Control renderfaces with render subsurf level. Displacement will work but there is a pitfall at sharp edges if the texture there is not neutral gray.
- **Manually subdivided meshes** Control renderfaces with number of subdivides. This can slow editing down because you can not turn down the subdivide level when editing dense meshes.
- **Metaballs** Control renderfaces with render wiresize. A small wiresize gives more faces.

The use of Displacement Maps on the following object types are possible but that they can give normal errors and visible seams when rendered.

The face count is directly connected to the U/V resolution of the surfaces. Higher resolution gives more renderfaces.

- Open Nurbs surfaces
- Closed Nurbs surfaces
- Curves and Text

**Note:** It is recommended that you convert curve and surface object types to meshes before applying displacement.

## Interface

The interface to Displacement Maps is two buttons and two value sliders in Materials context (F5), Map To panel. Figure 11-31

▼ Texture	Мар	Input		Мар То	
Col Nor C	sp Cn	nir f	Ref	Spec	Amb
Hard RayMir	Alpha	Emit	Т	ranslu	Disp
Stencil Neg No	RGB	Mix	_		\$
		Col 1.0	00		J
R 1.000		Nor 12.	50		1
G 0.000		Var 1.0	00		
B 1.000		Disp 0.5	500		
DVar 1.000	1				

Figure 11-31. Map To panel.

The intensity displacement is controlled with the  $\tt Disp$  slider and the normal displacement is controlled by the Nor slider.

#### **Displacement Map usage**

There are two modes in which displacement works in:

- Displace rendered vertices by intensity, vertices move along vertex normals.
- Displace rendered vertices by texture normal, vertices move according to texture's NOR input.

The two modes are not exclusive. The amount of each type can be mixed using the sliders in the Materials context (F5), Map To panel, Figure 11-31.

Not all textures provide both types of input though. Stucci, for example, only provides Normal, while Magic only provides Intensity. Cloud, Wood and Marble provide both Normal and Intensity. Image provides both Intensity and a derived Normal.

Note: Texture OSA is not currently working correct for images mapped to displacement.

Intensity displacement, gives a smoother, more continous surface, since the vertices are displaced only outward. Normal displace, gives a more aggregated surface, since the vertices are displaced in multiple directions.

Here is an example of Figure 11-30, (right object), but with a Nor slider setting of about **2.0**. Note that the Nor button is still unselected! You can clearly see the more aggregated displacement when Nor settings is used together with Disp.



#### Figure 11-32. Disp and Nor settings.

The depth of the displacement is scaled with an object's scale, but not with the relative size of the data. This means if you double the size of an object in Object mode, the depth of the displacement is also doubled, so the relative displacement appears the same.

If you scale in Editmode, the displacement depth is not changed, and thus the relative depth appears smaller or bigger.

The textures intensity defines the displacement. Neutral gray, RGB = 128,128,128, means no displacement. For positive displacement, Figure 11-33, white is a peak, black is a groove.

#### Disp

Figure 11-33. Positive Displacement selected.

For negative displacement, Figure 11-34, it is reversed.

#### Disp

Figure 11-34. Negative Displacement selected.

#### Example

Here is a example showing the effect the subdivision level has to the end result of diplacement maps. This is a Catmull-Clark subdivision type and the texture is a simple Cloud texture added to a Ico Sphere. See the Section called *Catmull-Clark Subdivision Surfaces* (-) in Chapter 7 for more on subdivision surfaces.

The subdivision levels range from 0 - (none) to 6 - (maximum) subdivision.



Figure 11-35. Subsurf level 0.

# Solid and Hollow Glass

#### Relevant to Blender v2.31

Glass and tranparent materials are generally tricky to render because they exhibit *refraction*; that is, the bending of light rays due to the different *optical density*, or *index of refraction* of the various materials. Unfortunately, to fully account for refraction a ray tracer is mandatory. Still, we can produce convincing results in Blender using EnvMaps and advanced Texturing techniques.

Consider a scene with some basic geometries, including a cube, a cone, a sphere, and a torus. As a first example we will make the sphere look like a solid ball of glass and, as a second example, that same sphere will become a glass bubble.

To create this effect, we need to make the light appear to bend as it passes through the sphere, since we would expect objects behind the solid glass sphere to appear heavily warped, as if through a very thick lens. On the other hand, the hollow glass sphere's center should be almost transparent while the sides should deflect light.

## Solid Glass

1. To begin, we set up an environment map for the sphere's material just as we did for the ball in the previous section, with an empty which locates the EnvMap's perspective at the center of the sphere.

2. To fake Refraction we'll tweak the output mapping with the ofsz, sizex, sizeY, sizeZ and Col sliders to warp the map in a way that creates the effect of refraction. To do so, use the settings in Figure 11-36.

Tex TETER Clear 1 &	UV         Object           Glob         Orco         Stick         Win         Nor         Refl           Flat         Cube         ofsX 0.000 +	Col         Nor         Csp         Cmir         Ref           Spec         Hard         Alpha         Emit           Stenc         Nor RGB         Mix         Mul         Add         Sub           R 1.000         I         Interview         Interview         Interview         Nor         Sub           B 1.000         Nor         0.500         Interview         Nor         Sub         Nor           DVar 1.00         Var 1.00         Var 1.00         Interview         Nor         Nor         Nor

Figure 11-36. Envmap settings to fake refraction.

3. Select the Mir RGB material sliders and lower the R and G a bit to give the texture a blue tint. (Our experience with the idiosyncrasies of Blender's handling of mirror colors dictates this unintuitive approach when combining environment-mapped reflections and refractions in a single material.)

4. Turn the Ref slider all the way down. (Figure 11-37). You should now have produced a blue-tinted refraction of the environment.



Figure 11-37. Material settings

5. Shiny glass also needs a reflection map, so we'll place the same texture into another texture channel. Press the Add, Col, and Emit buttons, and use the Refl button for the coordinates. Make the material Color black and turn Emit all the way up. (Figure 11-38).

Material Shaders			
MA:Material     X G F      Y     CU:SurfSphere OB CU < 1 Mat 1 →	Tex Y Y	UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
VCol Light VCol Pair TexFac Shadeless	v Tex	Fill         Cube         < ofsX 0.000            Tube         Sphe         < ofsY 0.000	Stend Ne No RGB Mix Mul Art Sub
Spe 8 0.000 B 0.010 Mir Alpha 1.000		X Y Z sizeY 1.00 = X Y Y Y Y Z sizeY 1.00 = X Y Y Y Y Y Y Y Z sizeY 1.00 = X Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y	G 1.000
RCE HS DYN SpecTra 0.0		X T Z 15122 1.00	DVar 1.0 Var 0.500

Figure 11-38. Reflection Map

6. This changes our first texture considerably. In order to return the refraction texture to a nice blue tint, we have to add a new texture, leaving the texture type set to None. Select the Mix and Cmir buttons, and set the Col slider about halfway up. Click the Neg button and set the texture input RGB sliders to a dark blue (Figure 11-39).

🔻 Мар То	
Col Nor C	Csp Cmir Ref
Spec Hard	Alpha Emit
Stenci Ne No RGB	Mix Mul Add Sub
R 0.000	
	Col 0.500
	Nor 0.500 II
DVar 1.0	Var 1.000

Figure 11-39. Final touches

The final result should look like Figure 11-40. The refraction effect is most noticeable when the scene is animated.



Figure 11-40. Rendering

# **Hollow Glass**

The procedure we've described above works fine for a solid lump of glass, but how do we produce the appearance of hollow glass, like a vase?

Thin glass has strong refraction only where it slopes away from the eye at a steep angle. We can easily mimic this effect by using Blender's Blend texture to control the object's transparency, as well as another transparency texture to keep the bright highlights visible.

1. Add a new texture to the material. Select Blend as the type and select the Sphere option.

2. Return to the material buttons. Select Nor as the mapping type, and disable the X and Y axes in the input coordinates.

3. Mix the texture with Alpha, then move the Alpha material slider to 0.0 and set the ZTransp option (Figure 11-41).

<ul> <li>Material</li> </ul>	<ul> <li>Shaders</li> </ul>	Texture Map Input	🔻 Мар То
	Lambert © Ref 0.00 - Halo	UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
VCol Light VCol Pair TexFac Shadeless	CookTor = Spec 0.00 - Radio Hard:255 Wire	Flat         Cube         < ofsX 0.000         >           Tube         Sphe         < ofsY 0.000	Stenci Ne No RGB
Spe         B 0.000           Mir         Alpha 0.156           SpeCia 0.0         B 0.000	Amb 0.5 Emt 1.0 OnlySh Add 0.0 Zoff: 0.000 Zinvert	X         Y         2         4         sizeX 1.00 >           X         Y         2         4         sizeY 1.00 >           X         Y         2         4         sizeY 1.00 >           X         Y         2         4         sizeY 1.00 >	Col 1.000

Figure 11-41. Setting transparency.

This produces the effect of nice transparency as the surface angles toward the eye, but we want the bright environment-mapped reflections to show up on those otherwise-transparent areas. For example, if you look at glass windows, you will see that bright light reflecting from the surface is visible, preventing you from seeing through a pane that would otherwise be transparent. We can produce this effect easily by selecting the environment-mapped reflection texture in the material window and enabling the Alpha option (Figure 11-42).

Tex Tex	UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
✓ Tex ✓ Tex ✓ Tex	Flat         Cube         ofsX 0.000           Tube         Sphe         ofsY 0.000           ofsZ 0.000         ofsZ 0.000	Stenci Ne No RGB Mix Mul Add Sub
	X         Y         Z           X         Y         Z           SizeY         1.00           X         Y           Z         sizeY           SizeZ         1.00	R 1.000         I           G 1.000         I           B 1.000         I           DVar 1.0         Var 0.500

Figure 11-42. Setting Reflections

That's all there is to it. The result should look like Figure 11-43.



Figure 11-43. Hollow Glass

# **UV Editor and FaceSelect**

Relevant to Blender 2.33

#### Introduction

UV mapping is a way to map 2D image textures to 3D models. It can be used to apply textures to arbitrary and complex shapes, like human heads or animals. Often these textures are painted images, created in applications like The Gimp, Photoshop, or your favorite painting application.

Procedurals as seen in the previous chapters are a nice way to texture a model. What is really nice about them is that they will always "fit". 2D images will not always fit on 3D shapes. Creating procedural textures is relatively easy, and they provide a quick way to get good results.

However, there are cases when this kind of textures is not good enough. For instance, the skin on a human head will never look quite right when procedurally generated. Wrinkles on a human head, or scratches on car do not occur in random places, but depend on the shape of the model and its usage. Manually painted images give the artist full control over the final result. Instead of playing with some numerical sliders, artists will be able to control every pixel on the surface. This means more work, but will lead to better images in the end.

A UV map is a way to assign a part of an image to a polygon in the model. Each polygon's vertex gets assigned to 2D coordinates that define which part of the image gets mapped. These 2D coordinates are called UVs (compare this to the XYZ coordinates in 3D). The operation of generating these UV maps is also called "unwrap", since it is as if the mesh were unfolded onto a 2D plane.

**Tip:** UV mapping is also essential in the Blender game engine, or any other game. It is the de facto standard for applying textures to models; almost any model you find in a game is UV mapped.

#### The UV Editor

UV Mapping is done in Blender within the UV Editor window and a special mode in the 3D Window called the Face Select Mode. The UV-Editor allows you to map textures directly onto the faces of meshes.

Each face can have individual texture coordinates and an individual image assigned to it, and can be combined with vertex colors to make the texture brighter or darker or to give it a color.

By using the UV-Editor each face of the Mesh is assigned two extra features:

- *four UV coordinates* These coordinates define the way an image or a texture is mapped onto the face. These are 2D coordinates, which is why they're called UV, to distinguish them from XYZ coordinates. These coordinates can be used for rendering or for realtime OpenGL display.
- *a link to an Image* Every face in Blender can have a link to a different image. The UV coordinates define how this image is mapped onto the face. This image then can be rendered or displayed in realtime.

A 3D window has to be in "Face Select" mode to be able to assign Images or change UV coordinates of the active Mesh Object.

Мо	de:
\$	Texture Paint
1	Vertex Paint
D	UV Face Select
	Edit Mode
K	Object Mode

Figure 11-44. Entering Face Select Mode.

First add a Mesh Object to your Scene, then enter the FaceSelect choosing the UV face select entry in the Mode menu.

Your Mesh will now be drawn Z-buffered. If you enter the Textured draw mode (**ALT-Z**, also called "potato mode") you will see your Mesh drawn in white, which indicates that there is currently no Image assigned to these faces. You can control the way the faces are drawn using the Draw Edges and Draw Faces buttons in the UV Calculation Panel. If Draw Edges is activated all faces will be drawn outlined. With Draw Faces activated, all selected faces will be shown in a light pink tone (or the theme colour).



Figure 11-45. Face Select Mode.



Figure 11-46. UV Calculation Panel.

Press **AKEY** and all faces of the Mesh will be selected and highlighted by dotted lines. You can select faces with **RMB**, or BorderSelect (**BKEY**) in the 3D window. If you have problems with selecting the desired faces, you can also enter EditMode and select the vertices you want. After leaving EditMode the faces defined by the selected vertices should be selected as well.

Only one face is active. Or in other words: the Image Window only displays the image of the active face. As usual within Blender, only the last selected face is active and selection is done with **RMB**.

Change one window into the UV Editor/Image Window with **SHIFT-F10**. Here you can load or browse an image with the Load button. Images have to be multiples of 64 pixels (64x64, 128x64 etc.) to be able to be drawn in realtime (note: most 3D cards don't support images larger than 256x256 pixels). However, Blender can render all assigned images regardless of size when creating stills or animations.



Figure 11-47. UV Editor.

Loading or browsing an image in FaceSelect automatically assigns the image to the selected faces. You can immediately see this in the 3D window (when in Textured view mode).

# **Unwrapping tools**

In the 3D window, you can press **UKEY** in FaceSelect mode to get a menu to calculate UV coordinates for the selected faces. You can also perform an unwrapping using the UV Calculation Panel in Edit Buttons. This panel also provides better control over the unwrapping process.

UY Calculation
Cube
Cylinder
Sphere
LSCM
Bounds to 1/8
Bounds to 1/4
Bounds to 1/2
Bounds to 1/1
Standard 1/8
Standard 1/4
Standard 1/2
Standard 1/1
From Window

Figure 11-48. UV pre-sets.

The available UV unwrapping algorithms are:

- *Cube* This determines cubical mapping.
- *Cylinder, Sphere* Cylindrical/spherical mapping, calculated from the center of the selected faces.
- *Bounds to 1/8, 1/4, 1/2, 1/1* UV Coordinates are calculated using the projection as displayed in the 3D window, then scaled to the given fraction of the image texture.
- *Standard 1/8, 1/4, 1/2, 1/1* Each face gets a set of default square UV coordinates which are then scaled to the requested fraction of the image texture.
- *From Window* UV coordinates are calculated using the projection as displayed in the 3D window.
- *LSCM* UV coordinates are calculated using the Least Squares Conforming Maps algorithm. Use it together with the marked seams.

In the UV mapping panel, you can tweak the way the mapping is done and how it is shown in the 3D window when the model is in Face Select Mode.

With View Aligns Face enabled, Cylindrical and Spherical unwrapping is affected by the view. The view is supposed to be in front of the Cylinder/Sphere, with the caps at the top and bottom of the view. The Cylinder/Sphere is *cut* at the opposite side of the view.

Size and Radius define the scaling of the map when using Cube or Spherical/Cylindrical mapping respectively.

With VA Top (View Aligns Top) enabled, the view must look through the Cylinder / Sphere. It is cut at the top of the view. With this activated you can also define how the view is rotated in respect to the poles using Polar ZX and Polar ZY options.

If Al Obj is enabled, the Cylinder/Sphere is rotated based on the Object's rotation.

Draw Edges and Draw Faces in the panel activate visualization of edges and faces in the 3D Window while in Face Select Mode. Selected faces in this mode will be drawn in a transparent purple (or the theme color), similar to Edit Mode. Drawing of Seams in Edit Mode and Face Select Mode can be toggled with Draw Seams. The colors of seams can also be changed in the Themes options.

# **Editing UV coordinates**

In the UV Editor you will see a representation of your selected faces as yellow or purple vertices connected with dotted lines. You can use the same techniques here as in the Mesh EditMode to select, move, rotate, scale, and so on. With the Lock button pressed you will also see realtime feedback in 3D of what you are doing. Scaling and Translating of vertices can be done in the local X or Y axis of the map if needed. Just press **XKEY** or **YKEY** after entering the scale command (**SKEY**). Proportional editing tool is also available and it works the same way that in Edit Mode for meshes. Vertices in the UV Editor can be hidden or shown using the **HKEY** and **ALT-H** respectively, the same way as in Edit Mode.

Grab/Move	G
Rotate	R
Scale	S
Weld/Align	W
Mirror	м

Figure 11-49. UV Transformation Menu.

There are several selection modes available in the UV Editor. Since a vertex is drawn in the Editor for each face it belongs, sometimes is hard to tell if we are selecting the same vertex or not.

With Stick UVs to Mesh Vertex enabled, a **RMB** click will not only select one UV vertex, but also all the UV vertices that belong to the same mesh vertex. You can use this mode even if it is not activated in the menu, by keeping **CTRL** pressed when selecting a vertex.

Stick Local UVs to Mesh Vertex works in the same way, but only select the UVs that are 'connected', meaning they are within a 5 pixel range of the first selected UV. You can also use this mode even if it is not set as default, by keeping **SHIFT** pressed when selecting a vertex.

These options are toggled on/off by respectively pressing CTRL-C and SHIFT-C.

With Active Face Select enabled, a **RMB** click will select a face, and make it the active face. This can be toggled on/off by pressing **CKEY**.

For all three of these options a special icon is displayed in the bottom left of the UV Editor. Note that Active Face Select and Stick UVs to Mesh Vertex can also be combined.

Unlink Selection will based on the current selection, only leave those UVs selected, of which the faces are fully selected. As the name implies, this is particularly useful to unlink faces and move them elsewhere. The hotkey is **ALT-L**.

Select Linked UVs works similar to Select Linked in the 3D View. It will select all UVs that are 'connected' to currently selected UVs. The difference with the 3D view is that in the UV Editor, UVs are connected 'implicitly'. Two UVs are considered selected if the distance between them is no longer than 5 pixels. The hotkey is **LKEY**.

Different parts of a UV map can be stitched if the border UV vertices correspond to the same mesh vertices by using the Stitch command (**VKEY**). The Stitch command works joining irregular outlines, just select the vertices at the border line using the "Stick UVs to Mesh Vertex".

Limit Stitch works in the same way. The difference is that it only snaps together UVs within a given range. The default limit is 20 pixels. Its advantage over 'Stitch' is that it prevents UVs, that are supposed to stay separate, from being stitched to-

gether. You can see on the screenshots how Limit Stitch prevents wraparounds when stitching together two parts of a Cylinder.



Figure 11-50. "Stitch" and "Limit Stitch".

You can merge UVs that do not correspond to the same mesh vertex by using the Weld command (**WKEY**). You can also use the Weld command to align in X or Y several vertices. After pressing **WKEY** press **XKEY** or **YKEY** to choose which axis you to align to

Some tips:

- Press **RKEY** in the 3D window to get a menu that allows rotation of the UV coordinates.
- Sometimes it is necessary to move image files to a new location on your hard disk. Press **NKEY** in the ImageWindow to get a Replace Image name menu. You can fill in the old directory name, and the new one. Pressing OK changes the paths of all images used in Blender using the old directory. (Note: use as new directory the code "//" to indicate the directory where the Blender file is).
- You can also use FaceSelect and VertexPaint (VKEY) simultaneously. Vertex painting then only works on the selected faces. This feature is especially useful to paint faces as if they don't share vertices. Note that the vertex colors are used to modulate the brightness or color of the applied image texture.



Figure 11-51. vertex colors modulate texture.

# LSCM Unwrap

LSCM means Least Squares Conforming Map. This is an advanced mathematical method to automatically create a UV-mapping while keeping texture stretch and deformation to a minimum. It works by preserving local angles. Just as any other existing UV unwrapping mode, it will unwrap the selected faces in UV Face Select Mode. It is available by either pressing the **UKEY** key, and then choosing LSCM, or by choosing LSCM Unwrap from the UV Calculation panel.

To be able to correctly unwrap a mesh with LSCM, you must make sure your mesh can be flattened without too much deformations (in mathematical term, it should be equivalent to a disc). This is done by defining seams, i.e. places where the mesh will be cut. You don't need to add seam if the mesh can be unwrapped directly in a plane.

In Edit Mode, selected edges can be marked or cleared as seams, using **CTRL-E**. Here you can see a cube with seams, and the resulting UV map after applying LSCM.



Figure 11-52. LSCM unwrapping method.

Often a mesh cannot be unwrapped as only one group of faces, but must be cut up into multiple groups. If seams divide the selected faces into multiple face groups, then LSCM unwrapping will unwrap them separately, and position them in the UV Editor so the face groups don't overlap. To ease selection of face groups, Select Linked in UV Face Select Mode (press **LKEY**) will select all linked faces, if no seam divides them. This way, you can select a face group by selecting one face of the group, and executing Select Linked.

To further tweak the result, UVs in the UV Editor can be pinned to a fixed position. If LSCM is executed, these UVs will stay in place, and the resulting UV map will adapt to the pinned UVs. In the UV Editor, selected UVs will be pinned or unpinned by pressing **PKEY** or **ALT-P**. Pressing **EKEY** in the UV Editor will launch LSCM unwrapping on the faces visible in the UV Editor. Pinned UVs are marked in red.

# **Texture Paint**

Once you have loaded an image into the UV Editor, you can modify it using the Texture Paint mode. Use the Paint Tool option in the View menu, to modify paintbrush Size, Opacity, and Colour. Currently there is only a default brush to paint, but work is being done to provide more brushes.

All changes you make will be instantly reflected in the 3D View if the model is in potato mode. However the modified texture will not be saved until you explicitly do so. Use the Save Image option in the Image menu to save your work with a different name or overwriting the original image.



Figure 11-53. The "Paint" tool in action.

Notice that the Draw Shadow Mesh option becomes very helpful to keep a reference of the UV map while texture painting.

# **Rendering and UV coordinates**

Even without an Image assigned to faces, you can render textures utilizing the UV coordinates. For this, use the green UV button in the MaterialButtons (F5) menu.

If you want to render the assigned Image texture as well, you will have to press the TexFace button in the MaterialButtons. Combine this with the VertexCol option to use vertex colors as well.

# **Unwrapping Suzanne**

*Relevant to Blender 2.34* 

by Claudio 'Malefico' Andaur

When dealing with complex models like characters, the need for more powerful tools becomes apparent. Since Blender 2.34 several new tools have been incorporated into the Blender source code, like Seams and the LSCM unwrapping method.

#### Chapter 11. Textures

In this tutorial, I'm going to use two approaches to fully unwrap Suzanne to show you how to use these new tools.

## Easy as it "seams"

A mesh can be organized with the use of seams which provide a nice way to create "Face Groups" prior to doing an unwrap. These "groups" are just a way to call a selection of faces, and not a separate entity like Vertex Groups are. Seams are created by selecting a loop of vertices in Edit Mode and pressing **CTRL-E** or, using the menus, by selecting Edge Menu->Mark Seam sub-option. A thick continuous line will be drawn in the 3D window showing the newly created Seam. It is possible to visualize this seam both in Edit Mode and in Face Select Mode activating the Draw Seams option in the Edit buttons.

Once a closed seam is marked, we can select each side of the model in Face Select Mode, by selecting first a face, and then pressing the **LKEY**. All connected faces that are isolated from the rest by this seam will be selected. By strategically creating seams in our model, we will be able to work later on with this group of faces only, thus greatly simplifying our job.

Ideally we should mark a seam wherever we want the UV map to have "cuts". For instance to isolate the arms of a character from its torso, we should mark a closed seam in each shoulder. There will be cases where we will not need to mark a closed seam but an open one.

Let's do some tests. Add a Suzanne to our scene, and select the vertical central loop of vertices, be sure to completely select it. Press **CTRL-E** to mark this loop as a seam. Let's test it, enter Face Select Mode, we should see the seam. Select a face, and the press **LKEY**, all faces from this side should have been selected. Had the whole model been selected instead, this would mean that we have missed some vertices from the central loop, if this is the case please go and select the missing vertices. Easy, right ? That's the idea.



Figure 11-54. Suzanne with marked seams.

Create some more seams, try to isolate complex zones from each other, like the ears from the head. When we unwrap it, these seams will act as cuts in the surface and also in the UV map.

Remember using the vertex loop selection tool **ALT-B** to help you in selecting vertex loops for seams. An interesting loop to add a seam to is the main face loop of Suzanne, so we can separate the front side from the back.
## Unwrapping the mesh

Open an UV Editor window besides the 3D viewport. Now select the model and enter Face Select Mode.

Select the faces in the left and right side of Suzanne's head including the ears. We are going to unwrap these groups using the Sphere option which will give us a nice starting point for LSCM later on. It is very likely that you'll get something "almost" perfect. However, I always get a couple of faces on the wrong side of the map. This can be easily fixed by hand, but I'm such a lazy guy that I will not do anything at all. Press **CKEY** in the UV Editor to enter Active Face Selection mode. Select some of the faces around the "ears", just the ones that look radially unwrapped. Leave this mode and select the vertices in the upper central zone. The ones in the central lower part are a bit tricky. We will do them in a while. Now, with these vertices selected, press **PKEY**. This is the Pin command. It will fix the locations of the selected vertices so "nothing" can damage our layout. More on this later.



Figure 11-55. Pinning UVs.

Now that we have pinned the nicely unwrapped vertices, unselect everything in the 3D window apart from the two faces in the chin. In the UV Editor, pin the central two vertices of these faces that were unaccessible previously. We are almost done.



Figure 11-56. Accessing occluded UVs.

#### Chapter 11. Textures

Now it's time for some LSCM magic. In the 3D window select all faces linked to the ones in the chin, (use **LKEY**, remember?). You should not select the ears this time. The selected faces should appear mapped in the UV editor. Now press **EKEY**. Blender will ask about doing LSCM. Accept the query. Wonderful things will happen. Suddenly, the "unpinned" faces will be relocated to a nicer location, overlapping of faces is magically fixed. Cameron Diaz is on the phone.

If you select the ear faces in the 3D window, you will see they are still there, don't worry about them right now.

Now we should make some space in the map for the front faces. We can do this by selecting the border vertices and scaling them. The UV Editor supports the Proportional Editing Tool (**OKEY**) just like meshes. It is very useful for making some space here without overlapping faces.



Figure 11-57. LSCM in action.

Once we have made some space, select the front faces, set the viewport to the front view, and mapped the faces using the "From Window" option. Scale them down a bit in Y or X so they fit in the space we have reserved for them in the map. Just press **SKEY** followed by **YKEY** or **XKEY** like you would do with meshes.

We now have the front faces nicely unwrapped in the UV Editor.



Figure 11-58. Front group mapped "From Window".



Figure 11-59. Two islands in the map.

If we unselect the currently selected faces in the 3D window, we will not be able to see them anymore in the UV editor, but you can activate the Draw Shadow Mesh option in the UV editor's View menu to help you visualize the unselected faces.

## **Stitching the Map**

Now we need to join these two "islands". Activate Stick UVs to Mesh (**CTRL-C**) so we can select all UVs to be stitched at once. Select a UV vertex from the face contour, you will see its homologous in the other island selected too. "Stitch" them together by pressing **VKEY**. They will merge into a UV located at the middle of the original ones.



Figure 11-60. Stitching.

Now, it might be a little tedious to continue vertex by vertex. Select all vertices in the border of Suzanne's face. Now stitch. Pin the selected vertices. Remember to pin only if there are no overlaps. If after a stitch, there are overlapping faces, keep stitching the rest and do not pin the stitched vertex.



Figure 11-61. Stitching all together.

Now we have both islands stitched. However there are zones where faces are overlapped. Be sure to have pinned the "nice" vertices and do some LSCM again. Just press **EKEY**, you don't need to select anything. If you do well, each LSCM step will fix vertices location at overlapping faces, that you should pin to "keep".

Feel free to scale or move the vertices before doing a LSCM calculation but always select pairs to keep the symmetry controlled.

## Again, please ;-)

All right. We have used a bunch of tools and got a more or less decent UV map. However in honor to truth, we could have done it a lot faster and better.

These seams I marked, do not have to be "closed" necessarily. As I stated before, these are only cuts in the surface which Blender will use to unfold it. So, if we are clever

enough we could do almost all the work in one or two LSCM steps.

Unmark all seams in Suzie. The main idea is to keep the stretching areas as far from the view as possible. Like a plastic surgeon, we don't want to leave our "seams" in evidence. So, I will mark a seam from somewhere in the middle of the top part of Suzie's head, to somewhere near the chin, without touching the face. I've just loop selected the central vertex loop, and then unselected the vertices near the face. Mark it as a seam. Notice that this seam is "open" meaning it's not a complete loop around the head as in our first unwrap.



Figure 11-62. Seams revisited.

Now I've selected a short loop from the initial vertex of the previous seam, to somewhere near one ear. Then select the symmetrical loop and marked both as seams. That's all.

Select all faces except the eyes faces, and unwrap them using LSCM. This definitely looks better.



Figure 11-63. New LSCM unwrap...

Now, we can stitch and pin some vertices just like we did before, but with a lot less work.

## Finishing it up

Now we are going to use the methods explained before, to fix all overlapping faces in Suzanne's ears.

First, try to separate the outer UVs from both ears, by selecting them and scale them in X. Pin the UVs and do a LSCM step. You will see that ear UVs start to unfold. Continue with an inner pair of UVs and scale them and move them away from the center of each ear. Pin the newly moved UVs. Do a LSCM step after every pinning. After a couple more steps, things should look pretty clear.

*Chapter 11. Textures* 



Figure 11-64. Unfolding the ears.

You can help yourself selecting whole faces from the inner part and scale them. You can even select the whole ear and rotate them 5 or 10 degrees clockwise and anticlockwise (depending on each ear).

With a little patience, you will get an almost completely pinned ear, with no overlapping whatsoever.





Figure 11-65. The unwrapping stages. Notice the pinned vertices.

There is no too much work to do. We can select the eyes face groups and map them From Window or LSCM. Since the eye meshes are not linked to the rest of the mesh, the Stitch command will not work. If you want to merge the eyes with the rest of the face you will have to do it with the Weld command (**WKEY**) instead. However I prefer having them aside, so when painting the texture I can give some extra detail to the eyes more comfortably.

If your character has eyelids, it is advisable to uvmap them almost closed in order to have a nice surface to paint the texture later on.

## What now ?

Once we have completed the unwrapping, we can export it using the Save UV Layout option in the menu, which will launch a Python script included with the Official Blender Release. There you have to set the Image size (remember that the UV layout is a *square* image), and set a proper name to it. It will save a TGA image of your UV map which you can load into Gimp, Photoshop or any other software, as a reference layer for painting your texture.



Figure 11-66. Final exported UV layout.

To be able to use this UV map as a reference, you need your 2D software to be able to manage transparent layers. In my case I chose Gimp since I work on a Linux box, but you can use whatever you find suitable.

Create a new image, same size as the UV map, and load the map into one layer. Add a few more layers on top of the map layer. I have created three layers, named COLOR, BUMP and SPEC-REF. These will generate three different images which I will use as separate texture maps for Col, Nor and Ref/Spec channels in my Blender material. However you might want to use a unique texture map for everything. The good thing about using several textures is that you can for instance tweak the Bumping of your material without altering the Colour work.

#### Chapter 11. Textures

Capas	Canales	Caminos	
Modo:	Normal	🗘 🗌 Fijar trans.	
Opacid	ad:	67	.5
	000	BUMP	-
	107	SPEC-REF	
•	6	COLOR	
۲	They are	UV Layout	
		BACKGROUND	-
۵	)[		

Figure 11-67. Gimp is a 2D app that supports transparent layers.

I have switched every layer off except for the layer I'm working on, and the UV map layer. You must reduce the opacity of the working layer so you can have a view of the UV map all the time.



Figure 11-68. Using the UV map as reference for texture maps.

The colour map doesn't have too many mysteries. These colours will be mapped over the model exactly as they look. The SPEC-REF and BUMP textures are a little more sophisticated. Everything in white will look as "more reflective" this is more lighted, in the first case, or "bumpy" in the second case. The black areas at the contrary will look darker or depressed respectively.



Figure 11-69. The three texture maps.

You can generate as many texture maps as channels you have available in Blender, counting Raytraced Mirroring, Translucency, Emit, Specularity, etc. All these texture maps will share the same UV map as long as you indicate it so using the UV option in the Texture Coordinate Input panel in the material buttons.

To use this texture maps we have created, we need to create first a material, and then the required Image textures. Take care in activating the UV option for every texture map, and applying to the intended texture channel. If you use the Colour map for Bump, it will look a little weird. You can mix a material colour with the Colour map if you feel so. Just slide down the Col slider for that texture.

You only need to load the texture map in the UV editor if you want to tweak the map based on the how the texture looks, or if you want to do some extra paint on it using the texture painting tools. Otherwise it's not really needed to get your render done.



Figure 11-70. Adjusting the colour texture map.

Well, there is no much left to say about UV texturing. In the rendered example I have used two materials with almost the same settings for the head and the eyes. I have used a non-zero value for Emit in the eyes, but everything else is the same. You can tweak the UV map, so the texture map fits better if you need it.

See you and keep blending !

#### Chapter 11. Textures



Figure 11-71. Final render. Really ugly textures... I should paint them again...

# **Texture Plugins**

#### Relevant to Blender v2.31

As a final note on texture, let's look at the fourth texture type button, Plugin.

Blender allows the dynamic linking at run time of shared objects, both texture and sequence plugins. In both cases these objects are pieces of C code written according to a given standard (Chapter 26). In the case of texture plugins, these chunks of code define functions accepting coordinates as input and providing a Color, Normal and Intensity output, exactly as the procedural Textures do.

To use a Texture plugin, select this option, and then click the Load Plugin button which appears in the Texture Buttons. A neighboring window will turn into a File Select window in which you can select a plugin. These plugins are .dll files on Windows and .so files on various Unix flavors.

Once a plugin is loaded it turns the Texture Buttons window into its own set of buttons, as described in the individual plugin references.

# Chapter 12. Lighting

## Introduction

Relevant to Blender v2.31

Lighting is a very important topic in rendering, standing equal to modelling, materials and textures.

The most accurately modelled and textured scene will yield poor results without a proper lighting scheme, while a simple model can become very realistic if skilfully lit.

Lighting, sadly, is often overlooked by the inexperienced artist who commonly believes that, since real world scenes are often lit by a single light (a lamp, the sun, etc.) a single light will also do in computer graphics.

This is false because in the real world even if a single light source is present, the light shed by such a source bounces off objects and is re-irradiated all over the scene making shadows soft and shadowed regions not pitch black, but partially lit.

The physics of light bouncing is simulated by Ray Tracing renderers and can be simulated within Blender by resorting to the Radiosity (Chapter 18) engine.

Ray tracing and radiosity are slow processes. Blender can perform much faster rendering with its internal scanline renderer. A very good scanline renderer indeed. This kind of rendering engine is much faster since it does not try to simulate the real behaviour of light, assuming many simplifying hypothesis.

In this chapter we will analyse the different type of lights in Blender and their behaviour, we will analyse their strong and weak points, ending with describing a basic 'realistic' lighting scheme, known as the three point light method, as well as more advanced, realistic but, of course, CPU intensive, lighting schemes.

# Lamp Types (-)

*Relevant to Blender v2.31 MISSING AREA LIGHT AND RAYSHADOW HINTS* Blender provides four Lamp types:

- Sun Light
- Hemi Light
- Lamp Light
- Spot Light

Any of these lamps can be added to the scene by pressing **SPACE** and by selecting the Lamp menu entry. This action adds a *Lamp Light* lamp type. To select a different lamp type, or to tune the parameters, you need to switch to the Shading Context window Figure 12-1 (F5) and Lamp Sub-context ( 🗵 ).

A column of toggle buttons, in the Preview Panel, allows you to choose the lamp type.

V Preview			Texture and Input Map To
Lamp Spot Sun Hemi	c         LALamp         IP Dist 28:001           Guad         Energy 100 11         Energy 100 11           Sphere         F1:00         Energy 100 11           Layer         6:100         Energy 100 11           Negative         8:100         Energy 100 11           No Species         Quad 1:000         Energy 100 11	SpetS 43 0.0           Shedowe           Very Dirth 20 and 2	Add New           Glob         View         Object            at X 0.00         - sizeX 1000            at X 0.00         - sizeX 1000            at 2 0.00         - sizeX 1000

Figure 12-1. Lamp Buttons.

The lamp buttons can be divided into two categories: Those directly affecting light, which are clustered in the Lamp and Spot Panels, and those defining textures for the light, which are on the right-hand Texture Panel, which has two Tabs. The tabs are very similar to those relative to materials. In the following subsections we will focus on the first two Panels (Figure 12-2), leaving a brief discussion on texture to the **Tweaking Light** section (the Section called *Tweaking Light*).

	Shadows Only Shadow Haloint 1.000 ShadowBuffer Size: 512
Layer G 1.000 B 1.000 B 1.000 C 1.000	Square «ClipSta: 0.50» ClipEnd: 40.00 Halo
No Diffuse Quad1 0.000 No Specular Quad2 1.000	Samples: 3 Halo step: 0 Bias: 1.000 Soft: 3.00

Figure 12-2. Lamp General Buttons.

The Lamp Panel contains buttons which are mostly general to all lamp types, hence deserve to be explained beforehand.

Negative - Makes the light cast 'negative' light, that is, the light shed by the lamp is subtracted, rather than added, to that shed by any other light in the scene.

Layer - Makes the light shed by the lamp affect only the objects which are on the same layer as the lamp itself.

No Diffuse - Makes the light cast a light which does not affect the 'Diffuse' material shader, hence giving only 'Specular' highlights.

No Specular - Makes the light cast a light which does not affect the 'Specular' material shader, hence giving only 'Diffuse' shading.

Energy - The energy radiated by the lamp.

R, G, B sliders - The red, green and blue components of the light shed by the lamp.

## Sun Light

The simplest light type is probably the Sun Light (Figure 12-3). A Sun Light is a light of constant intensity coming from a given direction. In the 3D view the Sun light is represented by an encircled yellow dot, which of course turns to purple when selected, plus a dashed line.

This line indicates the direction of the Sun's rays. It is by default normal to the view in which the Sun lamp was added to the scene and can be rotated by selecting the Sun and by pressing **RKEY**.



Figure 12-3. Sun Light.

The lamp buttons which are of use with the Sun are plainly those described in the 'general' section. An example of Sun light illumination is shown in Figure 12-4. As is evident, the light comes from a constant direction, has a uniform intensity and *does not cast shadows*.

This latter is a very important point to understand in Blender: no lamp, except for the "Spot" type, casts shadows. The reason for this lies in the light implementation in a scanline renderer and will be briefly discussed in the 'Spot' and 'Shadows' subsections.

Lastly, it is important to note that since the Sun light is defined by its energy, colour and *direction*, the actual *location* of the Sun light itself is not important.



Figure 12-4. Sun Light example.

Figure 12-5 shows a second set-up, made by a series of planes 1 blender unit distant one from the other, lit with a Sun light. The uniformity of lighting is even more evident. This picture will be used as a reference to compare with other lamp types.

#### Chapter 12. Lighting



Figure 12-5. Sun Light example.

**Sun Tips:** A Sun light can be very handy for a uniform clear day-light open-space illumination. The fact that it casts no shadows can be circumvented by adding some 'shadow only' spot lights. See the Section called *Tweaking Light*!

## Hemi Light

The Hemi light is a very peculiar kind of light designed to simulate the light coming from a heavily clouded or otherwise uniform sky. In other words it is a light which is shed, uniformly, by a glowing hemisphere surrounding the scene (Figure 12-6).

It is probably the least used Blender light, but it deserves to be treated before the two main Blender Lights because of its simplicity.

This light set-up basically resembles that of a Sun light. Its location is unimportant, while its orientation is important. Its dashed line represents the direction in which the maximum energy is radiated, that is the normal to the plane defining the cut of the hemisphere, pointing towards the dark side.



Figure 12-6. Hemi Light conceptual scheme.

The results of a Hemi Light for the 9 sphere set up are shown in Figure 12-7 the superior softness of the Hemi light in comparison to the Sun light is evident.



Figure 12-7. Hemi Light example.

**Hemi Light Tip:** To achieve quite realistic, were it not for the absence of shadows, outdoor lighting you can use both a Sun light, say of Energy 1.0 and warm yellow/orange tint, and a weaker bluish Hemi light faking the light coming from every point of a clear blue sky. Figure 12-8 shows an example with relative parameters. The figure also uses a World. See the pertinent chapter.



Figure 12-8. Outdoor Light example. Sun Light Energy=1 RGB=(1.,0.95,0.8) Sun direction in a polar reference is (135°,135°). Hemi Light Energy=0.5 RGB=(0.64,0.78,1.) pointing down.

## Lamp Light

The Lamp light is an omni-directional point light, that is a dimensionless point radiating the same amount of light in all directions. In blender it is represented by a

#### Chapter 12. Lighting

plain, circled, yellow dot.

Being a point light source the light rays direction on an object surface is given by the line joining the point light source and the point on the surface of the object itself. Furthermore, light intensity decays accordingly to a given ratio of the distance from the lamp.

Besides the above-mentioned buttons three more buttons and two sliders in the Lamp Panel are of use in a Lamp light (Figure 12-9):

Distance - This gives, indicatively, the distance at which the light intensity is half the Energy. Objects closer than that receive more light, object further than that receive less light.

Quad - If this button is off, a linear - rather unphysical - decay ratio with distance is used. If it is on, a more complex decay is used, which can be tuned by the user from a fully linear, as for Blender default, to a fully - physically correct - quadratic decay ratio with the distance. This latter is a little more difficult to master, it is governed by the two Quad1 and Quad2 Num Buttons and will be explained later on.

Sphere - If this button is pressed the light shed by the source is confined in the Sphere of radius Distance rather than going to infinity with its decay ratio.

▼ Lamp	▼ Spot
	Shadows         SpotSi 45.00         Image: SpotSi 45.00           Shadows         OnlyShadow         Image: SpotSi 45.00         Image: SpotSi 45.00           OnlyShadow         OnlyShadow         Image: SpotSi 45.00         Image: SpotSi 45.00         Image: SpotSi 45.00
Layer G 1.000 II Negative B 1.000 II	Square ClipSta: 0.50 ClipEnd: 40.00 Halo
No Diffuse         Quad1 0.000         Image: Control of the second secon	Samples: 3         Halo step: 0           Bias: 1.000         Soft: 3.00

Figure 12-9. Lamp Light Buttons.

Following Figure 12-10 shows the same set-up as in the latter Sun light example, but with a Lamp light of different Distance values and with Quadratic decay on and off.



Figure 12-10. Lamp Light example. In Quad examples Quad1=0, Quad2=1.

The effect of the Distance parameter is very evident, while the effect of the Quad button is more subtle. In any case the absence of shadows is still a major issue. As a matter of fact only the first plane should be lit, because all the others should fall in the shadow of the first.

For the Math enthusiasts, and for those desiring deeper insight, the laws governing the decay are the following.

Let D be the value of the Distance Numeric Button, E the value of the Energy slider and r the distance from the Lamp to the point where rhe light intensity I is to be computed.

If Quad and Sphere buttons are off:

$$I = E \frac{D}{D+r}$$

It is evident what affirmed before: That the light intensity equals half the energy for r = D.

If Quad Button is on:

$$I = E \frac{D}{D + Q_1 r} \frac{D^2}{D^2 + Q_2 r^2}$$

This is a little more complex and depends from the Quad1 ( $Q_1$ ) and Quad2 ( $Q_2$ ) slider values. Nevertheless it is apparent how the decay is fully linear for  $Q_1=1$ ,  $Q_2=0$  and fully quadratic for  $Q_1=0$ ,  $Q_2=1$ , this latter being the default. Interestingly enough if  $Q_1=Q_2=0$  then light intensity does not decay at all.

If the Sphere button is on the above computed light intensity I is further modified by multiplication by the term which has a linear progression for r from 0 to D and is identically 0 otherwise.

If the Quad button is off and the Sphere button is on:

$$I = E \frac{D}{D+r} \cdot \begin{cases} \frac{D-r}{D} & \text{if } r < D \\ 0 & \text{otherwise} \end{cases} = \\ = \begin{cases} E \frac{D-r}{D+r} & \text{if } r < D \\ 0 & \text{otherwise} \end{cases}$$

If both Quad and Sphe buttons are on:

$$I = E \frac{D}{D + Q_1 r} \frac{D^2}{D^2 + Q_2 r^2} \cdot \begin{cases} \frac{D - r}{D} & \text{if } r < D \\ 0 & \text{otherwise} \end{cases} = \\ = \begin{cases} E \frac{D - r}{D + Q_1 r} \frac{D^2}{D^2 + Q_2 r^2} & \text{if } r < D \\ 0 & \text{otherwise} \end{cases}$$





Figure 12-11. Light decays: a) Blender default linear; b) Blender default quadratic with Quad1=0, Quad2=1; c) Blender quadratic with Quad1=Quad2=0.5; d) Blender quadratic with Quad1=Quad2=0. Also shown in the graph the same curves, in the same colours, but with the Sphere button turned on.

Lamp Light Tip: Since the Lamp light does not cast shadows it shines happily through walls and the like. If you want to achieve some nice effects like a fire, or a candle-lit room interior seen from outside a window, the Sphere option is a must. By carefully working on the Distance value you can make your warm firelight shed only within the room, while illuminating outside with a cool moonlight, the latter achieved with a Sun or Hemi light or both.

## Spot Light

The Spot light is the most complex of Blender lights and indeed among the most used thanks to the fact that it is the only one able to cast shadows.

A Spot light is a cone shaped beam generated from the light source location, which is the tip of the cone, in a given direction. Figure 12-12 should clarify this.



Figure 12-12. Spot Light Scheme.

The Spot light uses all buttons of a Lamp Light, and with the same meaning, but it is so more complex that it needs a second Panel of buttons (Figure 12-13): Spot.

## **Spot Options**

🔻 Spot				
	SpotSi 75.00			
Shadows	SpotBI 0.150			
OnlyShadow	HaloInt 1.000			
	Shadow Buffer Size: 2880			
Square				
	∢ ClipSta: 8.00⊁	ClipEnd: 30.00		
Halo				
	✓ Samples: 3 →	🔹 Halo step: 0 🕨		
	<ul> <li>▲ Bias: 1.000 →</li> </ul>	<ul> <li>Soft: 3.00 →</li> </ul>		

Figure 12-13. The Lamp Options buttons

Shadows - Toggles shadow casting on and off for this spot.

Only Shadow - Let the spot cast only the shadow and no light. This option will be analysed later on in the Section called *Tweaking Light*.

Square - Spot lights usually by default cast a cone of light of circular cross-section. There are cases where a square cross section would be helpful, and indeed have a pyramid of light rather than a cone. This button toggles this option.

Halo - Let the spot cast a halo as if the light rays were passing through a hazy medium. This option is explained later on in the 'Volumetric Light' section (the Section called *Volumetric Light*).

#### Spot Buttons



Figure 12-14. Spot Light Buttons.

The rightmost column of buttons in the Spot Panel handles Spot geometry and shadows (Figure 12-14):

SpotSi - The angle at the tip of the cone, or the Spot aperture.

SpotB1 - The blending between the light cone and the surrounding unlit area. The lower the sharper the edge, the higher the softer. Please note that this applies only to the spot edges, not to the softness of the edges of the shadows cast by the spot, these latter are governed by another set of buttons described in the 'Shadows' subsection.

HaloInt - If the Halo button is On this slider defines the intensity of the spot halo. Again, you are referred to the Section called *Volumetric Light*.

The bottom button group of the Spot light governs shadows and it is such an ample topic that it deserves a subsection by its own. Before switching to Shadows, Figure 12-15 shows some results for a Spot light illuminating our first test case for different configurations.



Figure 12-15. Spot Light Examples for SpotSi=45°

Note: In Figure 12-15 shadows are turned off! Shadows are treated in the next section.

# **Ray Shadows**

TO BE WRITTEN

# **Buffer Shadows**

Relevant to Blender v2.31

The lighting schemes analysed up to now produce on the objects only areas which are more or less lit, but no cast- or self-shadowing, and a scene without proper shadowing looses depth and realism.

On the other hand, proper shadow calculation requires a full - and slow - ray tracer. For a scan liner, as Blender is, shadows can be computed using a *shadow buffer* for shadow casting lights. This implies that an 'image', as seen from the Spot Light itself, is 'rendered' and that the distance for each point from the spotlight saved. Any point of the rendered image further than any of those points is then considered to be in shadow.

The shadow buffer stores this data. To keep the algorithm compact, efficient and fast this shadow buffer has a size which is fixed from the beginning and which in Blender can be from 512x512 to 5120x5120. The higher value is the most accurate.

The user can control the algorithm via the bottom buttons in the Spot Panel (Figure 12-16).

▼ Spot	
	SpotSi 75.00
Shadows	SpotBI 0.150
Only Shadow	HaloInt 1.000
	Shadow Buffer Size: 2880 ►
Square	
	ClipSta: 8.00 ► ClipEnd: 30.00
Halo	
	✓ Samples: 3 ➤ < Halo step: 0 ➤
	▲ Bias: 1.000 ▶ ▲ Soft: 3.00 ▶

Figure 12-16. Spot Light shadow buttons.

ShadowBuffSize - Numeric Button, from 512 to 5120, defining the shadow buffer size.

ClipSta, ClipEnd - To further enhance efficiency the shadow computations are actually performed only in a predefined range of distances from the spot position. This range goes from ClipSta, nearer to the Spot light, to ClipEnd, further away (Figure 12-12). All objects nearer than ClipSta from the Spot light are never checked for shadows, and are always lit. Objects further than ClipEnd are never checked for light and are always in shadow. To have a realistic shadow ClipSta must be less than the smallest distance between any relevant object of the scene and the spot, and ClipEnd larger than the largest distance. For the best use of the allocated memory and better shadow quality, ClipSta must be as large as possible and ClipEnd as small as possible. This minimizes the volume where shadows will be computed.

Samples - To obtain soft shadows the shadow buffer, once computed, is rendered via its own anti-aliasing algorithm which works by averaging the shadow value over a square of a side of a given number of pixels. Samples is the number of pixels. Its default is 3, that is a 3x3 square. Higher values give better anti-aliasing, and a slower computation time.

Bias - Is the bias used in computing the shadows, again the higher the better, and the slower.

Soft - Controls the softness of the shadow boundary. The higher the value, the softer and more extended the shadow boundaries will be. Commonly it should be assigned a value which ranges from the same value of the Sample NumButton to double that value.

Halo step - The stepping of the halo sampling for volumetric shadows when volumetric light is on. This will be explained in the Section called *Volumetric Light*.



Figure 12-17. Spot Light shadow examples.

**Note:** For Shadows to be rendered, they must be enabled at a *global* level. This means that the shadow Button of the Render Panel in the Scene Context and Render Buttons must be on!

# Volumetric Light

Relevant to Blender v2.31

Volumetric light is the effect you see in a hazy air, when the light rays become visible because of light scattering which occurs due to mist, fog, dust etc.

If used carefully it can add much realism to a scene... or kill it.

The volumetric light in Blender can only be generated for Spot Lights, once the Halo button in the Spot Panel is pressed (Figure 12-18).

🔻 Spot	
	Spot Si 75.00
Shadows OnlyShadow	HaloInt 1.000
Square	Shadow Buffer Size: 2880
Halo	
	Samples: 3         Halo step: 0           Bias: 1.000         Soft: 3.00

Figure 12-18. Spot Light halo button.

If the test set up shown in Figure 12-19 is created, and the Halo button pressed, the rendered view will be like Figure 12-20.



Figure 12-19. Spot Light setup.



Figure 12-20. Halo rendering.

The volumetric light effect is rather strong. The intensity of the Halo can be regulated with the HaloInt slider (Figure 12-21). Lower values corresponding to weaker halos.

Quad	La	mp Spot Sun		Hemi	Dist: 20.00
Shadows	Spo	tSi 45.00 💷	⊇	Energy 1.0	
Halo Laver	Spo	tBI 0.150 🛥	∍	R 1.000	
Negative	Qua	ad1 0.000 <del>-</del>	⊇	G 1.000	
OnlyShadow	Qua	ad2 1.000 ←	티	B 1.000	
No Diffuse	Hal	oint 1.000	Ð	L	
No Specular		ClipSta: 0.50	)(	Samples: 3	Bias: 1.000
adowBuffSize:	512	ClipEnd: 40.00		Halo step: 0	Soft: 3.00

Figure 12-21. Halo Intensity Slider.

The result is interesting. We have volumetric light, but we lack volumetric shadow! The halo passes through the sphere, yet a shadow is cast. This is due to the fact that the Halo occurs in the whole Spot Light cone unless we tell Blender to do otherwise.

The cone needs to be sampled to get volumetric shadow, and the sampling occurs with a step defined by the HaloStep NumButton (Figure 12-22). The default value of 0 means no sampling at all, hence the lack of volumetric shadow. A value of 1 gives finer stepping, and hence better results, but with a slower rendering time (Figure 12-23), while a higher value gives worse results with faster rendering (Figure 12-24).

Quad Sphere	Lamp Spot Sun Hemi Dist: 20.00
Shadows	SpotSi 45.00
Halo	SpotBI 0.150
Layer	
OnlyShadow	Ouad2 1.000
Square	HaloInt 1.000
No Diffuse	
No Specular	ClipSta: 0.50 Samples: 3 Bias: 1.000
adowBuffSize:	512 ClipEnd: 40.00 Halo step: 0 Soft: 3.00

Figure 12-22. Halo Step NumButton.



Figure 12-23. Halo with volumetric shadow, Halo Step = 1



Figure 12-24. Halo with volumetric shadow, Halo Step = 12

HaloStep values: A value of 8 is usually a good compromise between speed and accuracy.

# **Tweaking Light**

#### Relevant to Blender v2.31

Ok, now you've got the basics. Now we can really talk of light. We will work on a single example, more complex than a plain 'sphere over a plane' setup, to see what we can achieve in realistic lighting with Blender.

We will resort to the setup in Figure 12-25. The simian figure is Cornelius, Suzanne's baby brother. He has a somewhat shiny light brown material (R=0.8, G=0.704 B=0.584, Ref=0.7, Spec=0.444, Hard=10 - Yes, not very monkey-like, but we are talking of lights, not of materials!) and stands on a blue plane (R=0.275, G=0.5, B=1.0, Ref=0.8, Spec=0.5, Hard=50). For now he's lit by a single spot (Energy=1.0, R=G=B=1.0, SpotSi=45.0, SpotBl=0.15, ClipSta=0.1, ClipEnd=100, Samples=3, Soft=3, Bias=1.0, BufSize=512).



Figure 12-25. Light tweaking setup.

A rendering of Cornelius in this setup, with OSA=8 and Shadows enabled, gives the result in Figure 12-26. The result is ugly. You have very black, unrealistic shadows on Cornelius, and the shadow cast by Cornelius himself is unacceptable.



Figure 12-26. Simple Light Spot set up.

The first tweak is on ClipSta and ClipEnd, if they are adjusted so as to encompass the scene as tightly as possible (ClipSta=5, ClipEnd=21) the results get definitely better, at least for projected shadow. Cornelius's own shadow is still too dark (Figure 12-27).



Figure 12-27. Single Spot Light set up with appropriate Clipping.

To set good values for the Clipping data here is a useful trick: Any object in Blender can act as a Camera in the 3D view. Hence you can select the Spot Light and switch to a view from it by pressing **CTRL-NUM0**. What you would see, in shaded mode, is shown in Figure 12-28.

All stuff nearer to the Spot than ClipSta and further from the spot than ClipEnd is not shown at all. Hence you can fine tune these values by verifying that all shadow casting objects are visible.



# Figure 12-28. Spot Light Clipping tweak. Left: ClipSta too high; Centre: Good; Right: ClipEnd too low.

What is still really lacking is the physical phenomenon of diffusion. A lit body sheds light from itself, hence shadows are not completely black because some light leaks in from neighbouring lit regions.

This light diffusion is correctly accounted for in a Ray Tracer, and in Blender too, via the Radiosity Engine. But there are set-ups which can fake this phenomenon in an acceptable fashion.

We will analyse these, from simplest to more complex.

## Three point light

The three point light set-up is a classical, very simple scheme to achieve a scene with softer lighting. Our Spot Light is the main, or *Key*, Light of the scene, the one casting shadow. We will add two more lights to fake diffusion.

The next light needed is called the *Back Light*. It is placed behind Cornelius (Figure 12-29). This illuminates the hidden side of our character, and allows us to separate the foreground of our picture from the background, adding an overall sense of depth.

Usually the Back Light is as strong as the Key Light, if not stronger. Here we used an Energy 1 Lamp Light (Figure 12-30).



Figure 12-29. Back Light set up.



Figure 12-30. Key Light only (left) Back Light only (centre) and both (right).

The result is already far better. Finally, the third light is the *Fill* Light. The Fill light's aim is to light up the shadows on the front of Cornelius. We will place the Fill Light exactly at the location of the camera, with an Energy lower than the lower of Key and Back Lights (Figure 12-31). For this example an Energy=0.75 has been chosen (Figure 12-32).



Figure 12-31. Fill Light set up.



# Figure 12-32. Key and Back Light only (left) Fill Light only (centre) and all three (right).

The Fill light makes visible parts of the model which are completely in darkness with the Key and Back light only.

**Color leakage:** The three-point set up can be further enhanced with the addition of a fourth light, especially when a bright coloured floor is present, like in this case.

If there is a bright coloured floor our eye expects the floor to diffuse part of the light all around, and that some of this light impinges on the model.

To fake this effect we place a second spot exactly specular to the Key Light with respect to the floor. This means that - if the floor is horizontal and a z=0, as it is in our example, and the Key light is in point (x=-5, y=-5, z=10), then the floor diffuse light is to be placed in (x=-5, y=-5, z=-10), pointing upward (Figure 12-33).



Figure 12-33. Floor Diffuse Light set up.

The energy for this light should be lower than that of the Key Light (here it is 0.8) and its colour should match the colour of the floor (here R=0.25, G=0.5, B=1.0). The result is shown in Figure 12-34.



Figure 12-34. Four Light set up.

Please note that we used a Spot light and not a lamp, so it would be completely blocked by the floor (shadowed) unless we set this spot shadeless by pressing the appropriate button.

Indeed we could have used a Lamp but if the floor is shiny the light it sheds is more reflected than diffused. Reflected light, physically is itself a cone coming from the specular source.

You can further enhance the effect by making the Spot cast shadows and by setting its ClipStart value high enough so that the plane casts no shadow, or by making it affect only its layer and placing the floor on another layer.

## **Three point light - Outdoor**

By using a Spot light as a key light the previous method is sadly bound to indoor settings or, at maximum, outdoor settings at night time. This is because the Key light is at a finite distance, its rays spread and the floor is not evenly illuminated.

If we were outdoor on a clear sunny day all the floor would be evenly lit, and shadows would be cast.

To have a uniform illumination all over the floor a Sun light is good. And if we add a Hemi light for faking the light coming from all points of the sky (as in Figure 12-8) we can achieve a nice outdoor light... but we have no shadows!

The setup of the Key light (the Sun, R=1.0, G=0.95, B=0.9, Energy=1.0) and the Fill/Back Light (both represented by the Hemi, R=0.8, G=0.9,B=1.0, Energy=0.4) is shown in Figure 12-35 and the relevant rendering in Figure 12-36



Figure 12-35. Sun and Hemi light for outdoor set up.



Figure 12-36. Sun and Hemi light for outdoor rendering.

The lack of shadow makes Cornelius appear as if he were floating in space. To have a shadow let's place a Spot coincident with the Sun and with the same direction. Let's make this spot a Shadow Only Spot (with the appropriate button). If Energy is lowered to 0.9 and all other settings are kept at the values used in the previous example (BufSize=512, Samples=3, Soft=3, Bias=1, ClipSta=5, ClipEnd=21) the result is the one of Figure 12-37 (center).



Figure 12-37. Outdoor rendering.

The shadow is a bit blocky because Cornelius has many fine details and the BufSize is too small, and the Samples value is to low to correctly take them into account. If BufSize is raised to 2560, Samples to 6 and Bias to 3.0 the result is the one in Figure 12-37 (right). Much smoother.

# Area Light

The concept of Light coming from a point is an approximation. No real world light source is dimensionless. All light is shed by surfaces, not by points.

This has a couple of interesting implications, mainly on shadows:

- Sharp shadows do not exist: shadows have blurry edges.
- Shadow edge blurriness depends on the relative positions and sizes of the light, the shadow casting object and the object receiving the shadow.

The first of these issues is approximated with the 'Soft' setting of the Spot light, but the second is not. To have a clearer understanding of this point imagine a tall thin pole in the middle of a flat plain illuminated by the Sun.

The Sun is not a point, it has a dimension and, for us Earthlings, it is half of a degree wide. If you look at the shadow you will notice that it is very sharp at the base of the pole and that it grows blurrier as you go toward the shadow of the tip. If the pole is tall and thin enough its shadow will vanish.

To better grasp this concept have a look at Figure 12-38. The Sun sheds light, the middle object completely obstructs the Sun's rays only in the dark blue region. For a point in the light blue region the Sun is partially visible, hence each of these areas is partially lit.



Figure 12-38. Area light and its shadow.

The light blue region is a partial shadow region where illumination drops smoothly from full light to full shadow. It is also evident, from Figure 12-38 than that this transition region is smaller next to the shadow casting object and grow larger far away from it. Furthermore, if the shadow casting object is smaller than the light casting object (and if the light casting object is the Sun this is often the case) there is a distance beyond which only partial shadow remains Figure 12-39.



Figure 12-39. Area light and its shadow 2

In Blender, if we place a single Spot at a fixed distance from a first plane and look at the shadow cast at a second plane as this second plane gets further away we notice that the shadow gets larger but not softer (Figure 12-40).


Figure 12-40. Spot light and its shadow

To fake an area light with Blender we can use several Spots, as if we were sampling the area casting light with a discrete number of point lights.

This can either be achieved by placing several Spots by hand, or by using Blender's DupliVert feature (the Section called *DupliVerts* in Chapter 22), which is more efficient.

Add a Mesh Grid 4x4. Where the spot is, be sure normals are pointing down, by letting Blender show the Normals and eventually flipping them, as explained in the Section called *Basic Editing* in Chapter 6 (Figure 12-41). Parent the Spot to the Grid, select the Grid and in the Object Context Anim Settings Panel (F7) press DupliVert and Rot. Rot is not strictly necessary but will help you in positioning the Area Light later on. You will have a set of Spots as in Figure 12-42.



Figure 12-41. Grid setup



Figure 12-42. Spot light and its dupliverts

Then decrease the Energy of the Spot. If for a single Spot you used a certain energy, now you must subdivide that energy among all the duplicates. Here we have 16 spots, so each should be allotted 1/16 of Energy (that is Energy=0.0625).

The same two renderings of above, with this new hacked area light will yield the results in Figure 12-43. The result is far from that expected, because the Spot light sampling of the Area light is too coarse. On the other hand, a finer sampling would yield a higher number of duplicated Spots and to unacceptable rendering times.



Figure 12-43. Fake area light with multiple spots.

A much better result can be attained by softening the spots, that is setting SpotBl=0.45, Sample=12, Soft=24 and Bias=1.5 (Figure 12-44).



Figure 12-44. Fake area light with multiple soft spots.

Finally, Figure 12-45 shows what happens to Cornelius once the Key Light is substituted with 65 duplicated Spots of Energy=0.0154 in a circular pattern. Please note how the shadow softly goes from sharp next to the feet to softer and softer as it gets further away from him. This is the correct physical behavior.



Figure 12-45. Cornelius under Area Light.

## **Global Illumination (and Global Shadowing)**

The above techniques work well when there is a single, or at least a finite number of lights, casting distinct shadows.

The only exceptions are the outdoor setting, where the Hemi Light fakes the light cast by the sky, and the Area Light, where multiple spots fake a light source of finite extension.

The first of these two is very close to nice outdoor lighting, were it not for the fact that the Hemi Light casts no shadows and hence you don't have realistic results.

To obtain a really nice outdoor setting, especially for cloudy daylight, you need light coming from all directions of the sky, yet casting shadows!

This can be obtained by applying a technique very similar to the one used for the Area Light setup, but using half a sphere as a parent mesh. This is usually referred to as "Global Illumination".

You can either use a UVsphere or an IcoSphere, the latter has vertices evenly distributed whereas the former has a great concentration of vertices at poles. Using an IcoSphere hence yields a more 'uniform' illumination, all the points of the sky radiating an equal intensity; a UVsphere casts much more light from the pole(s). Personally I recommend the IcoSphere.

Let's prepare a setup, comprising a plane and some solids, as in Figure 12-46. We will use simple shapes to better appreciate the results.

## Chapter 12. Lighting



Figure 12-46. Global Illumination scene.

Switch to top view and add an IcoSphere, a subdivision level 2 IcoSphere is usually enough, a level 3 one yields even smoother results. Scale the IcoSphere so that it completely, and loosely, contains the whole scene. Switch to front view and, in EditMode, delete the lower half of the IcoSphere (Figure 12-47). This will be our "Sky Dome" where the spots will be parented and dupliverted.



Figure 12-47. Sky dome.

Again in Top View add a Spot Light, parent it to the half IcoSphere (**CTRL-P**) and press the DupliVert and Rot buttons exactly as in the previous example. The result, in FrontView, is the one in Figure 12-48.



Figure 12-48. Sky dome with duplicated spots.

This is not what we want, since all spots point outwards and the scene is not lit. This is due to the fact that the IcoSphere normals point outward. It is possible to invert their directions by selecting all vertices in Edit Mode and by pressing the Flip Normals button in the Mesh Tools Panel of the Editing (F9) Context (Figure 12-49).

▼ N	1esh Tools				
	Beauty	Subdivide	Fract Subd		
	Noise	Hash	Xsort		
	To Sphere	Smooth	Split		
	Flip Normals Rem Double Limit: 0.0				
	Extrude				
	Screw	Spin	Spin Dup		
	∢ Degr: 90 ►	🔹 Steps: 9 🕨	🔹 Turns: 1 🕨		
	Keep Original Clockwise				
	Extrude Dup Offset: 1.000				

Figure 12-49. Flipping normals.

This leads to the new configuration in Figure 12-50.



Figure 12-50. Correct sky dome and dupliverted Spot Lights.

To obtain good results select the original Spot Light and change its parameters to a wide angle with soft boundaries (SpotSi=70.0; SpotBl=0.5); with suitable ClipSta and ClipEnd values; in this case 5 and 30, respectively, in any case appropriate values to encompass the whole scene; increase samples to 6 and softness to 12. Decrease energy to 0.1; remember you are using many spots, so each must be weak. (Figure 12-51).

LA:Lamp     Guad     Energy 0.100	SpotBi 0.500
Spriere         R 1.000           Layer         G 1.000           Negative         B 1.000	Square ClipSta: 5.00 ClipEnd: 30.00
No Diffuse Quad1 0.000 I	Halo

Figure 12-51. Spot Light setup.

Now you can make the rendering. If some materials are given and a world set, the result should be that of Figure 12-52. Note the soft shadows and the 'omnidirectional' lighting. Even better results can be achieved with a level 3 IcoSphere.



#### Figure 12-52. Spot Light setup.

This Global Illumination technique effectively substitutes, at a very high computational cost, the Hemi for the above outdoor setting.

It is possible to add a directional light component by faking the Sun either via a single spot or with an Area Light.

An alternative possibility is to make the IcoSphere 'less uniform' by subdividing one of its faces a number of times, as is done for one of the rear faces in Figure 12-53. This is done by selecting one face and pressing the Subdivide button, again in the Mesh Tools Panel of the Editing (F9) Context. Then de-select all, re-select the single inner small face and subdivide it again, and so on.



Figure 12-53. Making spots denser in an area.

The result is a very soft directional light together with a global illumination sky dome or, briefly, an anisotropic sky dome (Figure 12-54). This is quite good for cloudy conditions, but not so good for clear sunny days. For really clear days, it is better to keep the sky dome separate from the Sun light, so as to be able to use different colours for each.



Figure 12-54. Anisotropic skydome render.

# Chapter 13. The World and The Universe

Blender provides a number of very interesting settings to complete your renderings by adding a nice background, and some interesting 'depth' effects. These are accessible via the Shading Context (F5) and World Buttons sub-context ( ) shown in Figure 13-1. By default a very plain uniform world is present. You can edit it or add a new World.

😑 🔻 Panels 🛛 🗟 🔍 🔍 🖾	₩●■♣● < 1 →	
Freview     Feal Blend Faper	World     SW0-World     SW1 X F     HoR 0.05       Ze6 0.10       Ze6 0.10	Texture and Input Map To     Add New     d      Vew Object     ed     dx 0.00 - sizeX 1.000     ed 20.00 - sizeX 1.000

**Figure 13-1. World Buttons** 

# The World Background

Relevant to Blender v2.31

The simplest way to use the World Buttons is to provide a nice gradient background for images. The buttons in the World Panel (Figure 13-2) allow you to define the color at the horizon (HOR, HOG, HOB buttons) and at the zenith (ZeR, ZeG, ZeB buttons).



Figure 13-2. Background colors

These colors are interpreted differently, according to the Buttons in the Preview Panel (Figure 13-2):

- Blend The background color is blended from horizon to zenith. If only this button is pressed, the gradient runs from the bottom to the top of the rendered image regardless of the camera orientation.
- Real If this button is also pressed the blending is dependent on the camera orientation. The horizon color is exactly at the horizon (on the x-y plane), and the zenith color is used for points vertically above and below the camera.
- Paper If this button is pressed the gradient occurs on the zenith-horizon-zenith colors. Thus, there are two transitions on the image, which reflect the camera rotation but which keep the horizon color to the center and the zenith color to the extremes.

The World Buttons also provide a Texture Panel with two Tabs. They are used much like the Materials textures, except for a couple of differences (Figure 13-3):

- There are only six texture channels.
- Texture mapping Has only the Object and View options, with View being the default orientation.

• Affect - Texture affects color only, but in four different ways: It can affect the Blend channel, making the Horizon color appear where the texture is non-zero; the color of the Horizon; and the color of the Zenith, up or down (ZenUp, ZenDo).

Texture and Input Map To	Texture and Input Map To
Add New	Blend Hori ZenUp ZenDo
View Object	Stenci Neg No RG Mix Mul Add Sub
dX 0.00 → sizeX 1.000 dY 0.00 → sizeY 1.000 dZ 0.00 → sizeZ 1.000	R 1.000         I           G 0.000         I           B 1.000         I           Nor 0.50         I           DVcr 1.0         Vcr 1.000

**Figure 13-3. Texture buttons** 

# Exposure (-)

TO BE WRITTEN

## Mist

Relevant to Blender v2.31

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender basically mixes the background color with the object color and enhances the strength of the former, the further the object is away from the camera. The Mist settings are in the Mist Stars Physics Panel and are shown in Figure 13-4.

Sta: 0.00 Star Dist: 15.00
Sta: 0.00 StarDist: 15.00
Di: 0.00 ▶    < MinDist: 0.00
Hi: 0.00 → Size:2.00 ■
i0.000 Colnoise:

Figure 13-4. Mist buttons

The Mist Button toggles mist on and off. The row of three Toggle Buttons below this button set the decay rate of the mist as Quadratic, Linear, and Square Root. These settings control the law which governs the strength of the mist as you move further away from the camera.

The mist begins at a distance from the camera as defined by the Sta: button, and is computed over a distance defined by the Di: button. Objects further from the camera than Sta+Di are completely hidden by the mist.

By default, the mist covers all of the image uniformly. To produce a more realistic effect you might want to have the mist decrease with height (altitude, or z) using the Hi: NumButton. If the value of this button is non-zero it sets, in Blender units, an interval, around z=0 in which the mist goes from maximum intensity (below) to zero (above).

Finally, the Misi: NumButton defines the intensity, or strength, of the mist.

Figure 13-5 shows a possible test set up.



#### Figure 13-5. Mist test setup

Figure 13-6 shows the results with and without mist. The settings are shown in Figure 13-7; the texture is a plain procedural cloud texture with Hard noise set.



Figure 13-6. Rendering without mist (left) and with mist (right).



Figure 13-7. World set up.

**Mist distances:** To see what the mist will actually affect, select your camera, go to Editing Context (**F9**) and press the show Mist TogButton in the Camera Panel. The camera will show mist limits as a segment projecting from the camera starting from sta and of distance Di.

## Stars

Stars are randomly placed halo-like objects which appear in the background. The Stars settings are on the right-hand side of the Mist Stars Physics Panel (Figure 13-8).

Mist			Stars
Qua	Lin	Sqr	
4	Sta: 0.00	) 🕨	<ul> <li>StarDist: 15.00</li> </ul>
4	Di: 0.00	Þ	<ul> <li>MinDist: 0.00</li> </ul>
4	Hi: 0.00	⊬	Size:2.00
MisiC	.000		Colnoise:

#### Figure 13-8. Star buttons

When creating stars, you need to understand a few important concepts:

StarDist: is the *average* distance between stars. Stars are intrinsically a 3D feature, they are placed in space, not on the image!

MinDist: Is the *minimum* distance from the camera at which stars are placed. This should be greater than the distance from the camera to the *furthest* object in your scene, unless you want to risk having stars *in front* of your objects.

The Size: NumButton defines the actual size of the star halo. It is better to keep it much smaller than the proposed default, to keep the material smaller than pixel-size and have pin-point stars. Much more realistic.

The Colnoise: NumButton adds a random hue to the otherwise plain white stars. It is usually a good idea to add a little ColNoise.

Figure 13-9 shows the same misty image of Figure 13-7 but with stars added. The Stars settings used for the image are shown in Figure 13-10.



Figure 13-9. Star rendering.

<ul> <li>Mist Stars Physics</li> </ul>				
© Grav 9.80				
Mist	Stars			
Qua Lin Sqr				
<ul> <li>≤ Sta: 1.00</li> </ul>	🔹 Star Dist: 3.00 🔹			
✓ Di: 19.60 →	✓ MinDist: 50.00 →			
	Size:0.100			
Misi0.200	Colnoise:0 I			

Figure 13-10. Star settings.

## **Ambient Occlusion**

Ambient Occlusion is a sophisticated ambient trick which simulates soft global illumination by taking into account the amount of sky (which is assumed to be the lightsource) seen by a given point.

This is actually done by casting rays from each visible point, and by counting how many of them actually reach the sky, and how many, on the other hand, are obstructed by objects. The amount of light on the point is then proportional to the number of rays which have 'escaped' and have reached the sky.

This is done by firing a hemisphere of shadow-rays around. If a ray hits another face (it is occluded) then that ray is considered 'shadow', otherwise it is considered 'light'. The ratio between 'shadow' and 'light' rays defines how bright a given pixel is.

Ambient Occlusion (AO) settings are in the Shading Context, World Buttons Subcontext, in the Amb Occ Tab. AO is Off by default, if it is turned On, the Tab is populated by many buttons (Figure 13-11).

V Mist / Stars	s / Physi	Amb	Occ			
Ambient Occlusion						
<ul> <li>Samples:</li> </ul>	5 🕨	Rand	lom Sampling			
<ul> <li>Dist: 10.0</li> </ul>	0 🕨					
Use Distan	ces					
Add Su		ub	Both			
Plain	Plain Sky		Sky Texture			
Energy:1.00		Bias:0.050				

Figure 13-11. Ambient Occlusion Panel.

Rays are shot at the hemisphere according to a random pattern, this causes sensible differences in the occlusion pattern of neighbouring pixels unless the number of shot rays is big enough to produce good statistical data. This is why AO produces a granular pattern, which looks like dirt, if there are not enough rays. The number of shot rays is controlled via the Samples NumButton. The default value of 5 is usually good for previews. The actual amount of shot rays is the square of this number. (i.e. Samples=5 means 25 rays). Figure 13-12 shows a simple scene, with increasing number of samples. Of course rendering time increases as the number of samples increases!



Figure 13-12. Effect of the different number of samples.

The Dist and Use Distances Buttons allow for subtle control over shadowing by defining a distance dependent behaviour and damping in the occlusion.

The row of radio buttons Add, Sub and Both controls the occlusion behaviour:

- Add The pixel receives light according to the number of non-obstructed rays. The scene is lighter.
- Sub The pixel receives shadow (negative light) according to the number of obstructed rays. The scene is darker.

• Both - Both effects take place, the scene has more or less the same brightness.

**Note:** If sub is chosen then there *must* be some light source somewhere, otherwise the scene would be pitch black. In the other two cases the scene is lit even if no explicit light is present.

The row of radio buttons Plain, Sky Color and Sky Texture controls the light color:

- Plain The pixel receives pure white light according to the number of non-obstructed rays.
- Sky Color The pixel receives colored light, the color is computed on the basis of the portion of the sky hit by the non-obstructed rays (Figure 13-13).
- Sky Texture A Sky Image texture must be present, possibly an AngMap or a SphereMap. It behaves as Sky Color but the ray color depends on the color of the Sky texture pixel hit.



Figure 13-13. Ambient Occlusion with Sky Color. Zenith is blue, Horizon is orange, and type is Blend so that sky goes full orange at Nadir.

The Energy slider controls the actual amount of light/shadows the AO procedure creates.

Since AO occurs on the original faceted mesh, it is possible that the AO light makes faces visible even on objects with 'smooth' On. This is due to the way AO rays are shot, and can be controlled with the Bias Slider. The bias setting allows you to control how smooth 'smooth' faces will appear in AO rendering. The bias denotes the angle (in radians) the hemisphere will be made narrower. Values of 0.05 to 0.1 typically work well (Figure 13-14).



Figure 13-14. Ambient Occlusion bias values.

Please note that this is just raytracing, so it tends to be slow. Furthermore, performance severely depends on Octree size, see the Rendering Chapter for more information. Chapter 13. The World and The Universe

# **Chapter 14. Animation of Undeformed Objects**

Objects can be animated in many ways. They can be animated as Objects, changing their position, orientation or size in time; they can be animated by deforming them; that is animating their vertices or control points; or they can be animated via very complex and flexible interaction with a special kind of object: the Armature.

In this chapter we will cover the first case, but the basics given here are actually vital for understanding the following chapters as well.

Three methods are normally used in animation software to make a 3D object move:

- *Key frames* Complete positions are saved for units of time (frames). An animation is created by interpolating an object fluidly through the frames. The advantage of this method is that it allows you to work with clearly visualized units. The animator can work from one position to the next and can change previously created positions, or move them in time.
- *Motion Curves* Curves can be drawn for each XYZ component for location, rotation, and size. These form the graphs for the movement, with time set out horizontally and the value set out vertically. The advantage of this method is that it gives you precise control over the results of the movement.
- *Path* A curve is drawn in 3D space, and the Object is constrained to follow it according to a given time function of the position along the path.

The first two systems in Blender are completely integrated in a single one, the *IPO* (InterPOlation) system. Fundamentally, the IPO system consists of standard motion curves. A simple press of a button changes the IPO to a key system, without conversion, and with no change to the results. The user can work any way he chooses to with the keys, switching to motion curves and back again, in whatever way produces the best result or satisfies the user's preferences.

The IPO system also has relevant implication in Path animations.

## **IPO Block**

#### Relevant to Blender v2.31

The IPO block in Blender is universal. It makes no difference whether an object's movement is controlled or the material settings. Once you have learned to work with object IPOs, how you work with other IPOs will become obvious. Anyway Blender does distinguish between different *types* of IPOs and the interface keeps track of it automatically.

Every type of IPO block has a fixed number of available *channels*. These each have a name (LocX, SizeZ, etc.) that indicates how they are applied. When you add an IPO Curve to a channel, animation begins immediately. At your discretion (and there are separate channels for this), a curve can be linked directly to a value (LocX...), or it can affect a variance of it (dLocX...). The latter enables you to move an object as you would usually do, with the Grabber, without disrupting the IPO. The actual location is then determined by IPO Curves *relative* to that location.

The Blender interface offers many options for copying IPOs, linking IPOs to more than one object (one IPO can animate multiple objects.), or deleting IPO links. The IPO Window Reference section gives a detailed description of this. This chapter is restricted to the main options for application.

## **Key Frames**

Relevant to Blender v2.31

Insert Key
Loc
Rot
Size
LocRot
LocRotSize
Layer
Avail

Figure 14-1. Insert Key Menu.

The simplest method for creating an object IPO is with the "Insert key" (**IKEY**) command in the 3DWindow, with an Object selected. A Pop-up menu provides a wide selection of options (Figure 14-1). We will select the topmost option: Loc. Now the current location X-Y-Z, is saved and everything takes place automatically:

- If there is no IPO block, a new one is created and linked to the object.
- If there are no IPOCurves in the channels LOCX, LOCY and LOCZ, these are created.
- Vertices are then added in the IPOCurves with the exact values of the object location.

We go 30 frames further on (3 x **UPARROW**) and move the object. Again we use **IKEY**. Now we can immediately press **ENTER** since Blender remembers our last choice and will highlight it. The new position is inserted in the IPO Curves. We can see this by slowly paging back through the frames (**LEFTARROW**). The object moves between the two positions.

In this way, you can create the animation by paging through the frames, position by position. Note that the location of the object is *directly* linked to the curves. When you change frames, the IPOs are always re-evaluated and re-applied. You can freely move the object within the same frame, but as soon as you change frame, the object 'jumps' to the position determined by the IPO.

The rotation and size of the object are completely free in this example. They can be changed or animated with the Insert key procedure by selecting from the **IKEY** menu the other options such as Rotation, Size and any combination of these.

# The IPO Curves

Relevant to Blender v2.31



Figure 14-2. The IPO window.

Now we want to see exactly what happened. The first Screen initialised in the standard Blender start-up file is excellent for this. Activate it with **CTRL-LEFTARROW**. At the right we see the IPOWindow displayed (Figure 14-2). You can of course turn any window into an IPO window with the pertinent Window Type menu entry, but it is more handy to have both a 3D window and an IPO window at the same time. This shows all the IPO Curves, the channels used and those available. You can zoom in and out the IPO Window and translate it just as every other Blender Window.

In addition to the standard channels, which can be set via **IKEY**, you have the *delta* options, such as dLocX. These channels allow you to assign a *relative* change. This option is primarily used to control multiple objects with the same IPO. In addition, it is possible to work in animation 'layers'. You can achieve subtle effects this way without having to draw complicated curves.

Each curve can be selected individually with the **RMB**. In addition, the Grabber and Size modes operate here just as in the 3DWindow. You can select IPOs also by clicking the color button in the right channel names column. By clicking the IPO channel name you effectively hide/show the relative curve. Selecting all curves (**AKEY**) and moving them to the right (**GKEY**), you can move the complete animation in time.

Each curve can be placed in EditMode individually, or it can be done collectively. Select the curves and press **TAB**. Now the individual vertices and *handles* of the curve are displayed. The Bézier handles are coded, like it is in the Curve Object:

- Free Handle (black). This can be used any way you wish. Hotkey: **HKEY** (switches between Free and Aligned).
- Aligned Handle (pink). This arranges all the handles in a straight line. Hotkey: **HKEY** (toggles between Free and Aligned).
- Vector Handle (green). Both parts of a handle always point to the previous or next handle. Hotkey: **VKEY**.
- Auto Handle (yellow). This handle has a completely automatic length and direction. Hotkey: **SHIFT-HKEY**.

Handles can be moved by first selecting the middle vertex with **RMB**. This selects the other two vertices as well. Then immediately start the Grab mode with **RMB**-

#### Chapter 14. Animation of Undeformed Objects

hold and move. Handles can be *rotated* by first selecting the end of one of the vertices and then use the Grabber by means of the **RMB**-hold and move action.

As soon as handles are rotated, the type is changed automatically:

- Auto Handle becomes Aligned.
- Vector Handle becomes Free.

"Auto" handles are placed in a curve by default. The first and last Auto handles always move horizontally, which creates a fluid interpolation.

The IPOCurves have an important feature that distinguishes them from normal curves: it is impossible to place more than one curve segment horizontally. Loops and circles in an IPO are senseless and ambiguous. An IPO can only have 1 value at a time. This is automatically detected in the IPOWindow. By moving part of the IPOCurve horizontally, you see that the selected vertices move 'through' the curve. This allows you to duplicate parts of a curve (**SHIFT-D**) and to move them to another time frame.

It is also important to specify how an IPOCurve must be read *outside* of the curve itself. There are four options for this in the Curve>>Extend Mode Submenu in the IPO Window header (Figure 14-3).

Insert Keyframe	I		
Record Mouse Movemen	t R		
Duplicate	Shift D	Constant	
Delete	Х	Extrapolation	
Interpolation Mode	•	Cyclic	
Extend Mode	Þ	Cyclic Extrap	olation
Transform Properties	N	0 200	250
Curve 🗶 Object 🗢	2 🗢	IP:Oblpo	2 🗙 F

#### Figure 14-3. IPO extension options.

The effect of each of these can be appreciated in (Figure 14-4).



Figure 14-4. Extended IPOs.

From left to right:

#### **Extend mode Constant:**

The ends of selected IPOCurves are continuously (horizontally) extrapolated. It is the default behaviour.

#### Extend mode Extrapolation:

The ends of the selected IPOCurves continue in the direction in which they ended.

#### **Extend mode Cyclic:**

The complete width of the IPOCurve is repeated cyclically.

#### **Extend Mode Cyclic Extrapolation:**

The complete width of the IPOCurve is extrapolated cyclic.

In addition to Béziers, there are two other possible types for IPOCurves. Use the **TKEY** command, and the dialog which then pops-up, or the Curve>>Interpolation Mode submenu entry to select them. The interpolation of the selected IPOCurves can be set to:

- Constant after each vertex of the curve, this value remains constant. No interpolation takes place.
- Linear linear interpolation occurs between the vertices.
- Bezier the standard fluid interpolation.

The IPO curves need not be set only by Key Framing. They can also be drawn 'by hand'. Use the **CTRL-LMB** command. Here are the rules:

#### There is no IPO block yet (in this window) and one channel is selected:

a new IPOBlock is created along with the first IPOCurve with one vertex placed where the mouse was clicked.

#### There is already an IPO block, and a channel is selected without an IPOCurve:

a new IPOCurve with one vertex is added.

# There is already an IPO block, and a *channel* is selected with an existing IPOCurve:

A new point is added to the selected IPOCurve.

This is *not* possible if multiple IPOCurves are selected or in EditMode.

**Make an object rotate:** This is the best method for specifying axis rotations quickly: Select the object; in the IPOWindow, press one of the "Rot" channels and use **CTRL-LMB** to insert two points. If the axis rotation must be continuous, you must use the Curve>>Extend Mode>>Extrapolation Menu entry.

One disadvantage of working with motion curves is that the *freedom* of transformations is limited. You can work quite intuitively with motion curves, but only if this can be processed on an XYZ basis. For a location, this is outstanding, but for a size and rotation there are better mathematical descriptions available: matrices (3x3 numbers) for size and quaternions (4 numbers) for rotation. These could also have been processed in the channels, but this can quite easily lead to confusing and mathematically complicated situations.

Limiting the *size* to the three numbers XYZ is obvious, but this limits it to a rectangular distortion. A diagonal scaling such as 'shearing' is impossible. Simply working in hierarchies can solve this. A *non*-uniform scaled Parent will influence the rotation of a Child as a 'shear'.

The limitation of the three number XYZ rotations is less intuitive. This so-called Euler rotation is not uniform - the same rotation can be expressed with different numbers - and has the bothersome effect that it is *not* possible to rotate from any position to another, the infamous *gimbal lock*. While working with different rotation keys, the user may suddenly be confronted with quite unexpected interpolations, or it may turn out to be impossible to force a particular axis rotation when making manual changes. Here, also, a better solution is to work with a hierarchy. A Parent will *always* assign the specified axis rotation to the Child. (It is handy to know that the X, Y and Z rotations are calculated one *after* the other. The curve that affects the RotX channel, *always* determines the X axis rotation).

Luckily, Blender calculates everything internally with matrices and quaternions. Hierarchies thus work normally, and the Rotate mode does what you would expect it to. Only the IPOs are a limitation here, but in this case the ease of use prevails above a not very intuitive mathematical purity.

## **IPO Curves and IPO Keys**

Relevant to Blender v2.31

The easiest way to work with motion curves is to convert them to IPO Keys. We return to the situation in the previous example: we have specified two positions in an object IPO in frame 1 and frame 31 with **IKEY**. At the right of the screen, you can see an IPO Window. We set the current frame to 21 (Figure 14-5).



Figure 14-5. The IPOKey mode.

Press **KKEY** while the mouse cursor is in the 3DWindow. Two things will happen now:

- The IPOWindow switches to IPOKey mode.
- The selected object is assigned the "DrawKey" option.

The two actions each have separate meanings.

- The IPOWindow now draws vertical lines through all the vertices of all the visible IPOCurves (IPOS are now black). Vertices with the same 'frame' value are linked to the vertical lines. The vertical lines (the "IPOKeys") can be selected, moved or duplicated, just like the vertices in EditMode. You can translate the IPOKeys only horizontally.
- The object is not only shown in its current position but 'ghost' objects are also shown at all the Key positions. In addition to now being able to visualize the key positions of the object, you can also modify them *in* the 3DWindow. In this example, use the Grab mode on the object to change the *selected* IPOKeys.

Below are a number of instructions for utilizing the power of the system:

- You can only use the **RMB** to select IPOKeys in the IPOWindow. Border select, and *extend* select, are also enabled here. Select all IPOKeys to transform the complete animation system in the 3DWindow.
- The "Insert Key" always affects *all* selected objects. The IPOKeys for multiple objects can also be transformed simultaneously in the 3DWindow. Use the **SHIFT-**K command: Show and select all keys to transform complete animations of a group of objects all at once.
- Use the **PAGEUP** and **PAGEDOWN** commands to select subsequent keys in the 3DWindow.
- You can create IPOKeys with each arrangement of channels. By consciously *exclud-ing* certain channels, you can force a situation in which changes to key positions in the 3DWindow can only be made to the values specified by the visible channels. For example, with only the channel LocX selected, the keys can only be moved in the X direction.
- Each IPOKey consists of the vertices that have *exactly* the same frame value. If vertices are moved manually, this can result in large numbers of keys, each having only one curve. In this case, use the **JKEY** ("Join") command to combine selected IPOKeys. It is also possible to assign selected IPOKeys vertices for *all* the visible curves: use **IKEY** in the IPOWindow and choose "Selected keys".
- The DrawKey option and the IPOKey mode can be switched on and off independently. Use the button EditButtons->DrawKey to switch off this option or object. You can switch IPOKey mode on and off yourself with **KKEY** in the IPOWindow. Only **KKEY** in the 3DWindow turns on/off both the DrawKey and IPOKey mode.

## Other applications of IPO Curves

Relevant to Blender v2.31

There are several other application for IPOs other than just animating an Object movement.

The IPO Type Menu Buttons in the header (Figure 14-6) allow IPO Block type selection, the active one there is the Object IPO described up to now, but there are Material IPO, World IPO, Vertex Keys IPO, Constraints IPO and Sequence IPO. Not every entry is always present, depending on context. Curve IPO block appears if the selected object is a curve and not a Mesh, the Lamp IPO only appears if the selected object is a lamp.

lpo type:				
55	Sequence			
٩	Constraint			
Q	Vertex			
۲	World			
۲	Material			
K,	Object			
Ľ,	Object 🗢			

Figure 14-6. The IPO window.

Material IPO is a way of animating a Material. Just as with objects, IPO Curves can be used to specify 'key positions' for Materials. With the mouse in the ButtonsWindow, the command **IKEY** calls up a pop-up menu with options for the various Material variables. If you are in a Material, Lamp or World IPO Block then a small Num Button appears next to the IPO type Menu in the IPO Window toolbar. This indicates which texture channel is active. The mapping for all 8 channels can be controlled with IPOCurves!

Strictly speaking, with textures two other animations are possible. Since Objects can give texture coordinates on other objects (Each object in Blender can be used as a source for texture coordinates. To do this, the option "Object" must be selected in the green "Coordinates input" buttons and the name of the object must be filled in. An inverse transformation is now performed on the global render coordinate to obtain the *local* object coordinate) it is possible to animate the texture simply by animating the location, size, and rotation of the object.

Furthermore, at each frame, Blender can be made to load another (numbered) Image as a texture map instead of having a fixed one. It is also possible to use SGI movie files or AVI files for this.

# The Time Ipo

#### Relevant to Blender v2.31

With the TimeIpo curve you can manipulate the animation time of objects without changing the animation or the other Ipos. In fact, it changes the mapping of animation time to global animation time (Figure 14-7).



Figure 14-7. Linear time IPO

To grasp this concept, make a simple keyframe-animation of a moving object, from a position to another in, say, 50 frames. Then select the Time channel and create a TimeIpo in the IpoWindow going from point (1,1) to point (50,50). It is easy to set the start and end point of an IPO by using **NKEY** and entering the values numerically.

In frames where the slope of the TimeIpo is positive, your object will advance in its animation. The speed depends on the value of the slope. A slope bigger than 1 will animate faster than the base animation. A slope smaller than 1 will animate slower. A slope of 1 means no change in the animation, negative power slopes allow you to reverse the animation.

The TimeIpo is especially interesting for particle systems, allowing you to "freeze" the particles or to animate particles absorbed by an object instead of emitted. Other possibilities are to make a time lapse or slow motion animation.

**Multiple Time IPOs:** You need to copy the Timelpo for every animation system to get a full slow motion. But by stopping only some animations, and continue to animate, for example, the camera you can achieve some very nice effects (like those used to stunning effect in the movie "The Matrix")

## **Path Animation**

Relevant to Blender v2.31

A different way to have Objects move in the space is to constrain them to follow a given path.

When objects need to follow a path, or it is too hard to animate a special kind of movement with the keyframe method (Think of a planet following its way around the Sun. Animating that with keyframes is virtually impossible) curve objects can be used for the 3D display of an animation path.

If the Curve object contains more than a single continuous curve only the first curve in the object is then used.

CU:Curve.001	F OB:Curve		UV Orco	✓ DefResolU: 6 → Set
	0.M	at 0 2	Centre	Back Front 3D
	New Delete		Centre New	✓ Width: 1.000 →
			Centre Cursor	Ext1: 0.000
	Select	Deselect	✓ PathLen: 100 →	
	Ass	sign	CurvePath CurveFoll	BevResol: 0 >
AutoTexSpace	Set Smooth	Set Solid	PrintLen 0.0000	BevOb:

Figure 14-8. The Action Window with Path Buttons.

As for tracking, there are two Path animation methods, the old, pre 2.30 method described here and the new method, which actually defines a constraint, which will be described in the Section called *Constraints* in Chapter 16. When parenting an Object to a Curve you will be asked to choose a Normal Parent or a Follow Path option. The Former is what you need for conventional Path animation, but other actions needs to be taken later. The second option creates a "Follow Path" Constraint, and it is all you need to do.

Any kind of curve can become a path by setting the option CurvePath Toggle Button in the Animation Buttons window (F7) to ON (Figure 14-8).

When a Curve has childs it can be turned to a Path by selecting it, going int the Editing Context (F9) and activating the CurvePath Toggle button in the Curve and Surface Panel. Child objects of the Curve will now move along the specified path. It is a good idea to set the Curve to 3D via the 3D Toggle Button of the Curve Edit Buttons so that the paths can be freely modelled.

Otherwise, in the ADD menu under Curve->Path, there is a primitive with the correct settings already there. This is a 5th order NURBS spline, which can be used to create very fluid, continuous movements.

Normally a Path is 100 frames long and it is followed in 100 frames by children. You can make it longer or shorter by varying the PathLength: Num Button.

The *speed* along a path is determined with an appropriate curve in the IPO Window. To see it, in the IPO Window Header button you must select the Curve type for the IPO block. A single channel, Speed is there. The complete path runs in the IPO Window between the vertical values 0.0 and 1.0. Drawing a curve between these values links the time to the position on the path. Backward and pulsing movements are possible with this. For most paths, an IPO Curve must run *exactly* between the Y-values 0.0 and 1.0. To achieve this, use the Number menu (**NKEY**) in the IPO Window. If the IPO Curve is deleted, the value of PathLen determines the duration of the path. A linear movement is defined in this case. The Speed IPO is a finer way of controlling Path length. The path is long 1 for time IPO, and if the time IPO goes from 0 to 1 in 200 frames then the path is 200 frames long.

Using the option CurveFollow, in the Curve and Surface Panel, a rotation is also given to the Child objects of the path, so that they permanently point in the direction of the path. Use the "tracking" buttons in the Anim settings Panel of the Object (F7) context to specify the effect of the rotation (Figure 14-9) as you would do for Tracking:

▼ Anim settings								
TrackX Y Z - X - Y - Z UpX Y Z								
Draw Key Draw Key Sel Powertrack SlowPar								
DupliFrames DupliVerts Rot No Speed								
🔹 DupSta: 1 🔹 DupOn: 1 🕨								
✓ DupEnd 100 ►								
Offs Ob Offs Par Offs Particle 0.0000								

#### **Figure 14-9. Tracking Buttons**

TrackX, Y, Z, -X, -Y, -Z This specifies the direction axis, i.e. the axis that is placed on the path.

UpX, UpY, UpZ Specifies which axis must point 'upwards', in the direction of the (local) positive Z axis. If the Track and the Up axis coincides, it is deactivated.

**Note:** Curve paths have the same problem of Bevelled curves for what concern the definition of the "Up" direction.

To visualize these rotations precisely, we must make it possible for a Child to have its own rotations. Erase the Child's rotation with **ALT-R**. Also erase the "Parent Inverse": **ALT-P**. The best method is to 'parent' an *unrotated* Child to the path with the command **SHIFT-CTRL-PKEY**: "Make parent without inverse". Now the Child jumps directly to the path and the Child points in the right direction.

3D paths also get an extra value for each vertex: the 'tilt'. This can be used to specify an axis rotation. Use **TKEY** in EditMode to change the tilt of selected vertices in EditMode, e.g. to have a Child move around as if it were on a roller coaster.

Figure 14-10 shows a complex application. We want to make a fighter dive into a canyon, fly next to the water and then rise again, all this by following it with our camera and, possibly, having reflection in the water!

To do this we will need three paths. Path 1 has a fighter parented to it, the fighter will fly following it.



Figure 14-10. Complex path animation

The fighter has an Empty named 'Track' Parented to it in a strategic position. A camera is then parented to another curve, Path 2, and follows it, tracking the 'Track' Empty. The Fighter has a constant Speed IPO, the camera has not. It goes faster, then slower, always tracking the Empty, and hence the fighter, so we will have very fluid movements of the camera from Fighter side, to Fighter front, other side, back, etc. (Figure 14-11)



Figure 14-11. Some frames, the camera fluidly tracking the fighter.

Since we want our fighter to fly over a river, we need to set up an Env Map for the water surface to obtain reflections. But the Empty used for the calculations must be always in specular position with respect to the camera... and the camera is moving along a path!

Path 3 is hence created by mirroring path 2 with respect to the water plane, by duplicating it, and using **MKEY** in Edit Mode with respect to the cursor, once the cursor is on the plane.

The Empty for the Env Map calculation is then parented to this new path, and the Time IPO of Path 2 is copied to Path 3. Figure 14-12 shows a rendered frame. Some particle systems were used for trails.

The scene presents many subtle tricks, as particles for the jet streams, fog, a sky sphere encircling the scene and so on.



Figure 14-12. A frame of the final animation.

Chapter 14. Animation of Undeformed Objects

# **Chapter 15. Animation of Deformations**

Animating an Object/Material, is not the only thing you can do in Blender. You can change, reshape, deform your objects in time!

There are many ways of achieving this actually, and one technique is so powerful and general there is a full chapter for it: Character animation. The other techniques will be handled here.

# **Absolute Vertex Keys**

#### Relevant to Blender v2.31

VertexKeys (as opposed to Object keys, the specified positions of objects) can also be created in Blender; VertexKeys are the specified positions of vertices *within* an Object. Since this can involve thousands of vertices, separate motion curves are not created for each vertex, the traditional Key position system is used instead. A single IPOCurve is used to determine how interpolation is performed and the times at which a VertexKey can be seen.

VertexKeys are part of the Object Data, not of the Object. When duplicating the Object Data, the associated VertexKey block is also copied. It is not possible to permit multiple Objects to share the same VertexKeys in Blender, since it would not be very practical.

The Vertex Key block is universal and understands the distinction between a Mesh, a Curve, a Surface or a Lattice. The interface and use is therefore unified. Working with Mesh VertexKeys is explained in detail in this section, which also contains a number of brief comments on the other Object Data.

The first VertexKey position that is created is always the *reference* Key. This key defines the texture coordinates. Only if this Key is *active* can the faces and curves, or the *number* of vertices, be changed. It is allowed to assign other Keys a different number of vertices. The Key system automatically interpolates this.

A practical example is given below. When working with VertexKeys, it is very handy to have an IPO Window open. Use the first Screen from the standard Blender file, for example. In the IPO Window, we must then specify that we want to see the VertexKeys. To do this use the IPO type Menu Button and select Vertex. Go to the 3DWindow with the mouse cursor and press **IKEY**. With a Mesh object selected and active. The "Insert Key" menu has several options, the latter being Mesh. As soon as this has been selected, a new dialog appears (Figure 15-1) asking for Relative or Absolute Vertex Keys.

ĺ	Insert Vertex Keys								
I	F	Relative keys							
	Α	lbso	lute	key	/s				
	A	\bso	lute	key	/s	_	_	_	

#### Figure 15-1. Insert Vertex Keys Menu.

We will choose Absolute Keys; a yellow horizontal line is drawn in the IPO Window. This is the first key and thus the *reference* Key. An IPO Curve is also created for "Speed" (Figure 15-2).



Figure 15-2. Reference Key and Speed IPO.

**Vertex Key creation:** Creating VertexKeys in Blender is very simple, but the fact that the system is very sensitive in terms of its configuration can cause a number of 'invisible' things to happen. The following rule must therefore be taken into consideration.

As soon as a VertexKey position is inserted it is *immediately active*. All subsequent changes in the Mesh are linked to *this* Key position. It is therefore important that the Key position be added *before* editing begins.

Go a few frames further and again select: **IKEY**, Mesh (in the 3DWindow). The second Key is drawn as a light blue line. This is a normal Key; this key and all subsequent Keys affect only the vertex information. Press **TAB** for EditMode and translate one of the vertices in the Mesh. Then browse a few frames back: nothing happens! As long

as we are in EditMode, other VertexKeys are *not* applied. What you see in EditMode is *always* the *active* VertexKey.

Leave EditMode and browse through the frames again. We now see the effect of the VertexKey system. VertexKeys can only be selected in the IPO Window. We always do this *out* of Edit Mode: the 'contents' of the VertexKey are now temporarily displayed in the Mesh. We can edit the specified Key by starting Editmode.

There are three methods for working with Vertex Keys:

- The 'performance animation' method. This method works entirely in EditMode, chronologically from position to position:
  - Insert Key. The reference is specified.
  - A few frames further: Insert Key. Edit the Mesh for the second position.
  - A few frames further: Insert Key. Edit the Mesh for the third position.
  - Continue the above process...
- The 'editing' method.
  - We first insert all of the required Keys, unless we have already created the Keys using the method described above.
  - Blender is *not* in EditMode.
  - Select a Key. Now start EditMode, change the Mesh and leave EditMode.
  - Select a Key. Start EditMode, change the Mesh and leave EditMode.
  - Continue the above process....
- The 'insert' method
  - Whether or not there are already Keys and whether or not we are in EditMode does not matter in this method.
  - Go to the frame in which the new Key must be inserted.
  - Insert Key.
  - Go to a new frame, Insert Key.
  - Continue the above process...

While in EditMode, the Keys cannot be switched. If the user attempts to do so, a warning appears.

Each Key is represented by a line which is drawn at a given height. Height is chosen so that the key intersects the "Speed" IPO at the frame at which the Key is taken.

Both the IPO Curve and the VertexKey can be separately selected with **RMB**. Since it would otherwise be too difficult working with them, selection of the Key lines is switched off when the curve is in Edit Mode. The *channel* button can be used to temporarily hide the curve (**SHIFT-LMB** on "Speed") to make it easier to select Keys.

The Key lines in the IPO Window, once taken, can be placed at any vertical position. Select the line and use Grab mode to do this. The IPO Curve can also be processed here in the same way as described in the previous chapter. Instead of a 'value', how-ever, the curve determines the interpolation between the Keys, e.g. a sine curve can be used to create a cyclical animation.

During the animation the frame count gives a certain value of the speed IPO, which is used to chose the Key(s) which is/are to be used, possibly with interpolation, to produce the deformed mesh.

#### Chapter 15. Animation of Deformations

The Speed IPO has the standard behaviour of an IPO, also for interpolation. The Key line has three different interpolation types. Press **TKEY** with a Key line selected to to open a menu with the options:

- Linear: interpolation between the Keys is linear. The Key line is displayed as a dotted line.
- Cardinal: interpolation between the Keys is fluid, the standard setting.
- BSpline: interpolation between the Keys is extra fluid and includes four Keys in the interpolation calculation. The positions are no longer displayed precisely, however. The Key line is drawn as a dashed line.

Figure 15-3 shows a simple Vertex Key animation of a cylinder. When run the cylinder deforms to a big star, then deforms to a small star, then, since the Speed IPO goes back to 0 the deformation is repeated in reverse order.



Figure 15-3. Absolute Keys.

Some useful tips:

- Key positions are *always* added with IKEY, even if they are located at the same position. Use this to copy positions when inserting. Two key lines at the same position can also be used to change the effect of the interpolation.
- If *no* Keys are selected, EditMode can be invoked as usual. However, when you leave EditMode, all changes are undone. Insert the Key *in* EditMode in this case.
- For Keys, there is *no* difference between *selected* and *active*. It is therefore not possible to select multiple Keys.
- When working with Keys with differing numbers of vertices, the faces can become disordered. There are no tools that can be used to specify *precise* sequence of vertices. This option is actually suitable only for Meshes that have only vertices such as Halos.

## **Curve and Surface Keys**

As mentioned earlier, Curve and Surface Keys work exactly the same way as Mesh Keys. For Curves, it is particularly interesting to place Curve Keys in the bevel object. Although this animation is *not* displayed real-time in the 3DWindow, it will be rendered.
## Lattice Keys

As soon as one Key is present in a Lattice, the buttons that are used to determine the resolution are blocked.

# **Relative VertexKeys**

Relevant to Blender v2.31

Relative Vertex Keys (RVK) works differently inasmuch only the difference between the reference mesh and the deformed mesh is stored. This allows for blending several keys together to achieve complex animations.

We will walk through RVK via an example.

We will create a facial animation via RVK. While Absolute Vertex Keys are controlled with only *one* IPO curve, Relative Vertex Keys are controlled by one interpolation curve for every key position, which states 'how much' of that relative deformation is used to produce the deformed mesh. This is why relative keys can be mixed (added, subtracted, etc.).

For facial animation, the base position might be a relaxed position with a slightly open mouth and eyelids half open. Then keys would be defined for left/right eyeblink, happy, sad, smiling, frowning, etc.

The trick with relative vertex keys is that only the vertices that are changed between the base and the key affect the final output during blending. This means it is possible to have several keys affecting the object in different places all at the same time.

For example, a face with three keys: smile, and left/right eye-blink could be animated to smile, then blink left eye, then blink right eye, then open both eyes and finally stop smiling - all by blending 3 keys. Without relative vertex keys 6 vertex keys would have needed to be generated, one for each target position.

Consider the female head in Figure 15-4:



Figure 15-4. The female head we want to animate.

### Chapter 15. Animation of Deformations

To add an RVK just press **IKEY** and select Mesh as for AVK, but, from the pop up menu select Relative Vertex Keys. This stores the reference Key which will appear as an yellow horizontal line in the IPO window.

Relative keys are defined by inserting further vertex keys. Each time the **IKEY** is pressed and Mesh selected a new horizontal line appears in the IPO window. If frame number is augmented each time the horizontal lines are placed one above the other. For easier modelling let's hide all vertices except those of the face (Figure 15-5).



Figure 15-5. All but the face vertices hidden.

Now move to another frame, say number 5, and add a new Key. A cyan line will appear above the yellow, which now turns orange. Switch to Edit mode and close the left eyelid.

When you are done exit from Edit Mode. If you select the reference key you will see the original mesh. If you select your first RVK you will see the deformed one (Figure 15-6).



Figure 15-6. Left eye closed.

Repeat the step for the right eye. Beware that the newly inserted key is based on the mesh of the currently *active* key, so it is generally a good idea to select the reference key before pressing **IKEY**.

Then add a smile (Figure 15-7).

Chapter 15. Animation of Deformations



Figure 15-7. Smiling.

Your IPO Window will look like Figure 15-8.



Figure 15-8. Keys in the IPO Window.

The vertical order of the Vertex Keys (the blue lines) from bottom to top determines its corresponding IPO Curve, i.e. the lowest blue key line will be controlled by the Key1 curve, the second lowest will be controlled by the Key2 curve, and so on.

No IPO is present for the reference mesh since that is the mesh which is used if all other Keys have an IPO of value 0 at the given frame.

Select Key1 and add an IPO with your favourite method. Make it look like Figure 15-9.



Figure 15-9. The IPO curve of Key 1.

This will make our mesh undeformed up to frame 10, then from frame 10 to frame 20 Key 1 will begin to affect the deformation. From frame 20 to frame 40 Key 1 will completely overcame the reference mesh (IPO value is 1), and the eye will be completely closed. The effect will fade out from frame 40 to frame 50.

You can check with **ALT-A**, or by setting the frame numbers by hand. The second option is better, unless your computer is really powerful!

Copy this IPO by using the down pointing arrow button in the IPO Window toolbar (Figure 15-10). Select the  $Key_2$  and paste the curve with the up pointing arrow. Now both keys will have the same influence on the face and both eyes will close at the same time.



Figure 15-10. Clipboard buttons.

**Panning the Toolbar:** It may happen that the toolbar is longer than the window and some buttons are not shown. You can pan horizontally all toolbars by clicking **MMB** on them and dragging the mouse.

Add also an IPO for Key 3. Let's make this different (Figure 15-11).



Figure 15-11. All IPOs.

This way the eyes close and she begins to smile, smile is at maximum with eyes closed, then she smiles 'less' while the eyes re-open and keeps smiling (Figure 15-12).



### Figure 15-12. Sequence.

The IPO Curve for each key controls the blending between relative keys. These curves should be created in the typical fashion. The final position is determined by adding all of the effects of each individual IPO Curve.

**RVK in Action Window:** You can operate with RVK also in the Action (**SHIFT-F12**), not IPO, Window (Figure 15-13). The influence of any Key is given via a slider. Marks are present at Key points (i.e. where the IPO would have a control point).

Sliders	Þ								
Key 1			Ó	Ó		Ó	Ó		
Key 2			Ó	Ó		Ó	Ó		
Key 3			Ó				þ		
		U	10	20	30	40	50	60	
<b>X</b>									

Figure 15-13. RVK in Action Window.

Values out of [0,1] range: An important part of Relative Keys is the use of additive or extrapolated positions. For example, if the base position for a face is with a straight mouth, and a key is defined for a smile, then it is possible that the negative application of that key will result in a frown. Likewise, extending the IPO Curve above 1.0 will "extrapolate" that key, making an extreme smile.

# **Lattice Animation**

Relevant to Blender v2.31

Parenting a mesh to a lattice is a nice way to apply deformations to the former while modelling, but it is also a way to make deformations in time!

You can use Lattices in animations in two ways:

- Animate the vertices with vertex keys (or relative vertex keys);
- Move the lattice or the child object of the lattice.

The first technique is basically nothing new than what contained in the previous two sections but applied to a lattice which has an object parented to it.

With the second kind you can create animations that squish things between rollers, or achieve the effect of a well-known space ship accelerating to warp-speed.

Make a space ship and add a lattice around the ship. make the lattice with the parameters in Figure 15-14.



Figure 15-14. Lattice setup

Select the ship, extend the selection to the lattice (holding **SHIFT** while selecting), and press **CTRL-P** to make the lattice the parent of the ship. You should not see any deformation of the ship because the lattice is still regular.

For the next few steps it is important to do them in EditMode. So now select the lattice, enter EditMode, select all vertices (**AKEY**), and scale the lattice along its x-axis (press **MMB** while initiating the scale) to get the stretch you want. The ship's mesh shows immediately the deformation caused by the lattice (Figure 15-15).



### Figure 15-15. Stretching

Now edit the lattice in EditMode so that the right vertices have an increasing distance from each other. This will increase the stretch as the ship goes into the lattice. The right ends vertices I have scaled down so that they are nearly at one point; this will cause the vanishing of the ship at the end (Figure 15-16).

Select the ship again and move it through the lattice to get a preview of the animation. Now you can do a normal keyframe animation to let the ship fly through the lattice.



Figure 15-16. Final lattice deformation

**Camera tracking:** With this lattice animation, you can't use the pivot point of the object for tracking or parenting. It will move outside the object. You will need to vertex-parent an Empty to the mesh for that. To do so, select the Empty, then the mesh, enter EditMode and select one vertex, then press **CTRL-P**.

# Chapter 15. Animation of Deformations



Figure 15-17. Some frames of the resulting animation.

# **Chapter 16. Character Animation**

# Introduction: Lights, Camera and... ACTION !

As we have seen in the Section called *Rigging* in Chapter 4 Blender uses *Armatures* for character animation. An armature is just like a skeleton which once parented to our character mesh, will let us define a number of *poses* for our character along the timeline of our animation.

An armature is made up of an arbitrary number of *bones*. The size, position and orientation of every bone in your armature is up to you, and you will find through this chapter that different situations will require a particular arrangement of bones for your character to work properly.

As you animate your armature you will find that it is better to organize several related poses in something called an *action*, which is more or less the same as in the real world. When we walk, we can imagine ourselves passing through several instantaneous poses as if we were in the frames of a moving picture, the whole process of the walk is an action in the end. But there are actions and actions. As an animator you will need to acquire the capability of knowing how to split any natural movement or action into several simpler actions that will be easier to deal with. Working with simpler actions commonly saves time and work (and why not: money!) since these actions are usually reusable.

Once you have set-up your first actions you will be able to combine them using Blender's powerful *Non Linear Animation* (or *NLA*) editor, giving your character a living mood and natural manners.

In this chapter we will cover every single detail of Blender's functionalities related to Armatures, Actions and the NLA Editor. Furthermore we will see several armature set-ups that will give you a starting point for your own creations and ideas. Relax and enjoy.

# **General Tools**

Relevant to Blender v2.31

There are few Blender features which can make your life easier while animating a character. Let's see them, before going deep into details.

The auto-key feature can be found in the InfoWindow. When it is enabled, Blender will automatically set KeyFrames when you move Objects. This is helpful for people who are not used to explicitly inserting KeyFrames with **IKEY**. There are two separate toggles for auto-keying: one for Object Mode and one for Pose Mode. These two options can be set independently of one another from the Edit Method group of buttons in the User Preferences Window. (Figure 16-1).

	Material ObD	l linked Data	to: Object	Me t Sti	esh Undo eps:32	-	Auto keyfrar Action	ne on:	Object	
	Vie	ew & Co	ontrols	E E	Edit Metho	ds	Language & For	nt	Them	ne
i	🗢 File	Add	Timeline	Game	Render	Help	SCR:2-Model	×	¢ SCE:So	ce

### Figure 16-1. Auto key options

Auto Keyframe on Object will set KeyFrames for Objects that are moved in Object Mode. Users who are familiar with the Blender interface will likely want to leave this option disabled.

#### Chapter 16. Character Animation

Auto Keyframe on Action sets KeyFrames for transformations done in Pose Mode. This ensures that you will not lose a pose by forgetting to insert KeyFrames. Even users who are familiar with the Blender interface may find this to be a useful feature.

It is possible to display different IPOs in different windows. This is especially valuable while editing Actions, which have a different IPO for each bone.



Figure 16-2. Pinned Action IPOWindow

You can "pin" an IPO or Action (lock it to the current window) by pressing the pin icon in the header of the window (Figure 16-2). The contents of the window will stay there, even when the object is deselected, or another object is selected. Note that the colour of the IPO block menu will change, along with the background colour of the IPO Window. These serve as reminders that the window is not necessarily displaying the IPO of the currently selected object.

The browse menu is still available while a window is pinned. In this case however, changing the current data will not affect the current object; it merely changes which data is displayed.

## The Armature Object

### Relevant to Blender v2.31

The armature Object is the key Object of character animation. It is an object comprising several interconnected or not interconnected "bones". A series of interconnected bones is an "Inverse Kinematics (IK) Chain" or simply "Chain" of bones. An IK Chain is something more complex than a standard Parent relation inasmuch not only the movements of the "Parent" bone are transmitted to the children, but also the movements of the last child of the chain can transmit up in the chain to the parent bone if an Inverse Kinematics solution is asked for. Bones can be moved as if they were a set of rigid, undeformable Object with perfect joints. Consider an armature to be like a skeleton for a living creature. The arms, legs, spine and head are all part of the same skeleton object.

	Lattice	- 8
	Armatur	e
	Lamp	
	Camera	
	Empty	
	Text	
	MBall	
	Surface	- <b>F</b>
	Curve	•
	Mesh	- <b>-</b>
Object	Add	Select
Edit	Transform	View

Figure 16-3. Adding an Armature

To create a new armature, select **SPACE**>>Add>>Armature from the Toolbox (Figure 16-3). A new bone will appear with its root at the location of the 3D cursor. As you move the mouse, the bone will resize accordingly. **LMB** will finalize the bone and start a new one that is the child of the previous one. In this way you can make a complete chain. Pressing **ESC** will cancel the addition of the bone.

You can add another bone to an armature while it is in Edit Mode with **SPACE**>>Add>>Armature from the toolbox again. This will start the bone-adding mode again, and the new bones you create will be a part of the current armature but will form a separate chain.

You can also extrude bones from existing bones by selecting a bone joint and pressing **EKEY**. The newly created bone will be a child of the bone it is extruded from, but *not* of its IK chain.

While in Edit Mode, you can perform the following operations to the bones in an armature.

• *Adjusting* - Select one or more bone joints and use any of the standard transformation operations to adjust the position or orientation of any bones in the armature. Note that IK chains cannot have any gaps between their bones and as such moving the end point of a bone will move the start point, or *root* of its child.

You can select an entire IK chain at once by moving the mouse cursor over a joint in the chain and pressing **LKEY**. You can also use the boundary select tool (**BKEY**).

- *Deleting* You can delete one or more bones by selecting its start and end points. When you do this you will notice the bone itself will be drawn in a highlighted colour. Pressing **XKEY** will remove the highlighted bones. Note that selecting a single point is not enough to delete a bone.
- *Point Snapping* It is possible to snap bone joints to the grid or to the cursor by using the snap menu accessible with **SHIFT-S**.
- *Numeric Mode* For more precise editing, pressing **NKEY** will bring up the numeric entry box. Here you can adjust the position of the start and end points as well as the bone's roll around its own axis.

An easy way to automatically orient the z-axis handles of all selected bones (necessary for proper use of the pose-flipped option) is to press **CTRL-N**. Remember to do this before starting to create any animation for the armature. • *Undo* - While in Edit Mode, you can cancel the changes you have made in the current editing session by pressing **UKEY**. The armature will revert to the state it was in before editing began.

It is also possible to join two Armatures together into a single Object. To do this, ensure you are in Object Mode, select both armatures and press **CTRL-J**.

## **Naming Bones**

Assigning meaningful names to the bones in your armatures is important for several reasons. First it will make your life easier when editing Actions in the Action Window. Second, the bone names are used to associate Action channels with bones when you are attempting to re-use Actions, and third, the names are used when taking advantage of the automatic pose-flipping feature.

Note that bone names need only be unique within a given armature. You can have several bones called "Head" so long as they are all in different armatures.

To change the names of one or more bones, select the bones in Edit Mode and switch to the Editing Context Buttons with **F9**. A list of all the selected bones should appear in the Armature Bones Panel (Figure 16-4). Change a bone's name by **SHIFT-LMB** in the bone's name box and typing a new name.



#### Figure 16-4. EditButtons for an Armature

It is easier to name the bones by either only editing one bone at a time, or by making sure the DrawNames option is enabled in the EditButtons **F9** (Figure 16-5.

**Pose Flipping Conventions:** Character armatures are typically axially symmetrical. This means that many elements are found in pairs, one on the left and one on the right. If you name them correctly, Blender can flip a given pose around the axis of symmetry, making animation of walk-cycles much easier.

For every bone that is paired, suffix the names for the left and right with either ".L" and ".R" or ".Left" and ".Right". Bones that lie along the axis of symmetry or that have no twin need no suffix. Note that the part of the name preceding the suffix should be identical for both sides. So if there are two hands, they should be named "Hand.R" and "Hand.L".

### Parenting and IK chain

To change parenting relationships within the armature, select the bone that should be the *child* and switch to the Armature Bones Panel of the Edit Buttons Window. Next to the bone there should be a menu button labelled Child Of. To make the bone become the child of another bone, pick the appropriate parent from the list. Note that this is much easier if the bones have been correctly named. To dissolve a parenting relationship, choose the blank entry in the list.

Note that the parenting menu only contains the names of valid parents. Bones that cannot be parents (such as children of the current bone) will not be displayed.

The IK toggle next to each bone with a parent is used to determine if the IK solver should propagate its effects across this joint. If the IK button is active, the parent's end point will be moved to match its child's start point. This is to satisfy the requirement that there are no gaps in an IK chain. Deactivating the IK button will not restore the child's start point to its previous location, but moving the point will no longer affect the parent's end point.

**Note:** There can be only one IK relation between a Bone and it's child so only one of the IK Tog Buttons of the children of a given bone can be set at a time.

**Setting Local Axes:** To get the best results while animating, it is necessary to ensure that the local axes of each bone are consistent throughout the armature. This should be done before any animation takes place.

It is also necessary that when the armature object is in its untransformed orientation in object Mode, the front of the armature is visible in the front view, the left side is visible in the left view and so on. You can ensure this by orienting the armature so that the appropriate views are aligned and pressing **CTRL-A** to apply size and rotation. Again, this should be done before any animation takes place.

The orientation of the bones' *roll handles* is important to getting good results from the animation system. You can adjust the roll angle of a bone by selecting it and pressing **NKEY**. The exact number that must be entered here depends on the orientation of the bone.

The z-axis of each bone should point in a consistent direction for paired bones. A good solution is to have the z-axes point upwards (or forwards, when the bone is vertically oriented). This task is much easier if the "Draw Axes" option is enabled in the Armature Panel in the Edit Buttons Window.

## The Armature Panel

🔻 Armature		
	Rest Pos	
	Draw Axes	
	Draw Names	
	X-Ray	

Figure 16-5. Draw options for Armatures

This panel just contains a few toggle buttons. When the Rest Pos toggle is activated (Figure 16-5), the armature will be displayed in its rest position. This is useful if it becomes necessary to edit the mesh associated with an armature after some posing or animation has been done. Note that the Actions and poses are still there, but they are temporarily disabled while this button is pressed.

Draw Axes and Draw Names toggle buttons allow the local axes of each bone and its name to be displayed in the 3D Viewport.

The X-Ray toggle prevents the armature bones from being hidden by your model when in solid/shaded mode.

# Skinning

Relevant to Blender v2.31

Once the Armature - the 'character skeleton' - is ready it is necessary to parent the character 'skin' to it. Skinning is a technique for creating smooth mesh deformations with an armature. Essentially the skinning is the relationship between the vertices in a mesh and the bones of an armature, and how the transformations of each bone will affect the position of the mesh vertices.

When making a child of an armature, several options are presented:

Parent to Bone

In this case, a popup menu appears allowing you to choose which bone should be the parent of the child(ren) objects. This is great for robots, whose body parts are *separate* meshes which are not expected to bend and deform when moving.

#### Parent to Armature

Choosing this option will deform the child(ren) mesh(es) according to their vertex groups. If the child meshes don't have any vertex groups, they will be subject to automatic skinning. Indeed a second menu appears, asking:

- Don't create groups does nothing else, automatic skinning is used;
- Name Groups creates empty vertex groups whose names matches the bone names, but no vertices are assigned to them;
- Create from closest bone you want to create and populate automatically vertex groups.

#### Parent to Armature Object

Choosing this option will cause the child(ren) to consider the armature to be an Empty for all intents and purposes.

If you are going for character animation then most of the times you will parent your character to the Armature using the "Armature" Option. You are strongly advised to use the Name Groups option. This will provide you with the groups already created, saving the tedious operations of creating an naming them, and possibly avoiding typing errors.

The Create from closest bone feature is currently under heavy development. It will use the "Bone types" which can be defined via the menu right of the IK Tog Buttons (Figure 16-4) for optimal result.

Currently only the Skinnable and Unskinnable options are working. The first option makes Vertex Group be created (and populated, if this is asked for) for the given bone, the second option causes that bone to be ignored in the skinning process.

**Note:** The current vertex assignment algorithm creates non-optimal vertex groups, hence it is highly recommended to check each group, one by one.

If a mesh does not have any vertex groups, and it is made the child of an armature, Blender will attempt to calculate deformation information on the fly. This is very slow and is not recommended. It is advisable to create and use vertex groups instead.

Weight and Dist: The Weight and Dist settings next to the IK are only used by the automatic skinning which is a deprecated feature because it requires lot of CPU, produces slow downs and worse result than other methods.

## Vertex Groups

🔻 Link and	l Materials				
≑ ME:Arm		F OB:Arm			
Vertex Grou	Jps	Hand			
		🔹 1 Ma	at:1 ▶ ?		
< Weight	: 1.000 💿 🕨				
New	New Delete		Delete		
Assign	Assign Bemove		Deselect		
Select	Desel.	Assign			
AutoTe	xSpace	Set Smooth	Set Solid		

Figure 16-6. Vertex Groups

Vertex groups are necessary to define which bones deform which vertices. A vertex can be a member of several groups, in which case its deformation will be a weighted average of the deformations of the bones it is assigned to. In this way it is possible to create smooth joints.

To add a new vertex group to a mesh, you must be in Edit Mode. Create a new vertex group by clicking on the New button in the mesh's Edit Buttons Mesh Tools 1 Panel (Figure 16-6).

#### Chapter 16. Character Animation

A vertex group can be subsequently deleted by clicking on the Delete button.

Change the active group by choosing one from the pull-down group menu.

Vertex groups must have the *same* names as the bones that will manipulate them. Both spelling and capitalization matter. This is why automatic name creation is so useful! Rename a vertex group by **SHIFT-LMB** on the name button and typing a new name. Note that vertex group names must be unique within a given mesh.

Vertices can be assigned to the active group by selecting them and clicking the Assign button. Depending on the setting of the Weight button, the vertices will receive more or less influence from the bone. This weighting is only important for vertices that are members of more than one bone. The weight setting is not an absolute value; rather it is a relative one. For each vertex, the system calculates the sum of the weights of all of the bones that affect the vertex. The transformations of each bone are then divided by this amount meaning that each vertex always receives exactly 100% deformation.

Assigning 0 weight to a vertex will effectively remove it from the active group.

To remove vertices from the current group select them and click the Remove button.

Pressing the Select button will add the vertices assigned to the current group to the selection set. Pressing the Deselect button will remove the vertices assigned to the current group from the selection set. This is handy to check which vertices are in which group.

## Weight Painting

Weight painting is an alternate technique for assigning weights to vertices in vertex groups. The user can "paint" weights onto the model and see the results in real-time. This makes smooth joints easier to achieve.

To activate weight-painting mode, select a mesh with vertex groups and click on the weight paint icon (Figure 16-7).

Мо	de:	
Ø	Weight Paint	
C	Texture Paint	
Ì	Vertex Paint	
₽	UV Face Select	
۵	Edit Mode	
K.	Object Mode	
ť,	Object Mode 😑	4

Figure 16-7. Weight Paint Button.

The active mesh will be displayed in Weight-Colour mode. In this mode dark blue represents areas with no weight from the current group and red represent areas with full weight. Only one group can be visualized at a time. Changing the active vertex group in the Edit Buttons will change the weight painting display.

Weights are painted onto the mesh using techniques similar to those used for vertex painting, with a few exceptions. The "colour" is the weight value specified in the mesh's Edit Buttons. The <code>opacity</code> slider in the vertex paint Buttons is used to modulate the weight. To erase weight from vertices, set the weight to "0" and start painting.

**Note:** It is quite easy to change the weight since **TAB** will take you out of Weight Paint Mode into Edit Mode and Panels will automatically match the Context.

## Posemode

Relevant to Blender v2.31

To manipulate the bones in an armature, you must enter Pose Mode. In Pose Mode you can only select and manipulate the bones of the active armature. Unlike Edit Mode, you cannot add or delete bones in Pose Mode.

Enter Pose Mode by selecting an armature and pressing **CTRL-TAB**. Alternatively you can activate Pose Mode by selecting an armature and clicking on the Pose Mode menu entry in the Mode Menu of the 3D Window header (Figure 16-8). You can leave Pose Mode by the same method, or by entering Edit Mode.

Мо		
e	Pose Mode	
A	Edit Mode	
K,	Object Mode	
ť,	Object Mode 😑	₿:

Figure 16-8. Pose Mode Menu entry.

In Pose Mode, you can manipulate the bones in the armature by selecting them with **RMB** and using the standard transformation keys: **RKEY**, **SKEY** and **GKEY**. You cannot "grab" (translate) bones that are IK children of another bone, since the IK chain must stay continuous.

Press IKEY to insert KeyFrames for selected bones.

If you want to clear the posing for one or more bones, select the bones and press **ALT-R** to clear rotations, **ALT-S** to clear scaling and **ALT-G** to clear translations. Issuing these three commands with all bones selected will return the armature to its rest position.

It is frequently convenient to copy poses from one armature to another, or from one Action to a different point in the same Action. This is where the pose copying tools in the Armature Menu come into play.

For best results, be sure to select all bones in Edit Mode and press **CTRL-N** to autoorient the bone handles before starting any animation.

Insert K	Insert Keyframe			
Paste Flipped Pose				
Paste P	Paste Pose			
Сору С	Copy Current Pose			
Transform				
Transform Properties				
Armature	🕲 Pose Mode		¢	

Figure 16-9. Pose Mode Button.

To copy a pose, select one or more bones in Pose Mode, select the Armature>>Copy Current Pose Menu entry in the 3D Window header (Figure 16-9). The transformations of the selected bones are stored in the copy buffer until needed or until another copy operation is performed.

To paste a Pose, simply chose the Armature>>Paste Pose Menu entry (Figure 16-9). If Action auto key framing is active, KeyFrames will be inserted automatically.

To paste a mirrored version of the Pose (if the character was leaning left in the copied Pose, the mirrored Pose would have the character leaning right), use the Armature>>Paste Flipped Pose Menu entry (Figure 16-9). Note that if the armature was not set up correctly, the paste flipped technique may not work as expected.

# **Action Window**

#### Relevant to Blender v2.31

An Action is made of one or more Action channels. Each channel corresponds to one of the bones in the armature, and each channel has an Action IPO associated with it. The Action Window provides a means to visualize and Edit all of the IPOs associated with the Action together.

Tip: You can activate the Action Window with SHIFT-F12 (Figure 16-10).



Figure 16-10. Action Window

For every key set in a given Action IPO, a marker will be displayed at the appropriate frame in the Action Window. This is similar to the "Key" mode in the IPO Window. For Action channels with constraint IPOs, there will be one or more additional constraint channels beneath each Action channel. These channels can be selected independently of their owner channels (Figure 16-11).



Figure 16-11. Action Window with a Constraint

A block of Action keys can be selected by either **RMB** on them or by using the boundary select tool (**BKEY**). Selected keys are highlighted in yellow. Once selected, the keys can be moved by pressing **GKEY** and moving the mouse. Holding **CTRL** will lock the movement to whole-frame intervals. **LMB** will finalize the new location of the keys, while **ESC** cancels the Action and returns to previous state.

A block of Action keys can also be scaled horizontally (effectively speeding-up or slowing-down the Action) by selecting number of keys and pressing **SKEY**. Moving the mouse horizontally will scale the block. **LMB** will finalize the operation.

Delete one or more selected Action keys by pressing **XKEY** when the mouse cursor is over the KeyFrame area of the Action Window.

A block of Action keys can be duplicated and moved within the same Action by selecting the desired keys and pressing **SHIFT-D**. This will immediately enter grab mode so that the new block of keys can be moved. Subsequently **LMB** will finalize the location of the new keys. **ESC** will exit grab, but won't remove duplicates.

You can also delete one or more entire Action or constraint channels (and all associated keys) by selecting the channels in the left-most portion of the Action Window (the selected channels will be highlighted in blue). With the mouse still over the lefthand portion of the window, press **XKEY** and confirm the deletion. Note that there is no undo so perform this operation with care. Also note that deleting an Action channel that contains constraint channels will delete those constraint channels as well.

**Baking Actions:** If you have an animation that involves constraints and you would like to use it in the game engine (which does not evaluate constraints, and is not covered in this Book), you can bake the Action by pressing the BAKE button in the Action Window ToolBar. This will create a new Action in which every frame is a KeyFrame. This Action can be played in the game engine and should display correctly with all constraints removed. For best results, make sure that all constraint targets are located within the same armature.

You can actually see the Action IPO associated to a bone in the IPO Window instead of in the Action Window if you switch to an IPO Window (Figure 16-12). The Action IPO is a special IPO type that is only applicable to bones. Instead of using Euler angles to encode rotation, Action IPOs use quaternions, which provide better interpolation between Poses.



Figure 16-12. Action IPO

Quaternions use a four-component vector. It is generally difficult and unintuitive to describe the relationships of these quaternion channels to the resulting orientation, but it is often not necessary. It is best to generate quaternion KeyFrames by manipulating the bones directly, only editing the specific curves to adjust lead-in and lead-out transitions.

## **Non Linear Animation**

#### Relevant to Blender v2.31

Non Linear Animation is a technique somewhat akin to RVK used to merge different, simple, Actions in complex, fluid Actions. The NLA Window gives an overview of all of the animation in your scene. From here you can edit the timing of *all* IPOs, as if

they were in the Action Window. Much of the editing functionality is the same as the Action Window.

You can display the NLAWindow with CTRL-SHIFT-F12 (Figure 16-13).



Figure 16-13. NLA Window

You can also use this window to perform Action blending and other Non-Linear Animation tasks. You add and move Action Strips in a fashion similar to the Sequence Editor, and generate blending transitions for them.

In the NLA Window Actions are displayed as a single strip below the object's strip; all of the KeyFrames of the Action (constraint channel KeyFrames included) are displayed on one line (Figure 16-14). To see an expanded view of the Action, use the Action Window.



Figure 16-14. Expanded Action in NLA Window

Objects with constraint channels will display one or more additional constraint strips below the object strip. The constraint strip can be selected independently of its owner object (Figure 16-15).



Figure 16-15. Expanded Constraint in NLA Window

**RMB** clicking on object names in the NLA Window will select the appropriate objects in the 3D Window. Selected object strips are drawn in blue, while unselected ones are red.

You can remove constraint channels from objects by clicking **RMB** on the constraint channel name and pressing **XKEY**.

Note: Note that only armatures, or objects with IPOs will appear in the NLA Window.

### Working with Action Strips

Action strips can only be added to Armature objects. The object does not necessarily need to have an Action associated with it first.

Add an Action strip to an object by moving the mouse cursor over the object name in the NLA Window and pressing **SHIFT-A** and choosing the appropriate Action to add from the popup menu. Note that you can only have one Action strip per line.

You can select, move and delete Action strips along with other KeyFrames in the NLA Window.

The strips are evaluated top to bottom. Channels specified in strips later in the list override channels specified in earlier strips.

You can still create animation on the armature itself. Channels in the local Action on the armature override channels in the strips. Note that once you have created a channel in the local Action, it will always override all Actions. If you want to create an override for only part of the timeline, you can convert the local Action to an Action strip by pressing **CKEY** with your mouse over the armature's name in the NLA Window. This removes the Action from the armature and puts it at the end of the Action strip list.

Each strip has several options which can be accessed by selecting the strip and pressing **NKEY** (Figure 16-16). The options available are as follows:



Figure 16-16. NLA Action Strip Options

- StripStart/StripEnd The first and last fame of the Action strip in the timeline.
- ActionStart/ActionEnd The range of keys to read from the Action. The end may be less than the start which will cause the Action to play backwards.
- Blendin/Blendout The number of frames of transition to generate between this Action and the one before it in the Action strip list.
- Repeat The number of times the Action range should repeat. Not compatible with Use Path setting.
- Stride The distance (in Blender units) that the character moves in a single cycle of the Action (usually a walk cycle Action). This field is only needed if Use Path is specified.
- Use Path If an armature is the child of a path or curve and has a Stride value, this button will choose the frame of animation to display based on the object's position along the path. Great for walkcycles.
- Hold If this is enabled, the last frame of the Action will be displayed forever, unless it is overridden by another Action. Otherwise the armature will revert to its rest position.

#### Chapter 16. Character Animation

• Add - Specifies that the transformations in this strip should *add* to any existing animation data, instead of overwriting it.

# Constraints

Relevant to Blender v2.31

Constraints are filters that are applied to the transformations of bones and objects. This section is actually quite general and does not apply only to character animation since many other animations can benefit from constraints.

Blender Constraints can provide a variety of services including tracking and IK solving.

To add a constraint to an object, ensure you are in object Mode and in Object Context (F7) and that an Object is selected. If you are adding a Constraint to a Bone be sure to be in Pose Mode rather than Object Mode and select a Bone. The Object Context Buttons Window will present a Constraints Panel (Figure 16-17). Click on the Add button. A menu of possible constraints will appear.

Constraints	Effects
Add >> To	Object: Plane

Figure 16-17. Constraints Panel.

Once you selected the desired constraint its buttons will appear. A constraint can be deleted by clicking on the "X" icon next to it. A constraint can be collapsed by clicking on its orange triangle icon. When collapsed, a constraint can be moved up or down in the constraint list by clicking on it at choosing Move Up or Move Down from the popup menu.

For most constraints, a target must be specified in the appropriate field. In this field you must type in the name of the desired target object. If the desired target is a bone, first type in the name of the bone's armature. Another text box will appear allowing you to specify the name of the bone.

## **Constraint Types**

Several Constraints are possible. All apply to Bones, some also apply to other Objects:

• Copy Location - The constraint forces the Object to have an one or more coordinates (chosen via the three Toggle Buttons) of its location equal to those of the target (Figure 16-18).

X Copy Location Const	
OB:	
X Y Z	
Inf:1.000	Edit

Figure 16-18. Copy Location Constraint.

• Copy Rotation - This constraint copies the global rotation of the target and applies it to the constraint owner (Figure 16-19).

× Copy Rotation	Const.001	
OB:		
Inf:1.000 =	Edit	

Figure 16-19. Copy Rotation Constraint.

• Track To - This constraint causes the constraint owner to point one of its axes (by default the Y-axis) towards the target either in its positive or negative direction, depending on the selected Radio Buttons. The Object rotation will be computed so that another one of its axis (by default the Z-axis) will point up, again this can be changed via the pertinent Radio Buttons. (Figure 16-20).

× Track To	Const.00	)2 [	
OB:			
X V Z -	-X -Y -Z	XYZ	
Inf:1.000	_	Edit	

Figure 16-20. Track To Constraint.

• Locked Track - This constraint causes the constraint owner to point one of its axes (by default the Y-axis) to point towards the target either in its positive or negative direction, depending on the selected Radio Buttons. The Object rotation will be computed so that another one of its axis (by default the Z-axis) direction is *fixed*, again this can be changed via the pertinent Radio Buttons.

Actually this means that the Object is rotated around it fixed axis so that the Target lies on the plane defined by the locked axis and the pointing axis. (Figure 16-21).

X Locked Track	Const.003	
OB:		
<u>x y z -x</u>	<u>-V -Z X V Z</u>	
Inf:1.000	Edit	

Figure 16-21. Lock Track.

• Follow Path - This constraint needs the Target to be a Curve or Path. It causes the constraint owner to follow the path in time.

By default the Object translates along the curve in 100 frames. You can make the Object orientation follow the curve with the CurveFollow Toggle Button and by

#### Chapter 16. Character Animation

setting the Radio Buttons below to define which axis should be tangent to the curve and which should point up. To change the number of frames in which the Path is followed you need to edit the Curve's Speed IPO. (Figure 16-22).

× Follow Path	Const.004	
OB:		
CurveFollow	<ul> <li>Intersection of the section of the secti</li></ul>	
X Y Z -X	-Y-Z XYZ	
Inf:1.000	Edit	

Figure 16-22. Follow Path.

• IK Solver (Bone Only) - To simplify animation of multi-segmented limbs (such as arms and legs) you can add an IK solver constraint. IK constraints can only be added to bones. Once a target is specified, the solver will attempt to move the *root* of the constraint-owning bone to the target, by re-orienting the bone's parents (but it will not move the root of the chain). If a solution is not possible, the solver will attempt to get as close as possible. Note that this constraint will override the orientations on any of the IK bone's parents (Figure 16-23).

X IK Solver	Const.005	
Tolerance:	: 0.001 🛌 🛛 Iterations: !	500>
OB:		
Inf:1.000	Edit	

Figure 16-23. IK Solver Constraint.

**Note:** If the Target of the IK Constraint is another bone of the *same* Armature, as is highly recommended, you must make sure that this bone, usually denominated IK\_Tool, is *not* the child of any other bone of the IK chain, or weird results will happen.

• Action (Bone Only) - An Action constraint can be used to apply an Action channel from a different Action to a bone, based on the rotation of another bone or object. The typical way to use this is to make a muscle bone bulge as a joint is rotated. This constraint should be applied to the bone that will actually do the bulging; the target should point to the joint that is being rotated (Figure 16-24).

× Action	Const.006
08	3:
AC:	Start: 1 Min: 0.00
X Rot	♦ End: 1 ►
Inf:1.000	Edit

Figure 16-24. Action Constraint.

The AC field contains the name of the Action that contains the flexing animation. The only channel that is required in this Action is the one that contains the bulge animation for the bone that owns this constraint.

The Start and End fields specify the range of motion from the Action.

The Min and Max fields specify the range of rotation from the target bone. The Action between the start and end fields is mapped to this rotation (so if the bone rotation is at the Min point, the Pose specified at Start will be applied to the bone). Note that the Min field may be higher than the Max.

The pulldown menu specifies which component of the rotation is to be considered.

• Null - This is a constraint that does nothing at all; it doesn't affect the object's transformation directly. The purpose of a null constraint is to use it as a separator, and why this might be necessary will be clarified in the following section (Figure 16-25).

× Null	Const.007	

Figure 16-25. Null Constraint.

### **Constraints Evaluation Rules and Precedence**

Constraints can be applied to objects or bones. In the case of constraints applied to bones, any constraints on the armature *object* will be evaluated before the constraints on the bones are considered.

When a specific constraint is evaluated, all of its dependencies will have already been evaluated and will be in their final orientation/positions. Examples of dependencies are the object's parent, its parent's parents (if any) and the hierarchies of any targets specified in the constraint.

Within a given object, constraints are executed from top to bottom. Constraints that occur lower in the list may override the effects of constraints higher in the list. Each constraint receives as input the results of the previous constraint. The input to the first constraint in the list is the output of the IPOs associated with the object.

If several constraints of the same type are specified in a contiguous block, the constraint will be evaluated *once* for the entire block, using an average of all the targets. In this way you can constrain an object to track to the point between two other objects, for example. You can use a Null constraint to insert a break in a constraint block if you would prefer each constraint to be evaluated individually.

Looping constraints are not allowed. If a loop is detected, all of the constraints involved will be temporarily disabled (and highlighted in red). Once the conflict has been resolved, the constraints will automatically re-activate.

### Influence

The influence slider next to each constraint is used to determine how much effect the constraint has on the transformation of the object.

If there is only a single constraint in a block (a block is a series of constraints of the same type which directly follow one another), an influence value of 0.0 means the constraint has no effect on the object. An influence of 1.0 means the constraint has full effect.

If there are several constraints in a block, the influence values are used as ratios. So in this case if there are two constraints, A and B, each with an influence of 0.1, the resulting target will be in the centre of the two target objects (a ratio of 0.1:0.1 or 1:1 or 50% for each target).

Influence can be controlled with an IPO. To add a constraint IPO for a constraint, open an IPO Window and change its type to constraint by clicking on the chain icon. Next click on the Edit IPO Button next to the constraint you wish to work with. If there is no constraint IPO associated with the constraint yet, one will be created. Otherwise the previously assigned IPO will be displayed. At the moment, KeyFrames for constraint IPOs can only be created and edited in the IPO Window, by selecting the INF channel and **CTRL-LMB** in the IPO space.

When blending Actions with constraint IPOs, note that only the IPOs on the armature's local Action IPOs are considered. Constraint IPOs on the Actions in the motion strips are ignored.

**Important:** In the case of armatures, the constraints IPOs are stored in the current Action. This means that changing the Action will change the constraint IPOs as well.

# **Rigging a Hand and a Foot**

by Lyubomir Kovachev Relevant to Blender v2.31

## The Hand

Setting up a hand for animation is a tricky thing. The gestures, the movements of wrists and fingers are very important, they express emotional states of the character and interact with other characters and objects. That's why it's very important to have an efficient hand set-up, capable of doing all the wrist and fingers motions easily. Here is how to do it:



Figure 16-26. The Arm model

We'll use a simple cartoony arm mesh in this tutorial (Figure 16-26).

The following set-up uses one IK solver for the movement of the whole arm and four other IK solvers, one for each finger. The rotation of the wrist is achieved by a simple FK bone.

OK. Take a look at the arm mesh and let's start making the armature.



Figure 16-27. Drawing the armature

Position the 3D cursor in the shoulder, go to front view and add an armature. Make a chain of three bones - one in the upper arm, the second one in the lower arm and the third one should fit the palm, ending at the beginning of the middle finger. This is called a chain of bones. (Figure 16-27).



Figure 16-28. The armature in side view.



Figure 16-29. Placing the armature in side view.

Now change the view to side view and displace the bones so that they fit in the arm and palm properly (Figure 16-28 and Figure 16-29).



Figure 16-30. Wrist IK solver.

Zoom in the hand and position the cursor at the root of the bone, positioned in the palm. Add a new bone, pointing right, with the same length as the palm bone. This will be the IK solver for the arm. (Figure 16-30).



Figure 16-31. Rigging the finger.

Position the 3D cursor at the beginning of the middle finger and in front view start building a new chain, consisting of four bones (Figure 16-31). Three of them will be the actual bones in the finger, and the fourth bone will be a null bone - this is a small bone, pointing to the palm, that will help turning the whole chain to an IK chain later.

Again, change to side view and reshape the bones so that they fit the finger well. It could be a tricky part and you may also view the scene using the trackball while reshaping the bones (Figure 16-32).



Figure 16-32. Rigging the finger.



Figure 16-33. Adding the finger IK solver.

Now add the IK solver for this finger chain. Position the 3D cursor at the root of the null bone and add bone with the length of the other three bones in the finger (Figure 16-33).



Figure 16-34. Rigging the other fingers.

Repeat the same for the creation of the IK chains for the other three fingers. The only difference with the thumb is that it has two actual bones, instead of three. You can just copy and paste the chain and just reshape, reshape, reshape... (Figure 16-34).



Figure 16-35. Naming overview.

The time has come for the boring part - naming of the bones. You cannot skip this, because you'll need the bone names in the skinning part later. Bones are named as in Figure 16-35.

**Note:** The names of the bones of finger 1 and finger 2 are not shown here. They are identical to the names of the bones of finger 3, only the number changes.



Figure 16-36. Parenting the Thumb.

Now let's do some parenting.

Select the root thumb bone "ThumbA.R" (Figure 16-36) and in the edit menu click in the "child of" field and choose "Hand.R". You've just parented the thumb bone chain to the hand bone.

### Chapter 16. Character Animation



### Figure 16-37. Parenting the other fingers.

By repeating the same process parent the following bones (Figure 16-37):

- "Fing1A.R" to "Hand.R"
- "Fing2A.R" to "Hand.R"
- "Fing3A.R" to "Hand.R"
- "IK\_thumb.R" to "Hand.R"
- "IK\_fing1.R" to "Hand.R"
- "IK\_fing2.R" to "Hand.R"
- "IK\_fing3.R" to "Hand.R"

Why did we do all this? Why did we parent so much bones to "Hand.R"? Because when you rotate the hand (i.e. "Hand.R") all the fingers will follow the hand. Otherwise the fingers will stay still and only the palm will move and you'll get very weird results.

**Note:** No IK tool bone is child of any bone of the chain it controls. All of them are children of "Hand.R".



Figure 16-38. Setting the IK solver for the wrist. Selecting the bone.

Time to add constraints. Enter pose mode (Figure 16-38) and go in Object Context (F7). Choose "Hand.R" and add an IK solver constraint to it in the Constraints Panel. In the OB field type the object name: "Armature". The bone went to the centre of the armature, but we'll fix this now. In the new BO field, that appeared in the constraint window, type the bone name "IK\_arm.R". This will be the IK solver bone controlling the arm motion (Figure 16-39).

▼ Constraints	
Add >> To Bone: Hand.R	
X IK Solver Const	
Tolerance: 0.001 Iterations: 500	
OB:Armature	
BO:IK_arm.R	
Inf:1.000 Edit	

Figure 16-39. Setting the IK solver for the wrist. Setting the Constraint.

Now by repeating the same procedure:

- select "ThumbNull.R" and add IK solver "IK\_thumb.R",
- select "Fing1null.R" and add IK solver "IK\_fing1.R",
- select "Fing2null.R" and add IK solver "IK\_fing2.R",
- select "Fing3null.R" and add IK solver "IK\_fing3.R".

You're finished with the bone part. In pose mode select different IK solvers and move them to test the IK chains. Now you can move the fingers, the thumb, the whole arm and by rotating the "Hand.R" bone you can rotate the whole hand.

So let's do the skinning now. It's the part when you tell the mesh how to deform. You'll add vertex groups to the mesh. Each vertex group should be named after the

### Chapter 16. Character Animation

bone that will deform it. If you don't assign vertex groups, the deformation process will need much more CPU power, the animation process will be dramatically slowed down and you'll get weird results. It's highly recommended (almost mandatory) that you use subdivision surfaces meshes for your characters with low vertex count. Otherwise if you use meshes with lots of vertices, the skinning will be much more difficult. Don't sacrifice detail, but model economically, use as few vertices as possible and always use SubSurf.

Parent the Mesh to the Armature, in the Pop-Up select Armature and in the following select Name Groups. Your Mesh will be enriched by empty Vertex Groups.

Select the arm mesh, enter Edit Mode and switch to Editing (F9) Context. In the Mesh Tools 1 of the Edit Buttons Window notice the small group of buttons with the word Group on top. Thanks to the automatic naming feature, you have already created all the groups you needed. (Figure 16-40).



#### Figure 16-40. Vertex group names.

Actually the automatic Grouping scheme has created vertex groups also for the "IK" and "null" bones *unless* you have set them Unskinnable before. These are useless and you can safely delete them.

Now let's do the tricky part: Select the vertex group "ArmHi.R" from the edit buttons by clicking on the small button with the white minus sign. Now look at the 3D window. Select all the vertices that you want to be deformed by the "ArmHi.R" bone. (Figure 16-41).


Figure 16-41. ArmHi.R vertex group.

Now press the Assign button in the edit buttons window (Figure 16-42). You've just added the selected vertices to the "ArmHi.R" vertex group. These vertices will be deformed by the "ArmHi.R" bone.

Vertex Groups		
🗢 ArmHi.F	{	
🔹 Weight	: 1.000 🕞	
New Delete		
Assign Remove		
Select Desel.		
AutoTexSpace		

Figure 16-42. Assigning vertices to a group.

Repeat the same steps for the other vertex groups: select vertices and assign them to the corresponding group. This is a tricky process. Do it carefully. If you've assigned some vertices to a certain group by mistake, don't worry. Just select the unneeded vertices and press the Remove button. You can add a vertex to more than one vertex group. For example the vertices that build joints (of fingers, wrist, elbow, etc.) could be assigned to the two vertex groups that are situated close to it. You can also assign vertices to deform with different strength. The default strength is 1.000, but you can add vertices with strength 0.500 or less. The lower the strength value, the less deformation for that vertex. You can make a vertex deform 75% by one bone and 25% by another, or 50% by one and 50% by another. It's all a matter of testing the deformation until you achieve the result you want. In general if your arm model has half-flexed joints (as the model in this tutorial you will get good results without using strength values different than 1.000. My own rule of thumb when modelling a character is: always model the arms fingers and legs half-flexed, not straight. This is a guarantee for good deformation.

When you're finished adding vertices to vertex groups, if you haven't made any mistakes, you'll have a well set up arm with a hand. Select the armature, enter pose mode, select the different IK solvers and test the arm and fingers (Figure 16-43).



Figure 16-43. Different Poses.

### The Foot

The set-up of legs and feet is maybe the most important thing in the whole rigging process. Bad foot set-up may lead to the well known "sliding-feet" effect, which is very annoying and usually ruins the whole animation. A well made complex foot set-up must be capable of standing still on the ground while moving the body, and doing other tricky stuff like standing on tiptoe, moving the toes, etc. Now we're going to discuss several different foot set-ups that can be used for different purposes.



Figure 16-44. A (wrong) leg rig.

First let's see how a bad foot set-up looks like (Figure 16-44).

Start building a bone chain of three bones - one for the upper leg, the second one for the lower leg and the third one for foot. Now move the 3D cursor at the heel joint and add another bone - this will be the IK solver. Now add that bone as an IK solver constraint to the foot bone. (Figure 16-45).

X IK Solver Const Tolerance: 0.001 Iterations: 500 OB:Armature B0:IK_Foot	
Inf:1.000 Edit	

Figure 16-45. Assigning the IK constraint.



Figure 16-46. The rig in pose mode.

Test the armature: in pose mode grab the IK solver and move it - it's moving OK. Now grab the first bone in the chain (the upper leg) and move it. The foot is moving too and we don't want this to happen! (Figure 16-46).

Usually in an animation you'll move the body a lot. The upper leg bone is parented to the body and it will be affected by it. So every time you make your character move or rotate his body, the feet will slide over the ground and go under it and over it. Especially in a walkcycle, this would lead to an awful result.

#### Chapter 16. Character Animation



Figure 16-47. Adding a toe and some more IK Animation.

Now maybe you think this could be avoided by adding a second IK solver at the toes (Figure 16-47). Let's do it. Start a new armature. Add a chain of four bones: upper leg, lower leg, foot and toes. Add two IK solvers - one for the foot and one for the toes. Parent the toe IK solver bone to the foot IK solver bone.

**Note:** The toe IK solver is parented to the Foot IK solver. This latter must *not* be children of any other bone in the armature. Be sure of this and, to delete a parent relationship, remember that you can do so by selecting the empty entry in the Child of: menu. Remember to check this for all subsequent examples.



Figure 16-48. Moving the leg.

Test this setup - grab the upper leg bone and move it (Figure 16-48). Well, now the sliding isn't so much as in the previous setup, but it's enough to ruin the animation.



Figure 16-49. Rigging with a null bone.

Start a new armature. Make a chain of three bones - upper leg, lower leg and a null bone. The null bone is a small bone, that we'll add the IK solver to. Now position the 3D cursor at the heel and add the foot bone. Now add the foot bone as an IK solver constraint to the null bone (Figure 16-49). (You can also add another bone as an IK solver and add a "copy location" constraint to the foot bone, with the IK solver as target bone.)



Figure 16-50. Rigging with a null bone.

Test this - now it works. When you move the upper leg the foot stands still (Figure 16-50). That's good. But still not enough. Move the upper leg up a bit more. The leg chain goes up, but the foot stays on the ground. Well, that's a shortcoming of this setup, but you're not supposed the raise the body so much and not move the IK solver up too during animation...

#### Chapter 16. Character Animation



Figure 16-51. Adding the toe.

Again, build a chain of three bones - upper leg, lower leg and null bone. Position the 3D cursor at the heel and add a chain of two bones - the foot bone and a bone for the toes. Now add an IK solver to the foot bone (Figure 16-51).

Test it. This is a good set-up with a stable, isolated foot and moving toes. But you still cannot stand on tiptoe with this set-up.



Figure 16-52. Full complete leg rig.



Figure 16-53. Zoom on the foot rig.

Build a chain of three bones - upper leg, lower leg and null bone (name it LegNull) (Figure 16-52). Starting at the heel point, make a second chain of two bones only - foot bone (Foot) and a small null bone (FootNull). Position the 3D cursor at the end of the foot bone and add the toe bone (Toes). From the same point create an IK solver bone (IK\_toes). Now position the 3D cursor at the heel and add another IK solver there (IK\_heel). Finally, starting somewhere near the heel, add a bigger IK solver (IK\_foot) (Figure 16-53).

Now let's add the constraints. Do the following:

- To the bone "Toes" add a copy location constraint with target bone "IK\_toes".
- To "FootNull" an IK solver constraint (target "IK\_toes").
- To "Foot" copy location (target "LegNull").
- To "LegNull" IK solver (target "IK\_heel").

Well, that's it. Now test the armature. Grab "IK\_foot" and move it up. Now grab "IK\_toes" and move it down. The foot changes its rotation, but it looks like the toes are disconnected from it. But if you animate carefully you'll always manage to keep the toes from going away from the foot. Now return the armature to its initial pose. Grab "IK\_heel" and "LegHi" and move them up. Now the character is standing on his tiptoes. The foot may appear disconnected from the toes again, but you can fix the pose by selecting "IK\_heel" only and moving it a bit forward or backwards. This setup may not be the easiest one for animation, but it gives you more possibilities than the previous set-ups. Usually when you don't need to make your character stand on tiptoe, you're better to stick to some of the easier set-ups. You'll never make a perfect set-up. You can just improve, but there will always be shortcomings.



Figure 16-54. Testing the setup.

# **Rigging Mechanics**

Relevant to Blender v2.31

Armatures are great also for rigging mechanical stuff, like robots, WarriorMechs etc. (Figure 16-55).



Figure 16-55. Four spider-mech legs.

First step is to create the mesh for the arms. We are not here for organic, we are here for mechanics. So no single mesh thing. The arm/leg/whatever is made of rigid parts, each part is a single mesh, parts moves/rotates one with respect to the other.

Although Figure 16-55 has four spider-like legs arms, each of which have 5 sections, it is clearer to explain the tricks with just a single joint arm.

My suggestion is this: make the arm with two equal sections, and the forearm, on the right, made by just one section. Note the cylinders which represents the shoulder (left) the elbow (centre) and the wrist (right) (Figure 16-56).



Figure 16-56. The Arm model

The other cylinders in the middle of the arm and forearm are the places where the piston will be linked to.

Note that it is much easier if the axis of mutual rotation (shoulder, elbow, etc.) are exactly on grid points. This is not necessary though, if you master well Blender Snap menu.

### **Pivot axis**

Then add the mechanical axes in the pivot points. Theoretically you should add one at each joint and two for every piston. For the sake of simplicity here there are only the two axes for the piston, made with plain cylinders (Figure 16-57).



Figure 16-57. The Arm model with pivot axis.

Note two things:

- It is fundamental that the centre of the mesh is exactly in the middle and exactly on the axis of rotation of the piston.
- Each axis must be parented to the pertinent arm mesh.

### The Armature

Now it is time to set up the armature. Just two bones are enough (Figure 16-58).



Figure 16-58. The Arm model and its armature

To have an accurate movement, the joints must be precisely set on the pivoting axis (this is why I told you to place such axes on grid points before, so that you can use the Move Selected To Grid feature).

Name the bones smartly (Arm and Forearm, for example). Parent the Arm Mesh to the armature, selecting the *Bone* option and the Arm bone. Do the same with the forearm mesh and forearm bone.

**Parent to Bone:** Parent to Bone effectively makes the Object follow the bone without any deformation. This is what should happen for a robot which is made by undeformable pieces of steel!



Figure 16-59. The Arm model in Pose Mode

If you switch to Pose Mode you can move your arm by rotating the bones. (Figure 16-59). You can add an Inverse Kinematics (IK) solver as we did in the previous section if you like.

## **Hydraulics**



Figure 16-60. Hydraulic piston.

Make a piston with two cylinders, a larger one and a thinner one, with some sort of nice head for linking to the pivoting points (Figure 16-60).

It is *mandatory* for the two pieces to have the mesh centre exactly on the respective pivoting axis.

Place them in the correct position and parent each piston piece to the pertinent mesh representing the axis. (Figure 16-61).



Figure 16-61. Hydraulic piston on the arm.

If you now rotate the two pieces in the position they should have to form a correct *still* image you get a nice piston. (Figure 16-62, left).



Figure 16-62. Hydraulic piston in Pose Mode.

But if you switch to Pose Mode and start moving the Arm/Forearm the piston gets screwed up... (Figure 16-62, right).

To make a working piston you must make each half piston track *the other half piston's pivot axis* cylinder mesh (Not the other half piston! This would create a constraint loop). This is why the position of all the mesh centres is so critical (Figure 16-63).



Figure 16-63. Hydraulic piston with mutual tracking.

Select half a piston, select the other half piston's axis mesh, and, in Object Context (F7) and Constraints panel add a Track To Constraint. The buttons below X, Y... must be appropriately set (Figure 16-64).

Constraints	Effects
Add >> T	o Object: Cylinder.004
× Track To	AutoTrack
<u>OB:Cy</u>	/linder.00
X V Z ->	-Y-Z XYZ
Inf:1.000	l Edit

Figure 16-64. Track settings.

**Note:** If you prefer Old Track, remember also to press the PowerTrack button in the Anim Setting Panel for a nicer result.

Now, if you switch to Pose Mode and rotate your bones the piston will extend and contract nicely, as it should in reality. (Figure 16-65).



Figure 16-65. Pose Mode for the arm with hydraulics.

The next issue is, since pistons work when pressurised oil is pumped into them, for a really accurate model we should add some hydraulic hoses. But how to place a nicely deforming tube going from arm to piston? The two ends should stick to two rigid bodies reciprocally rotating. This requires IK!



Figure 16-66. Adding a flexible tube.

First add a mesh in the shape of the tube you want to model (Figure 16-66).

Personally I prefer to draw the tube in its bent position as a bevelled curve. This is done by adding a Bézier curve, adding a Bézier circle, and using the Bézier circle as BevOb of the Bézier curve. Then convert that to a mesh (**ALT-C**) to be able to deform it with an armature.



Figure 16-67. Adding the armature to the tube.

Then add an armature. A couple of bones are enough. This armature should go from the tube's 'fixed' end to the tube's 'mobile' end. Add a third bone which will be used for the Inverse Kinematics solution (Figure 16-67).

Be sure that the armature is parented to the object where the 'fixed' part of the tube is, well, fixed. In this case the robot arm. Also add an Empty at the 'mobile' end of the tube. (Figure 16-68).



Figure 16-68. The Empty for the IK Animation solution.



Figure 16-69. IK constraint.

Parent the Empty to the 'mobile' part of the structure. In this case the outer part of the piston to which the tube is linked. In Pose Mode go to the Object Context and Constraints Panel. Select the last bone, the one which starts from where the tube ends, and Add a constraint. Select the IK solver type of constraint and select the newly created Empty as target Object OB:. (Figure 16-69). You can play with Tolerance: and Iterations: if you like.

Lastly, parent the tube to the Armature via the 'Armature' option. Create Vertex groups if you like. Now if, in Pose Mode, you move the arm, the two parts of the piston keep moving appropriately, and the Empty follows. This obliges the IK Armature of the tube to move, to follow the Empty, and, consequently, the Tube to deform (Figure 16-69).



Figure 16-70. Full robot arm in Pose Mode.

**Note:** You can use a bone of the Armature, instead of an Empty, as an IK solver, but in this case you cannot parent the bone to the moving object. You can on the other hand, use a Copy Location constraint, but this is not as easy since the copy location would move the end of the armature to the center of the moving object, which is not the right place.

### How to setup a walkcycle using NLA

by Claudio 'Malefico' Andaur

*Relevant to Blender v2.31* 

In this tutorial we will set up a walkcycle and use it with the Path option in the Blender NLA Editor. Before starting let me tell you that you will need to have a basic knowledge of the animation tools, (armature set up), in order to follow the text, and have a lot of patience. It is highly recommended that you read all the preceding NLA related parts of this Book.

We are going to use a character set up like the one explained in the "Hand and Foot Rigging tutorial", that is with foot bones split up from the leg and using an extra null bone to store the IK solver constraint. For further details please check that section!

Having a rigged character, the first thing we need to do is to define actions: "WALKCYCLE", "WAVE\_HAND" and "STAND\_STILL". In WALKCYCLE and STAND\_STILL there will be KeyFrames set for almost all control bones while in "WAVE\_HAND" there are KeyFrames only for the arm and hand. This will allow our character to simultaneously wave its hand while walking.

The main idea behind this is to work on each single movement and later on combining everything in the NLA window.

#### The path to success

There are two main ways to animate a walkcycle, first one is to make the character actually advance through the poses of the cycle and the second one is to make the character walk *in place* thus without real displacement.

The latter option, though more difficult to set up, is the best choice for digital animation and it is our choice for this tutorial.

The whole walkcycle will be an "action" for our armature, so let's go an create a new action and switch to "pose mode" to get something like Pose 1 (the so called *"contact pose"* in Figure 16-71.



Figure 16-71. Some common poses in a walkcycle.

**Note:** There are some details to bear in mind at the time of setting up an armature for walkcycle. If we adopt Blender's naming convention introduced in the Section called *The Armature Object* you will be able to paste flipped poses. Also, before parenting your armature to your model, be sure their local axis are aligned to the global axis by selecting them and pressing **CTRL-SHIFT-A**.

To animate our walking model we will restrict us to animate a few control bones. In the case of the legs we are going to animate its feet since the IKA solvers will adjust the leg bones better than us. To ensure that feet will move in fixed distances, please activate the Grab Grid option in the User Preferences Window View and Controls buttons before start moving bones, reduce the grid size if needed.

A nice method is to hide, with the relative toggle button, all the bones we are not going to set KeyFrames for. This way is easier to see the model during animation and keeps our task simple.

Normally a walkcycle involves four poses, which are commonly known as *contact*, *recoil*, *passing* and *high-point*. Take a look at Figure 16-71.

The most important pose is "Contact pose". Most animators agree every walkcycle should start by setting up this pose correctly. Here the character covers the widest distance it's possible to do in one step. In "Recoil pose", the character is in its lower position, with all its weight over one leg. In "High-point pose", the character is in its higher position, almost falling forwards. "Passing pose" is more like an automatic pose in-between recoil and high-point.

The work routine is as follows:

- 1. Pose the model in contact pose in frame 1.
- 2. Insert KeyFrames for the control bones of your armature (those you use for grabbing, mainly IK solvers).
- 3. Without deselecting them press the "Copy Pose" button. Now the bone's location and rotations have been stored in memory.
- 4. Go a few frames forward and press "Paste Flip Pose". The flip pose will be pasted in this frame, so if in the previous frame the left leg was forwards now it will be backwards, and viceversa.
- 5. Now once again select your control bones and insert KeyFrames for them.
- 6. Go a few frames forward again (it is recommendable that you use the same number of frames as before, an easy choice is to go just 10 frames every ahead time) and press "Paste Pose", this will paste the initial pose ending the cycle. This way we have achieved a "Michael Jackson" style walkcycle since our character never lift its feet up from the ground.

#### Chapter 16. Character Animation

- 7. To fix it, go to some intermediate position between the first two poses and move the feet to get something like the Recoil Pose in Figure 16-71, where the waist reaches its lower position.
- 8. Insert KeyFrames and copy the pose.
- 9. Now go to a frame between the last two poses (inverse contact and contact) and insert the flip pose. Insert the required KeyFrames and we are done.

**Tip:** If on the contrary you see that the mesh is weirdly deformed, don't panic!, go to EditMode for the armature, select all bones and press **CTRL-N**. This will recalculate the direction of bones rolls which is what makes the twisting effect.

You should follow the same routine for all the poses you want to include in your walkcycle. I normally use the contact, recoil, and high point poses and leave Blender to make the passing pose.



Figure 16-72. Use copy, paste and paste-flip pose buttons to be happy!

Now if you do ALT-A you will see our character walking almost naturally.

It will be very useful to count how many Blender Units (B.U.) are covered with each step, which can be done counting the grid squares between both feet in Pose 1. This number is the STRIDE parameter that we are going to use later on in the NLA window.

Now we will focus on making the character actually advance through the scene.

First of all deselect the walkcycle action for our armature so it stops moving when pressing **ALT-A**. To do this, press the little X button besides the action name in the action window.

Then we will create a PATH object for our hero in the ground plane, trying not to make it too curved for now (the straighter the better), once done let's parent the character's Armature to the path (a *normal* parent, not a Follow Path!). If everything went OK, we will see our character moving stiffly along the path when pressing **ALT-A**.

Now go to the NLA window and add the walkcycle action in a channel as an NLA strip. With the strip selected press **N** and then push the **Use Path** button.

**Note:** It is convenient that at the moment of adding actions in the NLA window, that no action is selected for the current armature. Why? Because instead of an NLA strip, we'll see the individual KeyFrames of the action being inserted in the armature channel and this KeyFrames will override any prior animation strips we could have added so far. Anyway,

if you do insert an action in this way, you can always convert the KeyFrames into an NLA strip by pressing **CKEY**.



Figure 16-73. A nice stroll

Now if you start the animation again some funny things might happen. This is because we haven't set the Stride parameter.

This value is the number of Blender Units that should be covered by a single walkcycle and it is very important that we estimate it with accuracy. Once calculated we should enter it in the Stride Num Button which appears if, once you have selected the strip, you press the **NKEY**.

If we adjust it well and if the walkcycle was correctly set up, our character should not "slide" across the path.

One way to estimate the Stride value accurately is to count how many grid squares there are between the toes of the feet in Pose 1. This value is multiplied by 2 and by the grid scale (normally 1 grid square = 1 B.U. but this might not be the case, for instance in this example 2 grid squares = 1 B.U.) to provide the required STRIDE value.

In the example there are 7.5 squares with GRID=1.0, since the Grid scale is 1.0 we have: STRIDE =  $7.5 \times 1.0 \times 2 = 15$ 



Figure 16-74. Estimating the STRIDE. Refine the grid if needed!

It's likely that we want our character to walk faster or slower or even stop for a while. We can do all this by editing the path's Speed curve.

Select the path and open an IPO window. There we will see a Speed curve normalized between 0 and 1 in ordinates (Y axis) and going from frame 1 to the last one in the 335

#### Chapter 16. Character Animation

X axis. The Y coordinate represents the relative position in the path and the curve's slope is the speed of the parented objects. In Edit Mode we will add two points with the same Y coordinate. This flat part represents a pause in the movement and it goes from frame 40 to frame 60 in the figure.

The problem here is that when our character stops because of the pause in the curve, we will see him in a "frozen" pose with a foot on the ground and the other in the air.



#### Figure 16-75. Having a rest in the walk

To fix this little problem we will use the NLA window. What we have to do is to insert the "STAND\_STILL" action, this is a pose where our character is at rest. I have defined this action as only one frame by erasing all displacements and rotations of the bones. (See Clearing Transformations), and then moving a couple of bones to get a "resting" attitude.

Since the pause is from frame=78 to frame=112 we should insert this "still" action exactly there for it to perfectly fit the pause. For the animation doesn't start nor end briskly we can use the BlendIn and BlendOut options, where we can set the number of frames used to blend actions and in this way doing a more natural transition between them. In this way the character will smoothly change its pose and everything will look fine. If we do use a BlendIn or BlendOut value, to be set in the **NKEY** dialog, then we should start the action BlendIn frames earlier and finish it BlendOut frames later, because the character should be still moving while changing poses.

We can of course combine different walkcycles in the same path as for instance change from walking to running in the higher speed zone.

In all these situations we will have to bear in mind that the different effects will be added from one NLA strip to the precedent strips. So, the best option is to insert the walkcycle and still strips before any other.

#### Moving hands while walking

To add actions in the NLA window we have to locate the mouse pointer over the armature's channel and press **SHIFT-A**. A menu with all available actions will pop up. If we don't locate the pointer over an armature channel an error message ERROR: Not an armature" will pop up instead.

So, place the pointer over the armature strip and press **SHIFT-A** and add the "WAVE\_HAND" action.

As this particular action is just the waving of the left arm to say "hello" during some point in the walkcycle, we will not use the "Use Path" option but move it in time so it overlaps the arms KeyFrames from the walkcycle action. Move the pointer over the strip and press **NKEY** or just drag it and scale it to your satisfaction.



Figure 16-76. Hey guys!

Since this action is the last to be calculated (remember Blender evaluates actions from Top to Bottom in the NLA Editor), it will override any KeyFrames defined for the bones involved in the precedent actions.

Well, there is not much left to say about NLA and armatures. Now it is time for you to experiment and to show the results of your work to the world. One last recommendation though: it is possible to edit KeyFrames in the NLA window. We can duplicate frames (SHIFT-D), grab KeyFrames (GKEY) and also erase KeyFrames (XKEY), but if you do erase KeyFrames be careful because they will be lost forever from the currently selected action. So be careful and always convert to NLA strip before erasing anything.

Bye and good luck blenderheads!!

Chapter 16. Character Animation

# Chapter 17. Rendering

Relevant to Blender v2.31

Rendering is the final process of CG (short of postprocessing, of course) and is the phase in which the image corresponding to your 3D scene is finally created.

The rendering buttons window is accessed via the Scene Context and Render Subcontext (**F10** or the button). The rendering Panels and Buttons are shown in Figure 17-1.

2 Amp/ 2 Absolvuf	RENDER Pano Radio	ANIM	Game framing settings         >>         PAL           < SizeX: 640 +>         < SizeV: 480 +>         NTSC
© Mhype	OSA         MBLUB         100%           5         8         11         16         Bf: 0.500         75%         50%         25%	Do Sequence Bender Daemon	AspX: 100 > AspY: 100 >     Preview     PC
Backburf Edge Edge Settings>	< Xparts: 1 >	PLAY ( rt: 25 )	PNG 0 Crop PAN0 PAN0 PNG 0 Crop PAN0 PAN0 PNG 0 PNG
DispView DispWin Extensions	Sky Premd Key Border Gamma	< Stat 1 >> < End: 250 >>	5W RGB RGBA Unified Rende

Figure 17-1. Rendering Buttons.

The rendering of the current scene is performed by pressing the big RENDER button in the Render panel, or by pressing F12. The result of the rendering is kept in a buffer and shown in its own window. It can be saved by pressing F3 or via the File>>Save Image menu.

The image is rendered according to the dimensions defined in the Format Panel (Figure 17-2).



Figure 17-2. Image types and dimensions.

By default the dimensions SizeX and SizeY are 320x256 and can be changed as for any Num Button. The two buttons below define the aspect ratio of the pixels. This is the ratio between the X and Y dimensions of the pixel of the image. By default it is 1:1 since computer screen pixels are square, but can be varied if television shorts are being made since TV pixels are not square. To make life easier the rightmost block of buttons (Figure 17-3) provides some common presets:

#### Chapter 17. Rendering



Figure 17-3. Image pre-set dimensions.

- PAL 720x576 pixels at 54:51 aspect ratio.
- NTSC 720x480 pixels at 10:11 aspect ratio.
- Default Same as PAL, but with full TV options, as explained in the following sections.
- Preview 640x512 at 1:1 aspect ratio. This setting automatically scales down the image by 50%, to effectively produce a 320x256 image.
- PC 640x480 at 1:1 aspect ratio.
- PAL 16:9 720x576 at 64:45 aspect ratio, for 16:9 widescreen TV renderings.
- PANO Standard panoramic settings 576x176 at 115:100 aspect ratio. More about 'panoramic' renderings in the pertinent section.
- FULL 1280x1024 at 1:1 aspect ratio.

# **Rendering by Parts**

#### Relevant to Blender v2.31

It is possible to render an image in pieces, one after the other, rather than all at one time. This can be useful for very complex scenes, where rendering small sections one after the other only requires computation of a small part of the scene, which uses less memory.

By setting values different from 1 in the Xparts and Yparts NumButtons in the Render Panel (Figure 17-4), you force Blender to divide your image into a grid of Xparts times Yparts sub-images, which are then rendered one after the other and finally assembled together.

🔻 Render		
PENDER	Shadow	Env Map
nenven	Pano	Radio
OSA MBLUR	100	)%
5 8 11 16 Bf: 0.500	75% 50	% 25%
≪ Xparts: 1 →	Fields	Od X
Sky Premul Key	Border	Gamma

Figure 17-4. Rendering by parts buttons.

Note: Blender cannot handle more than 64 parts.

# **Panoramic renderings**

Relevant to Blender v2.31

To obtain nice panoramic renderings, up to a full  $360^{\circ}$  view of the horizon, Blender provides an automatic procedure.

If the Xparts is greater than 1 and the Pano button of the Render Panel is pressed (Figure 17-5), then the rendered image is created to be Xparts times SizeX wide and SizeY high, rendering each part by rotating the camera as far as necessary to obtain seamless images.

Shadow	EnvMap
Pano	Radio

Figure 17-5. Panorama button.

Figure 17-6 shows a test set up with 12 spheres surrounding a camera. By leaving the camera as it is, you obtain the rendering shown in Figure 17-7. By setting Xparts to 3 and selecting Pano the result is an image three times wider showing one more full camera shot to the right and one full to the left (Figure 17-8).

#### Chapter 17. Rendering



Figure 17-6. Panorama test set up.

To obtain something similar without the Panorama option, the only way is to decrease the camera focal length. For example Figure 17-9 shows a comparable view, obtained with a 7.0 focal length, equivalent to a very wide angle, or fish-eye, lens. Distortion is very evident.



Figure 17-7. Non-panoramic rendering.



Figure 17-8. Panoramic rendering.



Figure 17-9. Fish-eye rendering.

To obtain a full  $360^{\circ}$  view some tweaking is necessary. It is known that a focal length of 16.0 corresponds to a viewing angle of  $90^{\circ}$ . Hence a panoramic render with 4 xparts and a camera with a 16.0 lens yields a full  $360^{\circ}$  view, as that shown in Figure 17-10. This is grossly distorted, since a 16.0 lens is a wide angle lens, and distorts at the edges.



Figure 17-10. Full 360° panorama with 16.0 lenses.

To have undistorted views the focal length should be around 35.0. Figure 17-11 shows the result for a panorama with 8 xparts and a camera with a 38.5 lens, corresponding to a  $45^{\circ}$  viewing angle.



Figure 17-11. Full 360° panorama with 38.5 lenses.

The image is much less distorted, but special attention must be given to proportion. The original image was 320x256 pixels. The panorama in Figure 17-10 is  $4 \times 320$  wide. To keep this new panorama the same width, the Sizex of the image must be set to 160 so that  $8 \times 160 = 4 \times 320$ . But the camera viewing angle width occurs for the largest dimension, so that, if SizeX is kept to 256 the image spans  $45^{\circ}$  vertically but less than that horizontally, so that the final result is not a  $360^{\circ}$  panorama unless SizeX  $\geq$  SizeY or you are willing to make some tests.

### Antialiasing

#### Relevant to Blender v2.31

A computer generated image is made up of pixels, these pixels can of course only be a single colour. In the rendering process the rendering engine must therefore assign a single colour to each pixel on the basis of what object is shown in that pixel.

This often leads to poor results, especially at sharp boundaries, or where thin lines are present, and it is particularly evident for oblique lines.

To overcome this problem, which is known as *Aliasing*, it is possible to resort to an Anti-Aliasing technique. Basically, each pixel is 'oversampled', by rendering it as if it were 5 pixels or more, and assigning an 'average' colour to the rendered pixel.

#### Chapter 17. Rendering

The buttons to control Anti-Aliasing, or OverSAmple (OSA), are below the rendering button in the Render Panel (Figure 17-12). By pressing the OSA button antialiasing is activated, by selecting one of the four numeric buttons below it, the level of oversampling (from 5 to 16) is chosen.

RENDER		
OSA 5 8 11 16	MBLUR Bf: 0.500	
≪ Xparts: 1 →	Vparts: 1 🕨	
Sky Premul Ke	ey (	

Figure 17-12. OSA buttons.

Blender uses a Delta Accumulation rendering system with jittered sampling. The values of OSA (5, 8, 11, 16) are pre-set numbers that specify the number of samples; a higher value produces better edges, but slows down the rendering.

Figure 17-13 shows a rendering with OSA turned off and with 5 or 8 OSA samples.



Figure 17-13. Rendering without OSA (left) with OSA=5 (center) and OSA=8 (right).

### **Output formats**

Relevant to Blender v2.31

The file is saved in whichever format has been selected in the pertinent Menu button in the Format Panel (Figure 17-2). From here you can select many image or animation formats (Figure 17-14).

Save image as:	
Ftype	
Iris + Zbuffer	
Iris	
HamX	
Jpeg	
PNG	Ī
Targa Raw	Ì
Targa	1
QuickTime	•
AVI Jpeg	
AVI Raw	
PNG 🔶	

Figure 17-14. Image and animations formats.

The default image type is Targa, but, since the image is stored in a buffer and then saved, it is possible to change the image file type after the rendering and before saving using this menu.

By default Blender renders color (RGB) images (bottom line in Figure 17-2) but Black and White (BW) and colour with Alpha Channel (RGBA) are also possible.

Beware that Blender does *not* automatically add the extension to files, hence any .tga or .png extension must be explicitly written in the File Save window.

Except for the Jpeg format, which yields lossy compression, all the other formats are more or less equivalent. It is generally a bad idea to use Jpeg since it is lossy. It is better to use Targa and then convert it to Jpeg for web publishing purposes, keeping the original Targa.

Anyhow, for what concerns the other formats: TARGA raw is uncompressed Targa, uses a lot of disk space. PNG is Portable Network Graphics, a standard meant to replace old GIF inasmuch as it is lossless, but supports full true colour images. HamX is a self-developed 8 bits RLE (Run Length Encoded bitmap) format; it creates extremely compact files that can be displayed quickly. To be used only for the "Play" option. Iris is the standard SGI format, and Iris + Zbuffer is the same with added Zbuffer info.

Finally Ftype uses an "Ftype" file, to indicate that this file serves as an example for the type of graphics format in which Blender must save images. This method allows you to process 'colour map' formats. The colourmap data is read from the file and used to convert the available 24 or 32 bit graphics. If the option "RGBA" is specified, standard colour number '0' is used as the transparent colour. Blender reads and writes (Amiga) IFF, Targa, (SGI) Iris and CDinteractive (CDi) RLE colormap formats.

For what concerns animations:

- AVI Raw saves an AVI as uncompressed frames. Non-lossy, but huge files.
- AVI Jpeg saves an AVI as a series of Jpeg images. Lossy, smaller files but not as small as you can get with a better compression algorithm. Furthermore the AVI Jpeg format is not read by default by some players.
- AVI Codec saves an AVI compressing it with a codec. Blender automatically gets the list of your available codecs from the operating system and allows you to set its parameters. It is also possible to change it or change its settings, once selected, via the Set Codec button which appears (Figure 17-15).

Codec: Apple Pixlet Video			
QuickTim	e		Crop
Set codec Frs		/sec: 25>	
BW	RGB		RGBA

Figure 17-15. AVI Codec settings.

For an AVI animation it is also possible to set the frame rate (Figure 17-15) which, by default, is 25 frames per second.

# **Rendering Animations**

Relevant to Blender v2.31

The rendering of an animation is controlled via the Anim Panel (Figure 17-16).

🔻 Anim		
	ANIM	
	Do Sequence	
	Render Daemon	
	PLAY rt: 25	
	✓ Sta: 1    End: 250	

Figure 17-16. Animation rendering buttons.

The ANIM button starts the rendering. The first and last frames of the animation are given by the two NumButtons at the bottom (Sta: and End:), and by default are 1 and 250.

By default the 3D scene animation is rendered, to make use of the sequence editor the Do Sequence Tog Button must be selected.

▼ Output	
🖉 /tmp/	
🖉 //backbuf	
🖉 //ftype	
\$	
Backbuf	Edge Edge Settings >
DispView	DispWin Extensions

Figure 17-17. Animation location and extensions.

If an image format is chosen, on the other hand, a series of images named ####, ('####' being the pertinent frame number) is created in the directory. If the file name extension is needed, this is obtained by pressing the Extensions Tog Button (Figure 17-17).

**Complex animations:** Unless your animation is really simple, and you expect it to render in half an hour or less, it is always a good idea to render the animation as separate Targa frames rather than as an AVI file from the beginning.

This allows you an easy recovery if the power fails and you have to re-start the rendering, since the frames you have already rendered will still be there.

It is also a good idea since, if an error is present in a few frames, you can make corrections and re-render just the affected frames.

You can then make the AVI out of the separate frames with Blender's sequence editor or with an external program.

# **Motion Blur**

Relevant to Blender v2.31

Blender's animations are by default rendered as a sequence of *perfectly still* images.

This is unrealistic, since fast moving objects do appear to be 'moving', that is, blurred by their own motion, both in a movie frame and in a photograph from a 'real world camera'.

To obtain such a Motion Blur effect, Blender can be made to render the current frame and some more frames, in between the real frames, and merge them all together to obtain an image where fast moving details are 'blurred'.

OSA		MBLUR		
5	8	11	16	Bf: 0.500

Figure 17-18. Motion Blur buttons.

To access this option select the MBLUR button next to the OSA button in the Render Panel (Figure 17-18). This makes Blender render as many 'intermediate' frames as the oversampling number is set to (5, 8, 11 or 16) and accumulate them, one over the other, on a single frame. The number-button Bf: or Blur Factor defines the length of the shutter time as will be shown in the example below. Setting the OSA Button

#### Chapter 17. Rendering

is unnecessary since the Motion Blur process adds some antialiasing anyway, but to have a really smooth image OSA can be activated too. This makes each accumulated image have anti-aliasing.

To better grasp the concept let's assume that we have a cube, uniformly moving 1 Blender unit to the right at each frame. This is indeed fast, especially since the cube itself has a side of only 2 Blender units.

Figure 17-19 shows a render of frame 1 without Motion Blur, Figure 17-20 shows a render of frame 2. The scale beneath the cube helps in appreciating the movement of 1 Blender unit.



Figure 17-19. Frame 1 of moving cube without Motion Blur.



Figure 17-20. Frame 2 of moving cube without Motion Blur.

Figure 17-21 on the other hand shows the rendering of frame 1 when Motion Blur is set and 8 'intermediate' frames are computed. Bf is set to 0.5; this means that the 8 'intermediate' frames are computed on a 0.5 frame period starting from frame 1. This is very evident since the whole 'blurriness' of the cube occurs on half a unit before and half a unit after the main cube body.



Figure 17-21. Frame 1 of moving cube with Motion Blur, 8 samples, Bf=0.5.

Figure 17-22 and Figure 17-23 show the effect of increasing Bf values. A value greater than 1 implies a very 'slow' camera shutter.



Figure 17-22. Frame 1 of moving cube with Motion Blur, 8 samples, Bf=1.0.



Figure 17-23. Frame 1 of moving cube with Motion Blur, 8 samples, Bf=3.0.

Better results than those shown can be obtained by setting 11 or 16 samples rather than 8, but, of course, since as many *separate* renders as samples are needed a Motion Blur render takes that many times more than a non-Motion Blur one.

**Better Anti-Aliasing:** If Motion Blur is active, even if nothing is moving on the scene, Blender actually 'jitters' the camera a little between an 'intermediate' frame and the next. This implies that, even if OSA is off, the resulting images have nice Anti-Aliasing. An MBLUR obtained Anti-Aliasing is comparable to an OSA Anti-Aliasing of the same level, but generally slower.

This is interesting since, for very complex scenes where a level 16 OSA does not give satisfactory results, better results can be obtained using *both* OSA and MBlur. This way you have as many samples per frame as you have 'intermediate' frames, effectively giving oversampling at levels 25,64,121,256 if 5,8,11,16 samples are chosen, respectively.

### **Depth of Field**

*Relevant to Blender v2.31* 

Depth of Field (DoF) is an interesting effect in real world photography which adds a lot to CG generated images. It is also known as Focal Blur.

The phenomenon is linked to the fact that a real world camera can focus on a subject at a given distance, so objects closer to the camera and objects further away will be out of the focal plane, and will therefore be slightly blurred in the resulting photograph.

The amount of blurring of the nearest and furthest objects varies a lot with the focal length and aperture size of the lens and, if skilfully used, can give very pleasing effects.

Blender's renderer does not provide an automatic mechanism for obtaining DoF, but there are two alternative way to achieve it. One relies solely on Blender's internals, and will be described here. The other requires an external sequence plugin and will be outlined in the Sequence Editor Chapter.

The hack to obtain DoF in Blender relies on skilful use of the Motion Blur effect described before, making the Camera move circularly around what would be the aperture of the 'real world camera' lens, constantly pointing at a point where 'perfect' focus is desired.

Assume that you have a scene of aligned spheres, as shown on the the left of Figure 17-24. A standard Blender rendering will result in the image on the right of Figure 17-24, with all spheres perfectly sharp and in focus.



Figure 17-24. Depth of Field test scene.

The first step is to place an Empty (**SPACE**>>Add>>Empty) where the focus will be. In our case at the center of the middle sphere (Figure 17-25).



Figure 17-25. Setting the Focus Empty.

Then, assuming that your Camera is already placed in the correct position, place the cursor on the Camera (Select the Camera, **SHIFT-S**>>Curs->Sel) and add a NURBS circle (**SPACE**>>ADD>>Curve>>NURBS Circle).

Out of EditMode (**TAB**) scale the circle. This is very arbitrary, and you might want to re-scale it later on to achieve better results. Basically, the circle size is linked to the physical aperture size, or diaphragm, of your 'real world camera'. The larger the circle the narrower the region with perfect focus will be, and the more blurred near and far objects will be. The smaller the circle the less evident the DoF blurring will be.

Now make the circle track the Empty whith a constraint or the old Tracking as in Figure 17-26. Since the normal to the plane containing the circle is the local z-axis, you will have to set up tracking correctly so that the local z-axis of the circle points to the Empty and the circle is orthogonal to the line connecting its centre to the Empty.



Figure 17-26. NURBS circle tracking the focus Empty.

Select the Camera and then the circle and parent the Camera to the circle (**CTRL-P**). The circle will be the Path of the Camera so you can either use a normal parent relationship and then set the circle CurvePath Toggle Button on, or use a Follow Path Parent relationship.

With the circle still selected, open an IPO window select the Curve IPO type. The only available IPO is 'Speed'. **CTRL-LMB** twice at random in the IPO window to add an IPO with two random points. Then set these points numerically by using **NKEY** to Xmin and Ymin to 0, Xmax and Ymax to 1. To complete the IPO editing make it cyclic via the Curve>>Extend Mode>>Cyclic Menu entry. The final result should be as shown in Figure 17-27.



Figure 17-27. Speed IPO for the NURBS circle path.

With these settings we have effectively made the Camera circle around its former position along the NURBS circle path in exactly 1 frame. This makes the Motion Blur option take slightly different views of the scene and create the Focal Blur effect in the end.

There is still one more setting to perform. First select the Camera and then the focal Empty, and make the Camera track the Empty the way you prefear. The Camera should now track the Empty, as in Figure 17-28.



Figure 17-28. Camera tracking the focal Empty.

If you press **ALT-A** now you won't see any movement because the Camera does exactly one full circle path in each frame, so it appears to be still, nevertheless the Motion Blur engine will detect these moves.

The last touch is then to go to the rendering buttons window (F10) and select the MBLUR button. You most probably don't need the OSA button active, since Motion Blur will implicitly do some antialiasing. It is strongly recommended that you set the Motion Blur factor to 1, since this way you will span the entire frame for blurring, taking the whole circle length. It is also necessary to set the oversamples to the maximum level (16) for best results (Figure 17-29).
OSA				MBLUR
5	8	11	16	Bf: 1.000

#### Figure 17-29. Motion Blur settings.

A rendering (F12) will yield the desired result. This can be much slower than a non-DoF rendering since Blender effectively renders 16 images and then merges them. Figure 17-30 shows the result, to be compared with the one in Figure 17-24. It must be noted that the circle has been scaled much less to obtain this picture than has been shown in the example screenshots. These latter were made with a large radius (equal to 0.5 Blender units) to demonstrate the technique better. On the other hand, Figure 17-30 has a circle whose radius is 0.06 Blender units.



Figure 17-30. Motion Blur final rendering.

This technique is interesting and with it it's pretty easy to obtain small degrees of Depth of Field. For big Focal Blurs it is limited by the fact that it is not possible to have more than 16 oversamples.

# **Cartoon Edges**

## Relevant to Blender v2.31

Blender's new material shaders, as per version 2.28, include nice toon diffuse and specular shaders.

By using these shaders you can give your rendering a comic-book-like or mangalike appearance, affecting the shades of colours, as you may be able to appreciate in Figure 17-31.



Figure 17-31. A scene with Toon materials.

The effect is not perfect since real comics and manga also usually have china ink outlines. Blender can add this feature as a post-processing operation.

To access this option select the Edge button in the Output Panel of the Rendering (F10) Buttons (Figure 17-32). This makes Blender search for edges in your rendering and add an 'outline' to them.

▼ Output	
🖉 /tmp/	
🖉 //backbuf	
🖉 //ftype	
\$	
Backbuf	Edge Edge Settings >
DispView	DispWin Extensions

Figure 17-32. Toon edge buttons.

Before repeating the rendering it is necessary to set some parameters. The Edge Settings opens a window to set these (Figure 17-33).

		🔹 AntiS	hift O	
	≪Eint: 0 ►	Shift	AI	
	R 0.00	)0 🛏		
	G 0.00	00 🛏		三伯
	B 0.00	)0 🛏		
Edge	dge Settings		/	

Figure 17-33. Toon edge settings.

In this window it is possible to set the edge colour, which is black by default, and its intensity, Eint which is an integer ranging from 0 (faintest) to 255 (strongest). The other buttons are useful if the Unified render is used (see next section).

Figure 17-34 shows the same image as Figure 17-31 but with toon edges enabled, of black colour and maximum intensity (Eint=255).



Figure 17-34. Scene re-rendered with toon edge set.

# **The Unified Renderer**

Relevant to Blender v2.31

A less well known feature of Blender is the Unified Renderer button in the bottom right corner of the Rendering Buttons Format Panel (Figure 17-35).

▼ Format	
Game framing settings >>	PAL
Size X: 640 Size V: 512	NTSC
01267.040 01267.012	Default
AspX: 1 🕨 🔹 AspV: 1 🕨	Preview
	PC
	PAL 16:9
PNG   Crop	PANO
<ul> <li>▲ Quality: 100 ►</li> <li>▲ Frs/sec: 25 ►</li> </ul>	FULL
BW RGB RGBA	Unified Render

Figure 17-35. The Unified Renderer button.

Blender's *default* renderer is highly optimized for speed. This has been achieved by subdividing the rendering process into several passes. First the 'normal' materials are handled. Then Materials with transparency (Alpha) are taken into account. Finally Halos and flares are added.

This is fast, but can lead to less than optimum results, especially with Halos. The Unified Renderer, on the other hand, renders the image in a single pass. This is slower, but gives better results, especially for Halos.

Furthermore, since transparent materials are now rendered together with the conventional ones, Cartoon Edges can be applied to them too, by pressing the All button in the Edge Setting dialog.

If the Unified Renderer is selected an additional group of buttons appears in the Output Panel (Figure 17-36).

▼ Output
💋 //render/
\$
Backbuf Edge Edge Settings
Gamma:2.0 Post process >
DispView DispWin Extension

Figure 17-36. Unified Renderer additional buttons.

The Gamma slider is related to the OSA procedure. Pixel oversamples are blended to generate the final rendered pixel. The conventional renderer has a Gamma=1, but in the Unified Renderer you can vary this number.

The Post process button makes a dialog box appear (Figure 17-37). From this you can control three kinds of post processing: the Add slider defines a constant quantity to be added to the RGB colour value of each pixel. Positive values make the image uniformly brighter, negative uniformly darker.

		+
	Add:0.000	
	Mul:1.000	
Edge	Gamma:1.000	
	Post process >	🔹 Xparts: 1 🕨

Figure 17-37. Unified Renderer postprocess submenu.

The Mul slider defines a value by which all RGB values of all pixels are multiplied. Values greater than 1 make the image brighter, smaller than 1 make the image darker.

The Gamma slider does the standard gamma contrast correction of any paint program.

# Preparing your work for video

#### Relevant to Blender v2.31

Once you have mastered the trick of animation you will surely start to produce wonderful animations, encoded with your favourite codecs, and possibly you'll share them on the Internet with the rest of the community.

But, sooner or later, you will be struck by the desire of building an animation for Television, or maybe burning you own DVDs.

To spare you some disappointment, here are some tips specifically targeted at Video preparation. The first and principal one is to remember the double dashed white lines in the camera view!

If you render for PC then the whole rendered image, which lies within the *outer* dashed rectangle will be shown. For Television some lines and some part of the lines will be lost due to the mechanics of the electron beam scanning in your TV's cathode ray tube. You are guaranteed that what is within the *inner* dashed rectangle in camera view will be visible on the screen. Everything within the two rectangles may or may not be visible, depending on the given TV set you watch the video on.

Furthermore the rendering size is strictly dictated by the TV standard. Blender has three pre-set settings for your convenience:

- PAL 720x576 pixels at 54:51 aspect ratio.
- NTSC 720x480 pixels at 10:11 aspect ratio.
- PAL 16:9 720x576 at 64:45 aspect ratio, for 16:9 widescreen TV renderings.

Please note the "Aspect Ratio" stuff. TV screens do *not* have the square pixels which Computer monitors have, their pixels are somewhat rectangular, so it is necessary to generate *pre-distorted* images which will look bad on a computer but which will display nicely on a TV set.

## **Colour Saturation**

Most video tapes and video signals are not based on the RGB model but on the YUV (or YCrCb) model in Europe and YIQ in the USA, this latter being quite similar to the former. Hence some knowledge of this is necessary too.

The YUV model sends information as 'Luminance', or intensity (Y) and two 'Crominance' signals, red and blue. Actually a Black and White TV set shows only luminance, while colour TV sets reconstruct colour from Crominances. It is:

Y = 0.299R + 0.587G + 0.114B

U = Cr = R-Y

V = Cb = B-Y

Whereas a standard 24 bit RGB picture has 8 bits for each channel, to keep bandwidth down, and considering that the human eye is more sensitive to luminance than to crominance, the luminance signal is sent with more bits than the two crominance signals.

This results in a smaller dynamic of colours, in Video, than that which you are used to on Monitors. You hence have to keep in mind not all colours can be correctly displayed. Rule of thumb is to keep the colours as 'greyish' or 'unsaturated' as possible, this can be roughly converted in keeping the dynamics of your colours within 0.8.

In other words the difference between the highest RGB value and the lowest RGB value should not exceed 0.8 ([0-1] range) or 200 ([0-255] range).

This is not strict, something more than 0.8 is acceptable, but a RGB=(1.0,0,0) material will be very ugly.

## **Rendering to fields**

The TV standards prescribe that there should be 25 frames per second (PAL) or 30 frames per second (NTSC). Since the phosphorous of the screen does not maintain luminosity for very long, this could produce a noticeable flickering. To minimize this TVs do not represent frames as a Computer does but rather represents half-frames, or *fields* at a double refresh rate, hence 50 half frames per second on PAL and 60 half frames per second on NTSC. This was originally bound to the frequency of power lines in Europe (50Hz) and the US (60Hz).

In particular fields are "interlaced" in the sense that one field presents all the even lines of the complete frame and the subsequent field the odd ones.

Since there is a non-negligible time difference between each field (1/50 or 1/60 of a second) merely rendering a frame the usual way and splitting it into two half frames does not work. A noticeable jitter of the edges of moving objects would be present.

▼ Render					
BENDEB	Shadow	EnvMap			
	Pano	Radio			
OSA MBLUR	10	0%			
5 8 11 16 Bf: 1.000	75% 50	% 25%			
Xparts: 1  Vparts: 1  Fields					
Sky Premul Key Border Gamma					

## Figure 17-38. Field Rendering setup.

To optimally handle this issue Blender allows for field rendering. When the Fields button in the Render Panel is pressed (Figure 17-38), Blender prepares each frame in two passes. On the first it renders only the even lines, then it *advances in time by half a time step* and renders all the odd lines.



## Figure 17-39. Field Rendering result.

This produces odd results on a PC screen (Figure 17-39) but will show correctly on a TV set.

One of the two buttons next to the Fields button forces the rendering of Odd fields first (Odd) and the other disables the half-frame time step between fields (x).

**Setting up the correct field order:** Blender's default setting is to produce Even fields *before* Odd fields, this complies with European PAL standards. Odd fields are scanned first on NTSC.

Of course, if you make the wrong selection things are even worse than if no Field rendering at all was used.

Chapter 17. Rendering

# Chapter 18. Radiosity

Most rendering models, including ray-tracing, assume a simplified spatial model, highly optimised for the light that enters our 'eye' in order to draw the image. You can add reflection and shadows to this model to achieve a more realistic result. Still, there's an important aspect missing! When a surface has a reflective light component, it not only shows up in our image, it also shines light at surfaces in its neighbourhood. And vice-versa. In fact, light bounces around in an environment until all light energy is absorbed (or has escaped!).

Re-irradiated light carries information about the object which has re-irradiated it, notably colour. Hence not only the shadows are 'less black' because of re-irradiated light, but also they tend to show the colour of the nearest, brightly illuminated, object. A phenomenon often referred to as 'colour leaking' (Figure 18-1).



Figure 18-1. Radiosity example

In closed environments, light energy is generated by 'emitters' and is accounted for by reflection or absorption of the surfaces in the environment. The rate at which energy leaves a surface is called the 'radiosity' of a surface. Unlike conventional rendering methods, Radiosity methods first calculate all light interactions in an environment in a view-independent way. Then, different views can be rendered in real-time.

In Blender, since version 2.28, Radiosity is both a rendering and a modelling tool. This means that you can enable Radiosity within a rendering or rather use Radiosity to paint vertex colours and vertex lights of your meshes, for later use.

# The Blender Radiosity method

## Relevant to Blender v2.31

First, some theory! You can skip to the next section if you like, and come back here if questions arise.

During the late eighties and early nineties Radiosity was a hot topic in 3D computer graphics. Many different methods were developed, the most successful of these solutions were based on the "progressive refinement" method with an "adaptive subdivision" scheme. And this is what Blender uses.

To be able to get the most out of the Blender Radiosity method, it is important to understand the following principles:

#### • Finite Element Method

Many computer graphics or simulation methods assume a simplification of reality with 'finite elements'. For a visually attractive (and even scientifically proven) solution, it is not always necessary to dive into a molecular level of detail. Instead, you can reduce your problem to a finite number of representative and well-described elements. It is a common fact that such systems quickly converge into a stable and reliable solution.

The Radiosity method is a typical example of a finite element method inasmuch as every face is considered a 'finite element' and its light emission considered as a whole.

#### • Patches and Elements

In the Radiosity universe, we distinguish between two types of 3D faces:

*Patches*. These are triangles or squares which are able to *send energy*. For a fast solution it is important to have as few of these patches as possible. But, to speed things up the energy is modelled as if it were radiated by the Patch's centre; the size of the patches should then be small enough to make this a realistic energy distribution. (For example, when a small object is located above the Patch centre, all energy the Patch sends is obscured by this object, even if the patch is larger! This patch should be subdivided in smaller patches).

*Elements*. These are the triangles or squares which *receive energy*. Each Element is associated with a Patch. In fact, Patches are subdivided into many small Elements. When an Element receives energy it absorbs part of it (depending on its colour) and passes the remainder to the Patch, for further radiation. Since the Elements are also the faces that we display, it is important to have them as small as possible, to express subtle shadow boundaries and light gradients.

#### Progressive Refinement

This method starts with examining all available Patches. The Patch with the most 'unshot' energy is selected to shoot all its energy to the environment. The Elements in the environment receive this energy, and add this to the 'unshot' energy of their associated Patches. Then the process starts again for the Patch *now* having the most unshot energy. This continues for all the Patches until no energy is received anymore, or until the 'unshot' energy has converged below a certain value.

#### • The hemicube method

The calculation of how much energy each Patch gives to an Element is done through the use of 'hemicubes'. Exactly located at the Patch's center, a hemicube (literally 'half a cube') consist of 5 small images of the environment. For each pixel in these images, a certain visible Element is color-coded, and the transmitted amount of energy can be calculated. Especially with the use of specialized hardware the hemicube method can be accelerated significantly. In Blender, however, hemicube calculations are done "in software".

This method is in fact a simplification and optimisation of the 'real' Radiosity formula (form factor differentiation). For this reason the resolution of the hemicube (the number of pixels of its images) is approximate and its careful setting is important to prevent aliasing artefacts.

#### Adaptive subdivision

Since the size of the patches and elements in a Mesh defines the quality of the Radiosity solution, automatic subdivision schemes have been developed to define

the optimal size of Patches and Elements. Blender has two automatic subdivision methods:

1. *Subdivide-shoot Patches*. By shooting energy to the environment, and comparing the hemicube values with the actual mathematical 'form factor' value, errors can be detected that indicate a need for further subdivision of the Patch. The results are smaller Patches and a longer solving time, but a higher realism of the solution.

2. *Subdivide-shoot Elements*. By shooting energy to the environment, and detecting high energy changes (gradients) inside a Patch, the Elements of this Patch are subdivided one extra level. The results are smaller Elements and a longer solving time and maybe more aliasing, but a higher level of detail.

• Display and Post Processing

Subdividing Elements in Blender is 'balanced', that means each Element differs a maximum of '1' subdivide level with its neighbours. This is important for a pleasant and correct display of the Radiosity solution with Gouraud shaded faces. Usually after solving, the solution consists of thousands of small Elements. By filtering these and removing 'doubles', the number of Elements can be reduced significantly without destroying the quality of the Radiosity solution. Blender stores the energy values in 'floating point' values. This makes settings for dramatic lighting situations possible, by changing the standard multiplying and gamma values.

• Radiosity for Modelling

The final step can be replacing the input Meshes with the Radiosity solution (button Replace Meshes). At that moment the vertex colours are converted from a 'floating point' value to a 24 bits RGB value. The old Mesh Objects are deleted and replaced with one or more new Mesh Objects. You can then delete the Radiosity data with Free Data. The new Objects get a default Material that allows immediate rendering. Two settings in a Material are important for working with vertex colours:

*VColPaint*. This option treats vertex colours as a replacement for the normal RGB value in the Material. You have to add Lamps in order to see the Radiosity colours. In fact, you can use Blender lighting and shadowing as usual, and still have a neat Radiosity 'look' in the rendering.

*VColLight*. The vertexcolors are added to the light when rendering. Even without Lamps, you can see the result. With this option, the vertex colours are premultiplied by the Material RGB colour. This allows fine-tuning of the amount of 'Radiosity light' in the final rendering.

As with everything in Blender, Radiosity settings are stored in a datablock. It is attached to a Scene, and each Scene in Blender can have a different Radiosity 'block'. Use this facility to divide complex environments into Scenes with independent Radiosity solvers.

# **Radiosity Rendering**

#### Relevant to Blender v2.31

Let's assume you have a scene ready, and that you want to render it with the Radiosity Rendering. The first thing to grasp when doing Radiosity is that *no Lamps are necessary*, but some meshes with an Emit material property greater than zero are required, since these will be the light sources.

#### Chapter 18. Radiosity

You can build the test scene shown in Figure 18-1, it is rather easy. Just make a big cube for the room, give different materials to the side walls, add a cube and a stretched cube within it, and add a plane with a non-zero Emit value next to the roof, to simulate the area light (Figure 18-2).

You assign Materials as usual to the input models. The RGB value of the Material defines the Patch colour. The 'Emit' value of a Material defines if a Patch is loaded with energy at the start of the Radiosity simulation. The 'Emit' value is multiplied with the area of a Patch to calculate the initial amount of unshot energy.

**Emitting faces:** Check the number of 'emitters' on Blender console! If this is zero nothing interesting can happen. You need at least one emitting patch to have light and hence a solution.



Figure 18-2. Set-up for Radiosity test.

When assigning materials be sure that all of them have the Radio toggle on to enable the Shader Panel of the Material subcontext buttons (Figure 18-3).

▼ Shaders	
Lambert 🗢 Ref 0.800	Halo
	Traceable
	Shadow
CookTorr 🛊 Spec 0.500 🔳	Radio
Hard:50	Wire
·,	ZTransp
	Env
	OnlyShad
	No Mist
Add 0.00	Zinvert

Figure 18-3. Radiosity enabled material.

Please note that the light emission is governed by the direction of the normals of a mesh, so the light emitting plane should have a *downward* pointing normal and the *outer* cube (the room) should have the normals pointing inside, (flip them!).

Switch to the Radiosity Sub-context of the Shading Context. The Panels, shown in Figure 18-4, are two: Radio Rendering which governs Radiosity when used as a rendering tool (present case) and Radio Tool, which governs Radiosity as a modelling tool (next section).

	Collect	Meshes	Free Ra	adio Data
Max Iterations: 300	Replace	Meshes	Add nev	v Meshes
Mult: 30.00 🔸 🔹 Gamma: 2.000 🕨	Wire	Solid	Gour	ShowLim
vergence:0.100	EIMax: 100	«ElMin: 20»	PaMax: 500	PaMin: 20

Figure 18-4. Radiosity buttons for radiosity rendering.

The buttons define:

- Hemires: The hemicube resolution; the color-coded images used to find the Elements that are visible from a 'shoot Patch', and thus receive energy. Hemicubes are not stored, but are recalculated each time for every Patch that shoots energy. The 'Hemires' value determines the Radiosity quality and adds significantly to the solving time.
- Max Iterations: The maximum number of Radiosity iterations. If set to zero Radiosity will go on until the convergence criterion is met. You are strongly advised to set this to some non-zero number, usually greater than 100.
- Mult:, Gamma: The colourspace of the Radiosity solution is far more detailed than can be expressed with simple 24 bit RGB values. When Elements are converted to faces, their energy values are converted to an RGB colour using the Mult and Gamma values. With the Mult value you can multiply the energy value, with Gamma you can change the contrast of the energy values.
- Convergence: When the amount of unshot energy in an environment is lower than this value, the Radiosity solving stops. The initial unshot energy in an environment is multiplied by the area of the Patches. During each iteration, some of the energy is absorbed, or disappears when the environment is not a closed volume. In Blender's standard coordinate system a typical emitter (as in the example files) has a relatively small area. The convergence value is divided by a factor of 1000 before testing for that reason.

Set the Max Iterations: to 100 and turn to the Scene Context and Render Subcontext (F10).

Locate the Radio Tog Button (Figure 18-5) in the Render Panel and set it 'on' to enable Radiosity, then Render! (F12).

▼ Render		
DENDER	Shadow	EnvMap
	Pano	Radio
OSA MBLUR	100	)%
5 8 11 16 Bf: 0.500	75% 50	% 25%
≪ Xparts: 1 →	Fields	Od X
Sky Premul Key	Border	Gamma

Figure 18-5. Enabling Radiosity in the Rendering Buttons.

The rendering will take more time than usual, in the console you will notice a counter going up. The result will be quite poor (Figure 18-6, left) because the automatic radiosity render does not do adaptive refinement!

Select all meshes, one after the other, and in EditMode subdivide them at least three times. The room, which is much bigger than the other meshes, you can even subdivide four times. Set the Max Iterations a bit higher, 300 or more. Try Rendering again (F12). This time the rendering will take even longer but the results will be much nicer, with soft shadows and colour leakage (Figure 18-6, right).



Figure 18-6. Radiosity rendering for coarse meshes (left) and fine meshes (right).

**Note:** In the Radiosity Rendering Blender acts as for a normal rendering, this means that textures, Curves, Surfaces and even Dupliframed Objects are handled correctly.

# **Radiosity as a Modelling Tool**

Relevant to Blender v2.31

Radiosity can be used also as a Modelling tool for defining Vertices colours and lights. This can be very useful if you want to make further tweaks to your models, or you want to use them in the Game Engine. Furthermore the Radiosity Modelling allows for Adaptive refinement, whereas the Radiosity Rendering does not!

There are few important points to grasp for practical Radiosity Modelling:

Only Meshes in Blender are allowed as input for Radiosity Modelling. This because the process generates Vertex colours... and so there must be vertices. It is also important to realize that *each* face in a Mesh becomes a Patch, and thus a potential energy emitter and reflector. Typically, large Patches send and receive more energy than small ones. It is therefore important to have a well-balanced input model with Patches large enough to make a difference! When you add extremely small faces, these will (almost) never receive enough energy to be noticed by the "progressive refinement" method, which only selects Patches with large amounts of unshot energy.

**Non-mesh Objects:** *Only Meshes* means that you have to convert Curves and Surfaces to Meshes (**CTRL-C**) before starting the Radiosity solution!

## **Phase 1: Collect Meshes**

All selected and visible Meshes in the current Scene are converted to Patches as soon as the Collect Meshes button of the Radio Tool Panel is pressed (Figure 18-4). As a result a new Panel, Calculation, appears. Blender has now entered the Radiosity Modelling mode, and other editing functions are blocked until the newly created button Free Data has been pressed. The Phase text above the buttons now says Init and shows the number of Patches and Elements.

After the Meshes are collected, they are drawn in a pseudo lighting mode that clearly differs from the normal drawing.

The Radio Tool Panel (Figure 18-7) has three Radio Buttons: Wire, Solid, Gour. These are three drawmode options independent of the indicated drawmode of a 3DWindow. Gouraud display is only performed after the Radiosity process has started. Press the Gour button, to have smoother results on curved surfaces.

▼ Radio Tool					
Collect	Meshes	Free Radio Data			
Replace	Meshes	Add new Meshes			
Wire	Solid	Gour	ShowLim Z		
EIMax: 100	EIMax: 100 EIMin: 20		00 PaMin: 200		
Limit Subdivide					

Figure 18-7. Gourad button

# **Phase 2: Subdivision limits**

Blender offers a few settings to define the minimum and maximum sizes of Patches and Elements in the Radio Tools and Calculation Panels (Figure 18-8).

▼ Radio Tool						
Collec	t Meshes	Free	Free Radio Data			
Replac	e Meshes	Add	Add new Meshes			
Wire	Solid	Gour	ShowLim Z			
EIMax: 100	ElMax: 100 ElMin: 20		00 PaMin: 200			
Limit Subdivide						

Figure 18-8. Radiosity Buttons for Subdivision

Limit Subdivide With respect to the values "PaMax" and "PaMin", the Patches are subdivided. This subdivision is also automatically performed when a "GO" action has started.

PaMax, PaMin, ElMax, ElMin The maximum and minimum size of a Patch or Element. These limits are used during all Radiosity phases. The unit is expressed in 0.0001 of the boundbox size of the entire environment. Hence, with default 500 and 200 settings maximum and minimum Patch size 0.05 of the entire model (1/20) and 0.02 of the entire model (1/50).

ShowLim, z This option visualizes the Patch and Element limits. By pressing the z option, the limits are drawn rotated differently. The white lines show the Patch limits, cyan lines show the Element limits.

# **Phase 3: Adaptive Subdividing**

Last settings before starting the analysis (Figure 18-9).

▼ Calculation					
GO					
🔹 SubSh Patch: 1 🕞	🔹 SubSh Element: 2 🔶				
Subdiv Shoot Element	Subdiv Shoot Patch				
	✓ Max Subdiv Shoot: 1 ▶				
FaceFilter Element Filter					
RemoveDoubles < Lim: 0 >					

Figure 18-9. Radiosity Buttons

MaxEl The maximum allowed number of Elements. Since Elements are subdivided automatically in Blender, the amount of used memory and the duration of the solving time can be controlled with this button. As a rule of thumb 20,000 elements take up 10 Mb memory.

Max Subdiv Shoot The maximum number of shoot Patches that are evaluated for the "adaptive subdivision" (described below). If zero, all Patches with 'Emit' value are evaluated.

Subdiv Shoot Patch By shooting energy to the environment, errors can be detected that indicate a need for further subdivision of Patches. The subdivision is performed only once each time you call this function. The results are smaller Patches and a longer solving time, but a higher realism of the solution. This option can also be automatically performed when the GO action has started.

Subdiv Shoot Element By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are selected to be subdivided one extra level. The subdivision is performed only once each time you call this function. The results are smaller Elements and a longer solving time and probably more aliasing, but a higher level of detail. This option can also be automatically performed when the GO action has started.

SubSh P The number of times the environment is tested to detect Patches that need subdivision.

SubSh E The number of times the environment is tested to detect Elements that need subdivision.

**Note:** Hemires, Convergence and Max iterations in the Radio Render Panel are still active and have the same meaning as in Radiosity Rendering.

GO With this button you start the Radiosity simulation. The phases are:

- 1. *Limit Subdivide*. When Patches are too large, they are subdivided.
- 2. *Subdiv Shoot Patch*. The value of SubSh P defines the number of times the Subdiv Shoot Patch function is called. As a result, Patches are subdivided.
- 3. *Subdiv Shoot Elem.* The value of SubSh E defines the number of times the Subdiv Shoot Element function is called. As a result, Elements are subdivided.
- 4. *Subdivide Elements.* When Elements are still larger than the minimum size, they are subdivided. Now, the maximum amount of memory is usually allocated.
- 5. *Solve.* This is the actual 'progressive refinement' method. The mouse pointer displays the iteration step, the current total of Patches that shot their energy in the environment. This process continues until the unshot energy in the environment is lower than the Convergence value or when the maximum number of iterations has been reached.
- 6. *Convert to faces.* The elements are converted to triangles or squares with 'anchored' edges, to make sure a pleasant not-discontinue Gouraud display is possible.

This process can be terminated with ESC during any phase.

## Phase 4: Editing the solution

Once the Radiosity solution has been computed there are still some actions to take (Figure 18-10).

#### Chapter 18. Radiosity

V Ra		V Radio Tool				
	Hemires:300	Collect Meshes Free Radio Data			GO	
	< Max Iterations: 300 >	Benjace Meshes	Add new Meshes	4	SubSh Patch: 1 →	< SubSh Element: 2 →
	Mult: 20.00 b Comma: 2.000 b	Highlade meanes Aud new meanes		Ιſ	Subdiv Shoot Element	Subdiv Shoot Patch
	Camina 2.000 P	Wire Solid	Gour ShowLim Z		MaxEl: 10000	< Max Subdiv Shoot: 1 ►
	Convergence:0.100	EIMax: 100 (EIMin: 20)	PaMax: 500 PaMin: 200		Face Filter	Element Filter
		Limit Subdivide			RemoveDouble	s ← Lim: 0 →

Figure 18-10. Radiosity post process.

Element Filter This option filters Elements to remove aliasing artifacts, to smooth shadow boundaries, or to force equalized colours for the RemoveDoubles option.

RemoveDoubles When two neighbouring Elements have a displayed colour that differs less than the limit specified in the Lim NumButton, the Elements are joined. The Lim value used here is expressed in a standard 8 bits resolution; a color range from 0 - 255.

FaceFilter Elements are converted to faces for display. A FaceFilter forces an extra smoothing in the displayed result, without changing the Element values themselves.

Mult:, Gamma: These NumButtons have the same meaning as in Radiosity Rendering.

Add New Meshes The faces of the current displayed Radiosity solution are converted to Mesh Objects with vertex colours. A new Material is added that allows immediate rendering. *The input-Meshes remain unchanged*.

Replace Meshes As previous, but the input-Meshes are removed.

Free Radio Data All Patches, Elements and Faces are freed in Memory. You must always perform this action after using Radiosity to be able to return to normal editing.

# **Radiosity Juicy example**

*Relevant to Blender v2.31* 

To get definitely away from dry theory and shows what Radiosity Modelling can really achieve let's look at an example.

This will actually show you a true Global Illumination scene, with smoother results than the 'Dupliverted Spot Lights' technique shown in the Lighting Chapter to attain something like Figure 18-11.



Figure 18-11. Radiosity rendered Cylon Raider.

# Setting up

We have only two elements in the scene at start up: a Raider (if you remember some Sci-Fi Movie...) and a camera. The Raider has the default grey material, except for the main cockpit windows which are black. For this technique, we will not need any lamps.

The first thing that we will want to add to the scene is a plane. This plane will be used as the floor in our scene. Resize the plane as shown in Figure 18-12 and place it just under the Raider. Leave a little space between the plane and the Raider bottom. This will give us a nice "floating" look.



Figure 18-12. Add a plane

Next, you will want to give the plane a material and select a colour for it. We will try to use a nice blue. You can use the settings in Figure 18-13 for it.

▼ Material		
⇒ MA:Mate	erial.001	1 3 <b>X ∰</b> F 🕇 🗡
ME:Plane		OB ME
VCol Light	VCol	Paint TexFace Shadeless
	Col	R 0.081
	Spe	G 0.361
	Mir	
		Alpha 1.000
RGB HSV	DYN	SpecTra 0.00

**Figure 18-13. Plane colour** 

# The Sky Dome

We want to make a GI rendering, so the next thing that we are going to add is an icosphere. This sphere is going to be our light source instead of the typical lamps. What we are going to do is use its faces as *emitters* that will project light for us in multiple directions instead of in one direction as with a typical, single, lamp. This will give us the desired effect.

To set this up, add an icosphere with a subdivision of 3. While still in EditMode, use the **BKEY** select mode to select the lower portion of the sphere and delete it. This will leave us with our dome. Resize the dome to better fit the scene and match it up with your plane. It should resemble Figure 18-14.



Figure 18-14. Sky dome.

Next, we want to make sure that we have all the vertices of the dome selected and then click on the EditButtons (F9) and select Draw Normals. This allows us to see in which direction the vertices are "emitting". By default it will be outside, so hit the Flip Normals button, which will change the vertex emitter from projecting outward to projecting inward in our dome (Figure 18-15).

Beauty	Subdivide	Fract Subd		Ce	entre	]
Noise	Hash	Xsort				
To Sphere	Smooth	Split		Hide	Reveal	
Flip Normals	Rem Double	Limit: 0.009		Selec	t Swan	1
	Extrude				. on op	J
Screw	Spin	Spin Dup		<ul> <li>NSize</li> <li>Draw I</li> </ul>	: 0.100	
✓ Degr: 90 ►	🔹 Steps: 9 🕨	🔹 Turns: 1 🕨		Draw	Faces	
Keep (	Driginal	Clockwise		Draw	Edges	
Extrude D	)up 🔄 Ot	fset: 1.000 🔺	i i	All e	edges	

## Figure 18-15. Flipping the normals.

Now that we have created our dome, we need a new material. When you create the material for the dome change the following settings in the MaterialButtons (F5):

Add = 0.000 Ref = 1.000 Alpha = 1.000 Emit = 0.020

The Emit slider here is the key. This setting controls the amount of light "emitted" from our dome. 0.020 is a good default. Remember that the dome is the bigger part of the scene! you don't want too much light! But you can experiment with this setting to get different results. The lower the setting here though the longer the "solve" time later. (Figure 18-16).

🔻 Material	▼ Shaders
Compare and the second s	Lambert + Ref 1.000 Halo Traceabl
VCol Light VCol Pain TexFace Shadeless	CookTorr = Spec 0.50 Radio Hard:50 Wire
Spe         G 0.800            B 0.800	ZTransp Env OnlySha
Alpha 1.000         I           RGB HSV DYN         SpecTra 0.0	Amb 0.5         Emit 0.020         No Mist           Add 0.0          Zoffs: 0.000         ZInvert

Figure 18-16. Sky dome material.

At this point we have created everything that we need for our scene. The next step will be to alter the dome and the plane from "double-sided" to "single-sided". To achieve this, we will select the dome mesh and then go back to the EditButtons (F9). Click the Double Sided button and turn it off (Figure 18-17). Repeat this process for the Plane.

🔻 Mesh		
Auto Smooth	<ul> <li>Decin</li> </ul>	nator: 160 🔹 🕨
✓ Degr: 30 ▶	Apply	Cancel
SubSurf	TexMesh:	
Subdiv: 1> <1>		Centre
		Centre New
Sticky Make		Centre Cursor
VertCol Make	SlowerDra	Double Sided
TexFac Make	FasterDra	No V.Normal Fl

Figure 18-17. Setting Dome and plane 'single sided'.

# The Radiosity solution

Now the next few steps are the heart and soul of Global Illumination. Go to side view with **NUM 3** and use **AKEY** to select all of the meshes in our scene. Next hold **SHIFT** and double click on your camera. We do not want this selected. It should look similar to Figure 18-18.



Figure 18-18. Selecting all Meshes.

After selecting the meshes, go to camera view with **NUM 0** and then turn on shaded mode with **ZKEY** so we can see inside our dome.

Now select the Shading Context (F5) and the Radiosity Buttons Sub-context (...). In the Radio Tool Panel, click the Collect Meshes button. You should notice a change in the colours in your view. It should look similar to Figure 18-19.



Figure 18-19. Preparing the Radiosity solution.

Next, to keep the Raider smooth like our original mesh, we will want to change from Solid to Gour. This will give our Raider its nice curves back, in the same way Set Smooth would in the EditButtons. You'll also need to change the Max Subdiv Shoot to 1 (Figure 18-20). Do not forget this step!

▼ Radio Tool		<ul> <li>Calculation</li> </ul>	
Collect Meshes	Free Radio Data	GO	
Replace Meshes	Add new Meshes	SubSh Patch: 1 🔸 🔍 SubSh Element	:2 ⊧
neplace meshes	Hud new Weshes	Subdiv Shoot Element Subdiv Shoot Pa	atch
Wire Solid	Gour ShowLim Z	MaxEl: 10000 Max Subdiv Sho	ot: 1
IMax: 100 EIMin: 20	Max: 500 aMin: 200	FaceFilter Element Filte	r
Limit Su	ıbdivide	RemoveDoubles 🛛 🗸 Lim	:0 >

Figure 18-20. Radiosity settings.

After you have set Gour and Max Subdiv Shoot, click Go and wait. Blender will then begin calculating the emit part of the dome, going face by face, thus "solving" the render. As it does this, you will see the scene change as more and more light is added to the scene and the meshes are changed. You will also notice that the cursor in Blender changes to a counter much like if it were an animation. Let Blender run, solving the Radiosity problem.

Letting Blender go to somewhere between 50-500 depending on the scene can do, for most cases. The solving time depends on you and how long you decide to let it run... remember you can hit **ESC** at any time to stop the process. This is an area that can be experimented with for different results. This can take from 5 to 10 minutes and your system speed will also greatly determine how long this process takes. Figure 18-21 is our Raider after 100 iterations.



Figure 18-21. Radiosity solution.

After hitting the **ESC** key and stopping the solution, click Replace Meshes (or Add New Meshes) and then Free Radio Data. This finalizes our solve and replaces our previous scene with the new solved Radiosity scene.

**Note:** Adding rather than Replacing meshes is a form of Undo. You still have old meshes and you can re-run Radiosity again! But you must move these new meshes to a new layer and hide the old layers before rendering!

Now we are ready for F12 and render (Figure 18-22).



Figure 18-22. Rendering of the radiosity solution.

# Texturing

There you go folks! You now have a very clean looking render with soft 360 degree lighting using Radiosity. Very nice... But the next thing we want to do is add textures to the mesh. So go back to our main screen area.

Now try selecting your mesh and you will notice that it selects not only the Raider but the plane and dome as well. That is because Radiosity created a new *single* mesh through the solution process. To add a texture though, we only want the Raider.

So, select the mesh and then go into EditMode. In EditMode we can delete the dome and plane since they are no longer needed. You can use the **LKEY** to select the proper vertices and press **XKEY** to delete them. Keep selecting and deleting until you are left with only the Raider. It should look like in Figure 18-23. If we were to render it now with **F12**, we would get just a black background and our Raider. This is nice... but again, we want textures!



Figure 18-23. The Raider's mesh.

## Chapter 18. Radiosity

To add textures to mesh, we must separate out the areas that we are going to apply materials and textures to. For the Raider, We want to add textures to the wings and mid-section. To do this select the Raider mesh, and go back into EditMode. Select a vertex near the edge of the wing and then hit the **LKEY** to select linked vertices. Do the same on the other side. Next, click on the mid section of the ship and do the same thing. Select the areas shown in Figure 18-24. When you have those, hit the **PKEY** to separate the vertices selected.



Figure 18-24. Separating the Raider parts to be textured.

We now have our wing section separate and are ready to add the materials and textures. We want to create a new material for this mesh. To get a nice metallic look, we can use the settings in Figure 18-25.

▼ Material	▼ Shaders
⇒ MA:Material 3 🗙 🚭 F 🔺 🐨	Lambert + Ref 0.450 Halo
ME:Mesh.001 OB ME 4 16 Mat 1	Traceabl
VCol Light VCol Paint TexFace Shadeless	CookTorr + Spec 1.198 Radio
Col R 1.000	Hard:20 Wire
Spe G 1.000	Env
Mir Alpha 1 000	Amb 0.5 Emit 0.0 Mist
RGB HSV DYN SpecTra 0.00	Add 0.0 Zoffs: 0.000 Zinvert

Figure 18-25. "Metallic" material.

Time to add the textures. We want to achieve some pretty elaborate results. We will need two bump-maps to create grooves and two mask for painting and 'decals'. There are hence four textures for the Raider wings to be created, as shown in Figure 18-26.



Figure 18-26. Four textures, from upper left corner, clockwise: RaiderBM, RaiderDI, Markings, Raider.

The textures should be placed in four material channels in the raider top mesh. 'RaiderBM' and 'RaiderDI' should be set to a negative Nor (Figure 18-27 bottom - click Nor twice, it will turn yellow). 'Raider' should be set up as negative Ref (Figure 18-27 middle).

Which material?: A Mesh coming from a Radiosity solution typically has more than one material on it. It is important to operate on the right "original" material.

Col	Nor	Cs	0	Cmir	Ref
Spec	Har	rd	A	lpha	Emit
Col	Nor	Cs	0	Cmir	Ref
Spec	Har	rd	A	lpha	Emit
Col	Nor	Csp	0	Cmir	Ref
Spec	Hai	rd 📗	A	lpha	Emit

#### Figure 18-27. Texture set-ups.

The result is the desired metallic plating for the hull of the Raider. Finally the fourth texture, 'Markings', is set to Col in the MaterialButtons (Figure 18-27 top). This will give the Raider its proper striping and insignia. Our raider is quite flat, so the Flat projection is adequate. Were it a more complex shape some UV mapping would have been required to attain good results. The material preview for the mesh should look like Figure 18-28.



Figure 18-28. Complete material preview.

Our textures to *won't* show up in the rendering right now (except markings) because Nor and Ref type texture reacts to lighting, and there is no light source in the scene! Hence will now need to add a lamp or two, keeping in mind that our ship is still lit pretty well from the Radiosity solve, so lamps energy should be quite weak. Once you have your lamps, you try a test render. Experiment with the lamps until you get the results you like.

The final rendering (Figure 18-11) shows a nice well lit Raider with soft texturing.

# Chapter 19. Raytracing

TO BE WRITTEN

Chapter 19. Raytracing

# **Chapter 20. Particles**

# Introduction

Relevant to Blender v2.31 TO BE UPDATED

There are three kind of effects which can be linked to an Object, ideally working during animations but in practice precious even for Stills.

Effects are added to an Object by selecting it, switching to the Object Context and locating the Effects Tab in the Constraints Panel (F7 or 2). By pressing the New Effect button of Figure 20-1 an Effect is added.

		Combaints	Elico
Track (0.2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y)           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack KU (2 - K - Y) - 2         Upt (Y) - 2           Drack (2 - K - Y)	Dashpel Doan 200 Doan		NEW ENKL DOWN

#### Figure 20-1. Animation Buttons Window

The Delete button removes an effect, if one is there, while the drop down list which appears on the right once an effect is added (Figure 21-1) selects the type of effect.

More than one effect can be linked to a single Object. A row of small buttons, one for each effect, is created beneath the New Effect button, allowing you to switch from one to another to change settings.

The three effects are Build, Particles and Wave, the second being the most versatile and deserving a whole chapter for its own.

# **Simple Particles**

#### Relevant to Blender v2.31

The particle system of Blender is fast, flexible, and powerful. Every Mesh-object can serve as an emitter for particles. Halos can be used as particles and with the DupliVert option, so can objects. These dupliverted objects can be any type of Blender object, for example Mesh-objects, Curves, Metaballs, and even Lamps. Particles can be influenced by a global force to simulate physical effects, like gravity or wind.

With these possibilities you can generate smoke, fire, explosions, Fireworks or even flocks of birds. With static particles you can generate fur, grass, and even plants.

## A first Particle System

Reset Blender to the default scene, or make a scene with a single plane added from the top view. This plane will be our particle emitter. Rotate the view so that you get a good view of the plane and the space above it (Figure 20-2).



## Figure 20-2. The emitter.

Switch to the Effects Tab in the Object Context (F7 or ) and click the button NEW Effect in the middle part of the Panel. Change the dropdown MenuButton from Build to Particles. The Particle Buttons are now shown (Figure 20-3).

Constraints		Effec	ects		
NEW Effect	Delete	RecalcAll	Static Particles ¢		
≪ Tot: 1000► 🔍 Sta	a: 1.00⊧ [•	End: 100.00			
< Cur Mul: O> 🔍 M	lat: 1 🔺 🕒	(Mult: 0.000)	Life: 50.00 Child: 4		
andlife: 0.000 Seed: 0 Face Bspline Vect VectSize 0.000					
Norm: 0.000	0.000	Rand: 0.000	Tex: 0.000 Damp: 0.000		
X: 0.000 V: 0.0 Force: Z: 0.0	00 (*) 00 Te	K: 0.000 (V: exture: (Z:	0.000 Int RGB Grad 1.000 Int Nabla: 0.050		

Figure 20-3. The Particle Buttons.

Set the Norm: NumButton to 0.100 with a click on the right part of the button or use **SHIFT-LMB** to enter the value from the keyboard.

Play the animation by pressing **ALT-A** with the mouse over the 3DWindow. You will see a stream of particles ascending vertically from the four vertices.

Congratulations - you have just generated your first particle-system in a few easy steps!

To make the system a little bit more interesting, it is necessary to get deeper insight on the system and its buttons (Figure 20-4):

• The parameter Tot: controls the overall count of particles. On modern speedy CPUs you can increase the particle count without noticing a major slowdown.

- The total number of particles specified in the Tot: button are uniformly created along a time interval. Such a time interval is defined by the Sta: and End: Num-Buttons, which control the time interval (in frames) in which particles are generated.
- Particles have a lifetime, they last a given number of frames, from the one they are produced in onwards, then disappear. You can change the lifetime of the particles with the Life: NumButton.
- The Norm: NumButton used before made the particles having a starting speed of constant value (0.1) directed along the vertex normals. To make things more "random" you can set the Rand: NumButton to 0.1 too. This also makes the particles start with random variation to the speed.
- Use the Force: group of NumButtons to simulate a constant force, like wind or gravity. A Force: z: value of -0.1 will make the particles fall to the ground, for example.

Constraints	Effe	ects
NEW Effect	Delete RecalcAll	Static Particles C
≪ Tot: 3000⊁  ≪ Sta	a: 0.00) (4End: 100.00)	🕨 💶 Life: 60.00 🖉 Keys: 8 🕨
< Cur Mul: O> 🔍 🐇	1at: 1 🔺 🔳 Mult: 0.000>	Life: 50.00 Child: 4
andlife: 0.000	Seed: 0 🕨 🛛 Face 🛛 Bsplin	ne Vect VectSize 0.000
Norm: 0.100	o: 0.000 Rand: 0.100	Tex: 0.000 Damp: 0.000
X: 0.000 V: 0.0 Force: Z: -0.1	00 X: 0.000 V: 00 Texture: Z:	: 0.000 Int RGB Grad

Figure 20-4. Particles settings.

This should be enough to get you started, but don't be afraid to touch some of the other parameters while you're experimenting. We will cover them in detail in the following sections.

# Rendering a particle system

Maybe you've tried to render a picture from our example above. If the camera was aligned correctly, you will have seen a black picture with greyish blobby spots on it. This is the standard Halo-material that Blender assigns a newly generated particle system.

Position the camera so that you get a good view of the particle system. If you want to add a simple environment, remember to add some lights. The Halos are rendered without light, unless otherwise stated, but other objects need lights to be visible.

Go to the Material Buttons (F5) and add a new material for the emitter if none have been added so far. Click the Button "Halo" from the middle palette (Figure 20-5).



Figure 20-5. Halo settings

The Material Buttons change to the Halo Buttons. Choose Line, and adjust Lines: to a value of your choosing (you can see the effect directly in the Material-Preview). Decrease HaloSize: to 0.30, and choose a color for the Halo and for the lines (Figure 20-5).

You can now render a picture with **F12**, or a complete animation and see thousands of stars flying around (Figure 20-6).



**Figure 20-6. Shooting stars** 

## **Objects as particles**

It is very easy to use real objects as particles, it is exactly like the technique described in the Section called *DupliVerts* in Chapter 22.

Start by creating a cube, or any other object you like, in your scene. It's worth thinking about how powerful your computer is, as we are going to have as many objects, as Tot: indicates, in the scene. This means having as many vertices as the number of vertices of the chosen object times the value of Tot:!

Scale the newly created object down so that it matches the general scene scale.

Now select the object, then **SHIFT-RMB** the emitter and make it the parent of the cube using **CTRL-P**. Select the emitter alone and check the option "DupliVerts" in the Anim Settings Panel of the Object Context (F7). The dupliverted cubes will appear immediately in the 3DWindow.

	Constraints Effects
TrackX Y Z -X -Y -Z UpX Y Z	NEW Effect Delete RecalcA Static Particles +
Draw Key Draw Key Se Powertrack SlowPar	
DupliFrames DupliVerts Rot No Speed	Int: 400         Sta: 0.00         End: 100.00         End: 200.00         Keys: 8           CurMul: 0         Mat: 1         Mult: 0.000         Life: 50.00         Child: 4
DupSta: 1 DupOn: 1	andlife: 0.000  Seed: 0 Face Bsplin Vect VectSize 0.000
DupEnd 100	Norm: 0.100 Ob: 0.000 Rand: 0.052 Tex: 0.000 Damp: 0.100
Offs Ob Offs Par Offs Particle 0.0000 TimeOffset: 0.00 Automatic Time PrSpeed	X: 0.000         «Y: 0.000         «Y: 0.000         «Y: 0.000         RGB         Grad           Force:         -2: -0.100         Texture:         «Z: 1.000         Int         Nabla: 0.050

Figure 20-7. Setting Dupliverted Particles.

You might want to bring down the particle number before pressing **ALT-A** (Figure 20-7).

In the animation you will notice that all cubes share the same orientation. This can be interesting, but it can also be interesting to have the cubes randomly oriented.

This can be done by checking the option Vect in the particle-parameters, which causes the dupli-objects to follow the rotation of the particles, resulting in a more natural motion (Figure 20-7). One frame of the animation is shown in (Figure 20-8).

**Original Object:** Take care to move the original object out of the camera view, because, differently than in regular Mesh Dupliverts, in Dupliverted particles it will also be rendered!



Figure 20-8. Dupliverted particles rendering.

# Making fire with particles

The Blender particle system is very useful for making realistic fire and smoke. This could be a candle, a campfire, or a burning house. It's useful to consider how the fire is driven by physics. The flames of a fire are hot gases. They will rise because of their lower density when compared to the surrounding cooler air. Flames are hot and bright in the middle, and they fade and become darker towards their perimeter.

Prepare a simple set-up for our fire, with some pieces of wood, and some rocks (Figure 20-9).



Figure 20-9. Campfire setup.

## The particle system

Add a plane into the middle of the stone-circle. This plane will be our particle-emitter. Subdivide the plane once. You now can move the vertices to a position on the wood where the flames (particles) should originate.

Now go to the Object Context F7 and add a new particle effect to the plane. The numbers given here (Figure 20-10) should make for a realistic fire, but some modification may be necessary, depending on the actual emitter's size.

Constraints Effects
NEW Effect Delete RecalcAl Static Particles +
Tot: 1003 ta: -50.00 End: 100.00 Life: 30.00 Keys: 8
CurMul: 0 Mat: 1 Mult: 0.000 Life: 50.00 Child: 4
ndlife: 0.300 Seed: 0 Face Bsplin Vect VectSize 0.000
orm: -0.008 Ob: 0.000 and: 0.014 Tex: 0.000 amp: 0.100
X: 0.000 Y: 0.000 X: 0.000 Y: 0.000 RGB Grad
Force: Z: 0.202 Texture: Z: 1.000 abla: 0.050

Figure 20-10. Fire particles setup.

Some notes:

- To have the fire burning from the start of the animation make Sta: negative. For example, try -50. The value of End: should reflect the desired animation length.
- The Life: of the particles is 30. Actually it can stay at 50 for now. We will use this parameter later to adjust the height of the flames.
- Make the Norm: parameter a bit negative (-0.008) as this will result in a fire that has a bigger volume at its basis.
- Use a Force: Z: of about 0.200. If your fire looks too slow, this is the parameter to adjust.
- Change Damp: to 0.100 to slow down the flames after a while.
- Activate the Bspline Button. This will use an interpolation method which gives a much more fluid movement.
- To add some randomness to our particles, adjust the Rand: parameter to about 0.014. Use the Randlife: parameter to add randomness in the lifetime of the particles; a really high value here gives a lively flame.
- Use about 600-1000 particles in total for the animation (Tot:).

In the 3DWindow, you will now get a first impression of how realistically the flames move. But the most important thing for our fire will be the material.

### The fire-material

With the particle emitter selected, go to the Shading Context F5 and add a new Material. Make the new material a halo-material by activating the Halo button. Also, activate HaloTex, located below this button. This allows us to use a texture later.



Figure 20-11. Flames Material.

Give the material a fully saturated red colour with the RGB-sliders. Decrease the Alpha value to 0.700; this will make the flames a little bit transparent. Increase the Add slider up to 0.700, so the Halos will boost each other, giving us a bright interior to the flames, and a darker exterior. (Figure 20-11).

	Texture Cold	rs	
Mat	🗢 TE:Clouds 🛛 🗶 🕏	F	Default Color Soft noise Hard noise
World	Clouds Not	ie	
Lamp		e EnvMan	NoiseSize : 0.600      NoiseDepth: 2     NoiseDepth: 3     NoiseDepth: 3
2.2.2			
March 2014	Clou	ds Marble	
CONTRACTOR OF CONTRACTOR	Mag	ic Blend	
Default Var	Noi	e Plugin	

Figure 20-12. Flames Texture.

If you now do a test render, you will only see a bright red flame. To add a touch more realism, we need a texture. While the emitter is still selected, go to the Texture Panel and add a new Texture select the Cloud-type for it in the Texture (F6) Buttons. Adjust the NoiseSize: to 0.600. (Figure 20-12).

Go back to the Material Buttons F5 and make the texture colour a yellow colour with the RGB sliders on the right side of the material buttons. To stretch the yellow spots from the cloud texture decrease the SizeY value down to 0.30.

A test rendering will now display a nice fire. But we still need to make the particles fade out at the top of the fire. We can achieve this with a material animation of the Alpha and the Halo Size.

#### Chapter 20. Particles

Be sure that your animation is at frame 1 (SHIFT-LEFTARROW) and move the mouse over the Material Window. Now press IKEY and choose Alpha from the appearing menu. Advance the frame-slider to frame 100, set the Alpha to 0.0 and insert another key for the Alpha with IKEY. Switch one Window to an IPO Window. Activate the Material IPO Type by clicking the pertinent Menu Entry in the IPO Window header. You will see one curve for the Alpha-channel of the Material (Figure 20-13).

**Note:** An animation for a particle material is always mapped from the first 100 frames of the animation to the lifetime of a particle. This means that when we fade out a material in frame 1 to 100, a particle with a lifetime of 50 will fade out in that time.



Figure 20-13. Fire Material IPO

Now you can render an animation. Maybe you will have to fine-tune some parameters like the life-time of the particles. You can add a great deal of realism to the scene by animating the lights (or use shadow-spotlights) and adding a sparks particlesystem to the fire. Also recommended is to animate the emitter in order to get more lively flames, or use more than one emitter (Figure 20-14).





## A simple explosion

This explosion is designed to be used as an animated texture, for composing it with the actual scene or for using it as animated texture. For a still rendering, or a slow motion of an explosion, we may need to do a little more work in order to make it look really good. But bear in mind, that our explosion will only be seen for half a second (Figure 20-15).



**Figure 20-15. The explosion** 

As emitter for the explosion I have chosen an IcoSphere. To make the explosion slightly irregular, I deleted patterns of vertices with the circle select function in Edit Mode. For a specific scene it might be better to use an object as the emitter, which is shaped differently, for example like the actual object you want to blow up.

My explosion is composed from two particle systems, one for the cloud of hot gases and one for the sparks. I took a rotated version of the emitter for generating the sparks. Additionally, I animated the rotation of the emitters while the particles were being generated.

### The materials

The particles for the explosion are very straightforward halo materials, with a cloud texture applied to add randomness, the sparks too have a very similar material, see Figure 20-16 to Figure 20-18.

🔻 Preview					Tosture Map Input	
		B MA:Cloud     X ⊕ F     ★ ♥     ME:Sphere.010     OB     ME < 1 Mat 1 →	Halo Stre: 0.40	Hate Flare	UV Object Glob Dree Stick Win Nor Refl	Ox Nor Cap Cmir Ref Spec Hard Astro Emit
		R 0.800	Add 1.000	Rings Lines Star	152         Cube         c ofsX 0.000         c           Tube         Sphe         c ofsY 0.000         c           c ofsZ 0.000         c         c         c	Stend Neg No RG Mil Add Sub
	ৰ	Unit         B 0.000           Ring         Apha 0.832           Life         HS         DVN           Spectra 0.00         Spectra 0.00	star: 4 ⇒   < Seed: 0 ⇒	tstoTex tstoFun ( Alpha Shaded	X         Y         Z         isizeX 1.00         i           X         Y         Z         isizeY 1.00         i           X         Y         Z         isizeY 1.00         i	G 1.000 Col 1.000 Nor 0.500 Ver 1.00

Figure 20-16. Material for the explosion cloud.



Figure 20-17. Material for the sparks.



Figure 20-18. Texture for both.

Animate the Alpha-value of the Halo particles from 1.0 to 0.0 at the first 100 frames. This will be mapped to the life-time of the particles, as is usual. Notice the setting of Star in the sparks material (Figure 20-17). This shapes the sparks a little bit. We could have also used a special texture to achieve this, however, in this case using the Star setting is the easiest option.

Constraints	Effects
NEW Effect	Delete RecalcAl Static Particles +
✓ Tot: 401 ►	a: 1.00 Keys: 8
CurMul: 0	1at: 1 → Mult: 0.000 Life: 50.00 Child: 4
ndlife: 0.444	eed: 0 Face Bsplin Vect VectSize 0.000
Norm: 0.100	b: 0.000 and: 0.032 Tex: 0.000 amp: 0.062
X: 0.000 Y: 0.0	00 (RGB [Grad]
Force: Z: 0.0	00 Texture: Z: 1.000 abla: 0.050

### The particle-systems

Figure 20-19. Particle system for the cloud

Constraints	Effe	cts	
NEW Effect	Delete RecalcAl	Static Particles	\$
			_
◄ Tot: 403 ►	a: 1.00► 🖪 End: 4.00 ►	Life: 10.00 Keys:	8⊩
CurMul: 0	1at: 1 🕨 🛯 📶 4Mult: 0.000	Life: 50.00 Child:	4⊩
ndlife: 2.000	Seed: 0 Face Bspli	n Vect VectSize 0.0	00
Norm: 0.180 0	b: 0.000 and: 0.032	Tex: 0.000 amp: 0.0	62
X: 0.000 (Y: 0.0 Force: Z: 0.0	000 X: 0.000 Y: 00 Texture: Z:	0.000 Int RGB G 1.000 Int abla: 0.0	rad ISO

#### Figure 20-20. Particle system for the sparks

As you can see in Figure 20-19 and Figure 20-20, the parameters are basically the same. The difference is the Vect setting for the sparks, and the higher setting of Norm: which causes a higher speed for the sparks. I also set the Randlife: for the sparks to 2.000 resulting in an irregular shape.

I suggest that you start experimenting, using these parameters to begin with. The actual settings are dependent on what you want to achieve. Try adding more emitters for debris, smoke, etc.

### **Fireworks**

A button we have not used so far is the Mult: button. The whole third line of buttons in the Panel is related to this. Prepare a plane and add a particle system to the plane.

Adjust the parameters so that you get some particles flying into the sky, then increase the value of Mult: to 1.0. This will cause 100% of the particles to generate child particles when their life ends. Right now, every particle will generate four children. So we'll need to increase the Child: value to about 90 (Figure 20-21). You should now see a convincing firework made from particles, when you preview the animation with **ALT-A**.

Constraints	Effe	ects	
NEW Effect	Delete RecalcAl	Static	Particles 🗢
	·		
◄ Tot: 800 ►	a: 1.00 End: 100.00	) <ul> <li>Life: 50</li> </ul>	.00 🔍 Keys: 8>
≪CurMul: 0⊧ 🔍 🗸	1at: 1 🕨 📶 Mult: 1.000	Life: 50	.00 Child: 90
ndlife: 0.200	Seed: 0   Face  Bspl	in Vect	VectSize 0.000
Norm: 0.300 OI	b: 0.000 and: 0.100	Tex: 0.00	00 amp: 0.062
X: 0.000 Y: 0.0 Force: Z: -0.0	000 X: 0.000 Y 096 Texture: Z	: 0.000 : 1.000	nt RGB Grad abla: 0.050

Figure 20-21. Particle Multiplication buttons

When you render the firework it will not look very impressive. This is because of the standard halo material that Blender assigns. Consequently, the next step is to assign a better material.

Ensure that you have the emitter selected and go to the Shading Context and Material Buttons (F5). Add a new material with the Menu Button, and set the type to Halo.

Preview	▼ Material	Shaders
	AA:Material 1     X	HaloSize: 0.50 Flare Hard 50 Rings Add 0.000 Rings Star: 4 > Seed: 7 > HaloTex HaloPun X Alpha Shaded

Figure 20-22. Firework Material 1

I have used a pretty straightforward halo material; you can see the parameters in Figure 20-22. The rendered animation will now look much better, yet there is still something we can do.

While the emitter is selected go to the Editing Context F9 and add a new material index by clicking on the New button in the Link and Materials Panel (Figure 20-23).

Material 1					
4 2 Mat: 1 ▶ ?					
New Delete					
Select Deselect					
Assign					
Set Smooth Set Solid					

#### Figure 20-23. Adding a second material to the emitter.

Now switch back to the Shading Context. You will see that the material data browsehas changed colour to blue. The button labelled 2 indicates that this material is used by two users. Now click on the 2 button and confirm the popup. Rename the Material to "Material 2" and change the colour of the halo and the lines (Figure 20-24).

▼ Preview	<ul> <li>Material</li> </ul>	Shaders
	MA:Material 2     X G F     ME:Plane     OB ME     2 Mat 2     ME:Plane     OB ME     2 Mat 2     ME:Plane     Ring     Ring	# HaloSize: 0.50         Hard           Hard 50         Hard           Add 0.000         Hard           Rings: 4 > 4         Lines: 6 > Star           Star: 4 > 5         Seed: 7 >           HaloPunn         X Alpha           Shaded         Shaded

Figure 20-24. Material 2

Switch to the particle parameters and change the Mat: button to "2". Render again and you see that the first generation of particles is now using the first material and the second generation the second material! This way you can have up to 16 (that's the maximum number of material indices) materials for particles.

**Further enhancements:** Beside changing materials you also can use the material IPOs to animate material settings of each different material.

## **Controlling Particles via a Lattice**

Blender's particle system is extremely powerful, and the course of particles can not only be determined via forces but channelled by a lattice.

Prepare a single square mesh and add a particle system to it with a negative z-force and the general parameters in Figure 20-25.



Figure 20-25. Particle settings

This could be good for the smoke of four small fires fire in a windless day, but we want to twist it! Add a lattice and deform it as in as in Figure 20-26.



Figure 20-26. Lattice settings

Parent the particle emitter to the lattice (**CTRL-P**). If you now select the particle emitter, switch to Animation buttons (**F7**) and press RecalcAll you will notice that the particles follows, more or less, the lattice (Figure 20-27 on the left).

As a further tweak, rotate each horizontal section of the lattice 60 degrees clockwise in top view, incrementally, as if you were making a screw. After this, recalculate again the particles. The result is in Figure 20-27 on the right.



Figure 20-27. Lattice deformation effects

The twist is evident, and of course you can achieve even stronger effects by rotating the lattice more or by using a lattice with more subdivisions. If you give the emitter a halo material and you render you will see something like Figure 20-28 on the left.



Figure 20-28. Normal particles, left; Vector particles, centre; and DupliVerted objects following the particles, right.

If you select the emitter, turn to animation buttons and press the Vect Particle Button the particles will turn from points to segments, with a length and a direction proportional to the particle velocity. A rendering now will give the result of Figure 20-28 in the middle.

It you now Duplivert an object to the emitter, by parenting it and by pressing the Duplivert button, the DupliVerted objects will have the same orientation of the original object if the particles are normal particles, but will be rotated and aligned to the particle direction if the Particles are set to vert. By selecting the Original Object and by playing with the Track buttons you can change orientation (Figure 20-28 on the right).

## **Static Particles**

Static particles are useful when making objects like fibres, grass, fur and plants.

Try making a little character, or just a ball, to test the static particles. Try for example a small 'ball of fur' guy. An emitter is not rendered, so duplicate the mesh (or whatever object type you used and convert (**ALT-C**) it into a mesh). A fractal subdivide to the mesh to get some randomness into it, is usually a good idea. If you end up with mesh that is too dense, use "Remove Doubles" with an increased limit. Cut out parts with the circle select where you do not want to have fur.

Now, assign the particle system and, switch on the Static option.

<ul> <li>Constraints</li> </ul>	Effe	cts	
NEW Effect Del	ete RecalcAl	Static Pa	rticles 🗢
Tot: 10000	Step: 1 🔹 🕨	Interaction ■ Life: 1.00	≪Keys: 8⊧
≪CurMul: 0 ≤ Mat: 1	Mult: 0.000	Life: 50.00	<child: 4►<="" td=""></child:>
ndlife: 1.056 Seed:	0 Face Bsplin	n Vect Vec	tSize 1.000
Norm: 0.090 Ob: 0.0	00 and: 0.212	Tex: 0.000	amp: 0.024
X: 0.000 Y: 0.000 Force: Z: -0.296	X: 0.000 Y: Texture: Z:	0.000 Int	RGB Grad abla: 0.050

Figure 20-29. Static particle settings

Use these parameters in Figure 20-29. With the combination of Life and Norm you can control the length of the hair. Use a force in a negative z-direction to let the hair bend. Check Face to generate the particles, not only on the vertices but also distributed on the faces. Also check Vect; this will generate fibre like particles. The Step value defines how many particles per lifetime are generated. Set this to a lower value to get smoother curves for the particles, and be sure not to overlook setting the Rand value.

When you now render, you will get very blurred particles. The material used for static particles is very important, so add a material for the emitter in the Shading Context (F5).



Figure 20-30. Material settings

I use a very small HaloSize (0.001). In the Number Button you can't see that, so to adjust click the button with the **LMB** while holding **SHIFT**. Enable the Shaded option to have the particles influenced by the lights in the scene, and then activate HaloTex. We are going to use a texture to shape the hairs (Figure 20-30).



Figure 20-31. Texture Colorband settings

Switch to the Texture sub-context (**F6**) and add a new Blend type texture. Choose Lin as sub-type. Activate the Colorband option and adjust the colors as in Figure 20-31. You will get a nice blend, from transparent through to purple and back again to transparent.

▼ Map Input	
UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
Flat         Cube         ofsX 0.000           Tube         Sphe         ofsY 0.000           ofsZ 0.000	Stenc) Neg No RGB Mix Mul Add Sub
X         Y         Z         sizeX         7.00           X         Y         Z         sizeY         3.10           X         Y         Z         sizeZ         1.00	G 1.000 Col 0.553 Col 0.55

Figure 20-32. Texture settings in the Material buttons

Go back to the Material Buttons sub-context and make sure that Alpha is activated in the texture mapping output on the right of the Material Buttons. Then use sizeX and sizeY to shape the halo in the material preview to a small fiber (Figure 20-32).

If your fur is not dense enough, then increase the particle count with Tot or add more emitters. Also, change the particle parameters for these additional emitters a little so that you get some variation in the hairs (Figure 20-33).



Figure 20-33. Final result

## **Particle Interaction**

*by Kenneth Styrberg* Relevant to Blender v2.34

## Introduction

The Blender particle system allow particles to interact in two ways - force fields and deflection.

#### **Force fields**

There is two variants of force fields: Standard Force field that behaves like a gravity force field and the Vortex field. A vortex field gives more like a tornado effect, with spiralling particles around the vortex center. Force fields are available for all object types. Currently a point-based field with a spherical fall-off is implemented.

Force fields can be set to any object. Particles will then be: attracted with a negative Strength value, or deflected if you use a positive Strength value. There is also a Fall-off parameter which define how much the strength diminishes by the distance from object origin.

#### Deflection

This allows you to set any **mesh** object as a particle deflector, particles will then bounce on the surface of the mesh. You can control how much the particles bounce with the Damping value, add some randomness to the bounce with Rnd Damping and you can define the percentage of particles which pass through the mesh with the Permeability parameter.

**Note:** Make sure that the normals of the mesh surface are facing towards the particles for correct deflection.

### Interface

The particle interaction settings are controlled via the Particle Interaction panel in object context (F7).



Figure 20-34. Particle Interaction panel.

Force/Vortex field

- Strength The strength of the field effect
- Fall-off How much the strength diminishes with distance

When adding a force/vortex field to an object, the object will get a small graphic indicating that there is a particle interaction connected to the object. The force field will have small grey circles Figure 20-35, and vortex fields will have a spiral drawn Figure 20-36.



Figure 20-35. Force field indicator.



Figure 20-36. Vortex field indicator.

The lower section of the panel, Figure 20-34, handle the settings for deflection.

**Note:** You will not see any extra graphic indicators with deflectors as you do with force fields.

#### **Deflection parameters**

• Damping Controls the amount of bounce that surfaces will have.

0.0 - No damping, particles will have maximum bounce.

- 1.0 Maximum damping, particles will not bounce at all.
- Rnd Damping Adds a random element to the bounce. The damping will vary starting from, not around the Damping value. Damping + Rnd Damping.
- Permeability Percentage of particles passing through the mesh. 0.0 - No particles pass through.
  - 1.0 All particles pass through the deflector.

If you set up a particle deflector you'll have to make sure sufficient keys are available for Blender to calculate the collisions with sufficent detail. If you see particles moving through your deflector or bounce at wrong positions, then there might be problem with too few keys or your particle/deflector is moving too fast.

**Note:** You can animate moving deflectors but particles can leak through the mesh if the deflector moves to fast or if mesh is complicated. This can be partly solved by increasing the  $\kappa_{eys}$  parameter for the particle emitter.

After changing any parameters, you will need to select your particle emitter and go back to the Effects tab and press **RecalcA** button. Figure 20-37

More keys means longer calculation times and usage of more memory. See the Section called *Introduction* for how to set up particle emitters.

🔻 Constra	ints	E	ffects		Pa	rticle	Interactio	n
NEW Effect	rt	Delete	Recal	cA	Static	Pa	rticles	\$
								_
Tot: 1000 ►	∢Sta	: 1.00▶	End: 10	0.00	<life: 5<="" td=""><td>0.00</td><td>≪Keys:</td><td>8 ⊧</td></life:>	0.00	≪Keys:	8 ⊧
≪CurMul: 0►	< M	at: 1 🕨	∢Mult: 0.	.000⊾	<life: 5<="" td=""><td>0.00</td><td></td><td>4 ⊧</td></life:>	0.00		4 ⊧
andlife: 0.000	) < S	eed:0⊳	Face E	Bspline	e Vect	√ec	tSize 0.0	00
Norm: 0.000	-Ob	: 0.000	Rand: 0	0.000	Tex: 0.0	000	)amp: 0.0	000
X: 0.000 Y Force: Z	: 0.0( : 0.0(	00) (	X: 0.000 'exture:	Y: ( Z: 1	0.000	Int	RGB Gr abla: 0.0	ad 50

Figure 20-37. Effects panel.

For all parameters for force fields and deflectors, Ipo keys can be inserted. The Ipo curves is edited as Object Ipo types in the Ipo window. See Chapter 14 for more on Animation and Ipo.

## Example

Here is a small example demonstrating particle deflectors.

- 1. Start with default scene with a cube object.
- 2. In top view, add a Mesh Circle and accept default 32 vertices. Press TAB to exit edit mode. Switch to a
- **3.** With circle selected, press **S**, and scale it down to 0.1 in all directions.
- 4. Now press F7 and go to the Effects tab. Press NEW Effect and in the Popup-list select Particles. Y
- 5. Increase particle life from default 50 to 100 by changing the Life field to 100. In the force field in the lo



Figure 20-38. Example frame 40.

6. Now select our cube. With the Effect context still active, go to the Particle Interaction tab and set



Figure 20-39. Example Deflection settings.

7. Now select the circle and press **RecalcA** in the Effects tab. If you play the animation, **ALT-A**, you will **8.** Select the cube and tilt it 10 degrees in either direction. Select the circle and recalculate the particles. No



Figure 20-40. Example end result.

# **Chapter 21. Other Effects**

## Introduction

Relevant to Blender v2.31

There are two other kind of effects, besides particles, which can be linked to an Object, ideally working during animations but in practice precious even for Stills.

These two effects are Build, and Wave, the following sections will describe each of them in detail.

## **Build Effect**

Relevant to Blender v2.31

The Build effect works on Meshes and causes the faces of the Object to appear, one after the other, over time. If the Material of the Mesh is a Halo Material, rather than a standard one, then the vertices of the Mesh, not the faces, appear one after another.

Constraints Effects	
NEW Effect Delete	Build \$
Len: 100.00 Sfra: 1.00	

#### **Figure 21-1. Build Effect**

Faces, or vertices, appear in the order in which they are stored in memory. This order can be altered by selecting the Object and pressing **CTRL-F** out of EditMode. This causes faces to be re-sorted as a function of their value (Z co-ordinate) in the local reference of the Mesh.

**Reordering:** If you create a plane and add the Build effect to see how it works you won't be happy. First, you must subdivide it so that it is made up of many faces, not just one. Then, pressing **CTRL-F** won't do much because the Z-axis is orthogonal to the plane. You must rotate it in EditMode to have some numerical difference between the co-ordinates of the faces, in order to be able to reorder them.

The Build effect only has two NumBut controls (Figure 21-1):

Len - Defines how many frames the build will take.

Sfra - Defines the start frame of the building process.

## Wave Effect

Relevant to Blender v2.31

#### Chapter 21. Other Effects

The Wave effect adds a motion to the Z co-ordinate of the Object Mesh.

<ul> <li>Constraints</li> </ul>	Effects
NEW Effect Delete	Wave +
≪ Sta x: 0.00 ⊁   ≪ Sta y: 0.00≻	X Y Cycl
Speed:0.500	
Heigth:0.500	Lifetime: 0.00
Width:1.500	▲ Damptime: 10.00 →
Name of E00	

#### Figure 21-2. Wave Control Panel

The wave effect influence is generated from a given starting point defined by the Sta X and Sta Y Num Buttons. These co-ordinates are in the Mesh local reference (Figure 21-3).

≪ Sta x: 0.00 ⊁ ≪ Sta y: 0.00⊁

#### Figure 21-3. Wave Origin

The Wave effect deformation originates from the given starting point and propagates along the Mesh with circular wave fronts, or with rectilinear wave fronts, parallel to the X or Y axis. This is controlled by the two x and x toggle buttons. If just one button is pressed fronts are linear, if both are pressed fronts are circular (Figure 21-4).

The wave itself is a gaussian-like ripple which can be either a single pulse or a series of ripples, if the Cycl button is pressed.

X Y Cycl

#### Figure 21-4. Wave front type

The Wave is governed by two series of controls, the first defining the Wave form, the second the effect duration.

For what concerns Wave Form, controls are Speed, Height, Width and Narrow (Figure 21-5).

Speed:0.500	
Heigth:0.500	
Width:1.500	
Narrow:1.500	

Figure 21-5. Wave front controls

The Speed Num Button controls the speed, in Units per Frame, of the ripple.

The Height Num Button controls the height, in Blender Units and along Z, of the ripple (Figure 21-6).

If the Cycl button is pressed, the Width Num Button states the distance, in Blender Units, between the topmost part of two subsequent ripples, and the total Wave effect is given by the envelope of all the single pulses (Figure 21-6).

This has an indirect effect on the ripple amplitude. Being ripples Gaussian in shape, if the pulses are too next to each other the envelope could not reach the z=0 quote any more. If this is the case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude value, as shown in Figure 21-6 at the bottom.

The actual width of each Gaussian-like pulse is controlled by the Narrow Num Button, the higher the value the narrower the pulse. The actual width of the area in which the single pulse is significantly non-zero in Blender Units is given by 4 over the Narrow Value. That is, if Narrow is 1 the pulse is 4 Units wide, and if Narrow is 4 the pulse is 1 Unit Wide.



#### Figure 21-6. Wave front characteristics

To obtain a Sinusoidal-like wave: To obtain a nice Wave effect similar to sea waves and close to a sinusoidal wave it is necessary that the distance between following ripples and the ripple width are equal, that is the Width Num Button value must be equal to 4 over the Narrow Value.

The last Wave controls are the time controls. The three NumButs define:

Time sta the Frame at which the Wave begins;

Lifetime the number of frames in which the effect lasts;

Damptime is an additional number of frames in which the wave slowly dampens from the Amplitude value to zero. The Dampening occurs for all the ripples and begins in the first frame after the Lifetime is over. Ripples disappear over Damptime frames.

4	Time sta: 0.00	$\rightarrow$
4	Lifetime: 0.00	Þ
4	Damptime: 10.00	Þ

Figure 21-7. Wave time controls

Chapter 21. Other Effects

# Chapter 22. Special modelling techniques

Claudio "malefico" Andaur

## Introduction

Relevant to Blender v2.31

Once we have overcome the "extrusion modelling fever" and started to look at more challenging modelling targets, we might start the quest for alternative methods to do the job. There are a group of modelling techniques in Blender which not only make our modelling job easier but sometimes make it *possible*.

These so called "special" modelling techniques involve not only some vertex manipulation but the use of non-intuitive procedures which require a deeper knowledge or experience from the user than the average beginner.

In this chapter we will describe these techniques in detail and explain their utility in several modelling applications which could not have been solved any other way.

## **DupliVerts**

#### Relevant to Blender v2.31

"DupliVerts" are not a rock band nor a dutch word for something illegal (well maybe it is) but is a contraction for "DUPLIcation at VERTiceS", meaning the duplication of a base Object at the location of the Vertices of a Mesh (or even a Particle system). In other words, when using DupliVerts on a mesh, an instance of the base object is placed on every vertex of the mesh.

There are actually two approaches to modelling using DupliVerts. They can be used as an arranging tool, allowing us to model geometrical arrangement of objects (eg: the columns of a Greek temple, the trees in a garden, an army of robot soldiers, the desks in a classroom). The object can be of any object type which Blender supports.

The second approach is to use them to model an Object starting from a single part of it (i.e.: the spikes in a club, the thorns of a sea-urchin, the tiles in a wall, the petals in a flower).

## **DupliVerts as an Arranging Tool**

All you need is a base object (eg: the "tree" or the "column") and a mesh with its vertices following the pattern you have in mind.

I will use a simple scene for the following part. It consists of a camera, the lamps, a plane (for the floor) and a strange man I modelled after Magritte's famous character (Figure 22-1). If you don't like surrealism you will find this part extremely boring.



Figure 22-1. A simple scene to play with.

Anyway, the man will be my "base Object". It is a good idea that he will be at the centre of the co-ordinate system, and with all rotations cleared. Move the cursor to the base object's centre, and From Top View add a mesh circle, with 12 vertices or so (Figure 22-2).



Figure 22-2. The parent mesh can be any primitive.

Out of Edit Mode, select the base Object and add the circle to the selection (order is very important here). Parent the base object to the circle by pressing **CTRL-P**. Now, the circle is the parent of the character (Figure 22-3). We are almost done.



Figure 22-3. The man is parented to the circle.

<ul> <li>Anim settings</li> </ul>
TrackX Y Z - X - Y - Z UpX Y Z
Draw Key Draw Key Se Powertrack SlowPar
DupliFrames DupliVerts Rot No Speed
DupSta: 1 DupOn: 1
Offs Ob Offs Par Offs Particle 0.0000
TimeOffset: 0.00 Automatic Time PrSpeed

Figure 22-4. The Animation Buttons

Now select only the circle, switch the Buttons Window to the Object Context (via or F7) and select the Dupliverts Button in the Anim Settings Panel (Figure 22-4).



Figure 22-5. In every vertex of the circle a man is placed.

Wow, isn't it great? Don't worry about the object at the centre (Figure 22-5). It is still shown in the 3D-views, but it will *not* be rendered. You can now select the base object, change (scale, rotate, Edit Mode)<sup>1</sup> it and all DupliVerted objects will reflect the changes. But the more interesting thing to note is that you can also edit the parent circle.

**Note:** The base Object is not rendered if DupliVerted on a Mesh but it *is* rendered if DupliVerted on a Particle System!

Select the circle and scale it. You can see that the mysterious men are uniformly scaled with it. Now enter the Edit Mode for the circle, select all vertices **AKEY** and scale it up about three times. Leave Edit Mode and the DupliVerted objects will update (Figure 22-6). This time they will still have their original size but the distance between them will have changed. Not only we can scale in Edit Mode, but we can also delete or add vertices to change the arrangement of men.



Figure 22-6. Changing the size of the circle in Edit Mode.

Select all vertices in Edit Mode and duplicate them (SHIFT-D). Now scale the new vertices outwards to get a second circle around the original. Leave Edit Mode, and a second circle of men will appear (Figure 22-7).



Figure 22-7. A second row of Magritte's men.

Until now all Magritte's men were facing the camera, ignoring each other. We can get more interesting results using the Rot Button next to the DupliVerts button in the Anim Settings Panel. With this Tog Button active, we can rotate the DupliVerted

objects according to the normals of the parent Object. More precisely, the DupliVerted Objects axis are aligned with the normal at the vertex location.

Which axis is aligned (X, Y or Z) with the parent mesh normal depends on what is indicated in the TrackX, Y, Z buttons and the UpX, Y, Z buttons top in the Anim Settings Panel. Trying this with our surrealist buddies, will lead to weird results depending on these settings.

The best way to figure out what will happen is first of all aligning the "base" and "parent" objects' axis with the World axis. This is done selecting both objects and pressing **CTRL-A**, and click the Apply Size/Rot? menu.



Figure 22-8. Show object's axis to get what you want.

Then make the axis of the base object and the axis and normals in the parent object visible (Figure 22-8 - in this case, being a circle with no faces, a face must be defined first for the normal to be visible - actually to exist at all).

Now select the base object (our Magritte's man) and play a little with the Tracking buttons. Note the different alignment of the axis with the different combinations of Upx, Y, Z and TrackX, Y, Z (Figure 22-9, Figure 22-10, Figure 22-11, Figure 22-12).



Figure 22-9. Negative Y Axis is aligned to vertex normal (pointing to the circle's

center)



Figure 22-10. Positive Y axis is aligned to normal



Figure 22-11. Positive X axis is aligned to normal



Figure 22-12. Positive Z axis is aligned to normal (weird, huh?)

## **DupliVerts to Model a Single Object**

Very interesting models can be made using DupliVerts and a standard primitive.

Starting from a cube in Front View, and extruding a couple of times I have modelled something which looks like a tentacle when SubSurfs are activated (Figure 22-13). Then I added an Icosphere with 2 subdivisions.



Figure 22-13. Strange tentacle and SubSurfed version.

I had special care to be sure that the tentacle was located at the sphere centre, and that both the tentacle axis and the sphere axis were aligned with the world axis as above (Figure 22-14).



Figure 22-14. Local reference of the tentacle.

Now, I simply make the icosphere the parent of the tentacle. Select the icosphere alone and made it DupliVert in the Anim Settings Panel (Figure 22-15).

Press the Rot button to rotate the tentacles (Figure 22-16).



Figure 22-15. DupliVerts not rotated.



Figure 22-16. DupliVerts rotated.

Once again to make the tentacle point outwards we have to take a closer look to its axis. When applying Rot, Blender will try to align one of the tentacle axis with the normal vector at the parent mesh vertex.

We didn't care about the Parent circle for Magritte's men, but here we should care about the Sphere, and you will soon notice that it is *not* rendered. You probably would like to add an extra renderable sphere to complete the model.

You can experiment in Edit Mode with the tentacle, moving its vertices off the centre of the sphere, but the object's centre should always be at the sphere's centre in order to get a symmetrical figure. However take care not to scale up or down in one axis in Object Mode since it would lead to unpredictable results in the DupliVerted objects when applying the Rot button.



Figure 22-17. Our model complete.

Once you're done with the model and you are happy with the results, you can select the tentacle and press **SHIFT-CTRL-A** and click on the Make duplis real ? menu to turn your virtual copies into real meshes (Figure 22-17).

## **DupliFrames**

#### Relevant to Blender v2.31

You can consider DupliFrames in two different ways: an arranging or a modelling tool. In a way, DupliFrames are quite similar to DupliVerts. The only difference is that with DupliFrames we arrange our objects by making them follow a curve rather than using the vertex of a mesh.

DupliFrames stands for DUPLIcation at FRAMES and is a very useful modelling technique for objects which are repeated along a path, such as the wooden sleepers in a railroad, the boards in a fence or the links in a chain, but also for modelling complex curve objects like corkscrews, seashells and spirals.

### Modelling using DupliFrames

We are going to model a chain with its links using DupliFrames.

First things come first. To explain the use of DupliFrames as a modelling technique, we will start by modelling a single link. To do this, add in front view a Curve Circle (Bézier or NURBS, whatever). In Edit Mode, subdivide it once and move the vertices a little to fit the link's outline (Figure 22-18).



Figure 22-18. Link's outline

Leave Edit Mode and add a Surface Circle object (Figure 22-19). NURBS-surfaces are ideal for this purpose, because we can change the resolution easily after creation, and if we need to, we can convert them to a mesh object. It is very important that you do not confuse Curve Circle and Surface Circle. The first one will act as the shape of the link but it will not let us do the skinning step later on. The second one will act as a cross section of our skinning.



Figure 22-19. Link's cross section

Now parent the circle surface to the circle curve (the link's outline) as a Normal parent (not a Curve Follow constraint). Select the curve and in the Object Context and Anim Settings Panel press CurvePath and CurveFollow (Figure 22-20).

▼ Curve and Surface	
UV Orco	✓ DefResolU: 6 ▶ Set
Centre	Back Front 3D
Centre New	Width: 1.000
Centre Cursor	✓ Ext1: 0.000 →
I ← PathLen: 35 →	
CurvePath CurveFoll	🔹 BevResol: 0 🕞
PrintLen 0.0000	BevOb:

Figure 22-20. Curve's settings: Curve Path and Curve Follow.

It's probable that the circle surface will appear dislocated. Just select it and press **ALT-O** to clear the origin (Figure 22-21).



Figure 22-21. Clearing origin.

If you hit **ALT-A** the circle will follow the curve. Now you probably will have to adjust the TrackX, Y, Z and UpX, Y, Z animation buttons, to make the circle go perpendicular to the curve path (Figure 22-22).



Figure 22-22. Tracking the right axis.

Now select the Surface Circle and go to Anim Settings Panel and press DupliFrames. A number of instances of the circular cross section will appear along the curve path (Figure 22-23).



Figure 22-23. DupliFrames!

You can adjust the number of circles you want to have with the DupSta, DupEnd, DupOn and DupOff buttons. These buttons control the Start and End of the duplication, the number of duplicates each time and also the Offset between duplications. If you want the link to be opened, you can try a different setting for DupEnd (Figure 22-24).

▼ Anim settings	
TrackX Y Z UpX Y Z	
Draw Key Draw Key S Powertrack SlowPa	
DupliFrames DupliVerts Rot No Speed	
🔹 DupSta: 1 🔺 🖌 DupOn: 1 🔶	
✓ DupEnd 35 → ✓ DupOff 0 →	
Offs Ob Offs Par Offs Particle 0.0000	
TimeOffset: 0.00 Automatic Time PrSpeed	

Figure 22-24. Values for DupliFrames. Note "DupEnd: 35" will end link before curve's end.

To turn the structure into a real NURBS-object, select the Surface Circle and press **CTRL-SHIFT-A**. A pop-up menu will appear prompting OK? Make Dupli's Real (Figure 22-25).



Figure 22-25. Making Dupli's Real.

Do not deselect anything. We now have a collection of NURBS forming the outline of our object, but so far they are not skinned, so we cannot see them in a shaded preview or in a rendering. To achieve this, we need to join all the rings to one object. Without deselecting any rings, press **CTRL-J** and confirm the pop-up menu request. Now, enter EditMode for the newly created object and press **AKEY** to select all vertices (Figure 22-26). Now we are ready to skin our object. Press **FKEY** and Blender will automatically generate the solid object. This operation is called "Skinning" and is fully described in the Section called *Skinning* in Chapter 9.



Figure 22-26. Skinning the link.

When you leave Edit Mode, you can now see the object in a shaded view. But it is very dark. To correct this, enter Edit Mode and select all vertices, then press **WKEY**. Choose Switch Direction from the menu and leave Edit Mode. The object will now be drawn correctly (Figure 22-27).

The object we have created is a NURBS object. This means that you can still edit it. Even more interestingly, you can also control the resolution of the NURBS object via the Edit Buttons.

Here you can set the resolution of the object using ResolU and ResolV, so you can adjust it for working with the object in a low resolution, and then set it to a high resolution for your final render. NURBS objects are also very small in file size for saved scenes. Compare the size of a NURBS scene with the same scene in which all NURBS are converted (ALT-C) to meshes.

Finally you can delete the curve we used to give the shape of the link, since we will not use it anymore.



Figure 22-27. Skinned link.

## Arranging objects with DupliFrames

Now we will continue modelling the chain itself. For this, just add a Curve Path (we could use a different curve but this one gives better results). In Edit Mode, move its vertices until get the desired shape of the chain (Figure 22-28). If not using a Curve Path, you should check the button 3D in the Edit Buttons to let the chain be real 3D.



Figure 22-28. Using a curve path to model the chain.

Select the object "Link" we modelled in the previous step and parent it to the chain curve, again as a normal parent. Since we are using a Curve Path the option CurvePath in the AnimButtons will be automatically activated, however the CurveFollow option will not, so you will have to activate it (Figure 22-29).

▼ Curve and Surface	
UV Orco	✓ DefResolU: 6 ► Set
Cantra	Back Front 3D
Centre	
Centre New	✓ Width: 1.000 ►
Centre Cursor	✓ Ext1: 0.000 →
◄ PathLen: 35 →	
CurvePath CurveFoll	BevResol: 0 >
PrintLen 0.0000	BevOb:

Figure 22-29. Curve settings.

If the link is dislocated, select it and press **ALT-O** to clear the origin. Until now we have done little more than animate the link along the curve. This can be verified by playing the animation with **ALT-A**.

Now, with the link selected once again go to the Object Context and Anim settings Panel. Here, activate the option DupliFrames as before. Play with the Dup-Sta:, DupEnd: and DupOf: NumButtons. Normally we are going to use DupOf: 0 but for a chain, if using DupOf: 0 the links are too close from each other you should change the value PathLen for the path curve to a lesser value, in the Editing Context and Curve and Surface Panel and then correspondingly change the DupEnd: value for the link to that number (Figure 22-30).

▼ Anim settings
TrackX Y Z UpX Y Z
Draw Key Draw Key S Powertrack SlowPa
DupliFrames DupliVerts Rot No Speed
🔹 DupSta: 1 🔺 🔹 DupOn: 1 🔶
■ DupEnd 35 ■ DupOff 0 ■
Offs Ob Offs Par Offs Particle 0.0000
TimeOffset: 0.00 Automatic Time PrSpeed

Figure 22-30. Adjusting the DupliFrames.

We need it so that the link rotates along the curve animation, so we have each link rotated 90 degrees with respect to the preceding one in the chain. For this, select the link and press Axis in the Edit Buttons to reveal the object's axis. Insert a rotation keyframe in the axis which was parallel to the curve. Move 3 or 4 frames ahead and rotate along that axis pressing **RKEY** followed by **XKEY-XKEY** (**XKEY** twice), **YKEY-YKEY**, or **ZKEY-ZKEY** to rotate it in the *local* X, Y or Z axis (Figure 22-31).


Figure 22-31. Rotating the link.

Open an IPO window to edit the rotation of the link along the path. Press the Extrapolation Mode so the link will continually rotate until the end of the path. You can edit the IPO rotation curve to make the link rotate exactly 90 degrees every one, two or three links (each link is a frame). Use **NKEY** to locate a node exactly at X=2.0 and Y=9.0, which correspond to 90 degrees in 1 frame (from frame 1 to 2).

Now we got a nice chain (Figure 22-32)!



Figure 22-32. Dupliframed chain.

### More Animation and Modelling

You are not limited to use Curve Paths to model your stuff. These were used just for our own convenience, however in some cases there are no need of them.

In Front View add a surface circle (you should know why by now Figure 22-33). Subdivide once, to make it look more like a square. Move and scale some vertices a little to give it a trapezoid shape (Figure 22-34).



Figure 22-33. A Surface Circle.



Figure 22-34. Trapezoidal cross-section.

Then rotate all vertices a few degrees. Grab all vertices and displace them some units right or left in X (but at the same Z location). You can use **CTRL** to achieve this precisely. Leave Edit Mode (Figure 22-35).



Figure 22-35. Trapezoidal cross section, rotated and translated.

From now on, the only thing we are going to do is editing IPO animation curves. So you can call this "Modelling with Animation" if you like. We will not enter Edit Mode for the surface any more.

Switch to Top View. Insert a KeyFrame for rotation at frame 1, go ahead 10 frames and rotate the surface 90 degrees over its new origin. Insert one more KeyFrame. Open an IPO window, and set the rotation IPO to Extrapolation Mode (Figure 22-36).



Figure 22-36. Rotation IPO for the cross section.

Go back to frame 1 and insert a keyframe for Location. Switch to Front View. Go to frame 11 (just press **UPARROW**) and move the surface in Z a few grid units. Insert a new keyframe for Location. In the IPO window set the LocZ to Extrapolation Mode (Figure 22-37).



Figure 22-37. Translation IPO for the cross section.

Now, of course, go to the Animation buttons and press DupliFrames. You can see how our surface is ascending in a spiral through the 3D space forming something like a spring. This is nice, however we want more. Deactivate DupliFrames to continue.

In frame 1 scale the surface to nearly zero and insert a keyframe for Size. Go ahead to frame 41, and clear the size with **ALT-S**. Insert a new keyframe for size. This IPO will not be in extrapolation mode since we don't want it scales up at infinitum, right (Figure 22-38)?



Figure 22-38. Size IPO for the cross section.

If you now activate DupliFrames you will see a beautiful outline of a corkscrew (Figure 22-39). Once again the last steps are: Make Duplis Real, Joining the surfaces, Select all vertices and skinning, Switch direction of normal if needed and leave Edit Mode (Figure 22-40).



Figure 22-39. Using a curve path to model the chain.



Figure 22-40. Using a curve path to model the chain.

You can see this was a rather simple example. With more IPO curve editing you can achieve very interesting and complex models. Just use your imagination.

# **Modelling with lattices**

### Relevant to Blender v2.31

A Lattice consists of a non-renderable three-dimensional grid of vertices. Their main use is to give extra deformation to any child object they might have. These child objects can be Meshes, Surfaces and even Particles.

Why would you use a Lattice to deform a mesh instead of deforming the mesh itself in EditMode?

There are a couple of reasons for that:

- 1. First of all: It's easier. Since your mesh could have a zillion vertices, scaling, grabbing and moving them could be a hard task. Instead, if you use a nice simple lattice your job is simplified to move a couple of vertices.
- 2. It's nicer. The deformation you get looks a lot better!
- 3. It's fast! You can put all or several of your child objects in a hidden layer and deform them all at once.
- 4. It's a good practice. A lattice can be used to get different versions of a mesh with minimal extra work and consumption of resources. This leads to an optimal scene design, minimizing the amount of modelling job. A Lattice does not affect the texture coordinates of a Mesh Surface. Subtle changes to mesh objects are easily facilitated in this way, and do not change the mesh itself.

### How does it work?

A Lattice always begins as a  $2 \times 2 \times 2$  grid of vertices (which looks like a simple cube). You can scale it up and down in Object Mode and change its resolution through the Lattice Panel in the Editing Context Buttons U, V, W.

After this initial step you can deform the Lattice in Edit Mode. If there is a Child Object, the deformation is continually displayed and modified. Changing the U, V, W values of a Lattice returns it to a uniform starting position.

Now we are going to see a very simple case in which having a lattice will simplify and speed up our modelling job.

I have modelled a very simple fork using a plane subdivided couple of times. It looks really ugly but it's all I need. Of course it is completely flat from a Side View. Wow, it is *really* ugly (Figure 22-41). The only important detail is that it has been subdivided enough to ensure a nice deformation in the Lattice step. You cannot bend a two vertices segment!



Figure 22-41. An ugly fork.

In Top View, now add a Lattice. Before changing its resolution, scale it up so it completely envelopes the fork's width (Figure 22-42). This is very important. Since I want to keep the lattice vertices count low (it doesn't make sense it has the same number of vertices than the mesh, right?) I need to keep resolution low but still set the lattice to convenient size.



Figure 22-42. A 2x2x2 Lattice.

Adjust the Lattice resolution to complete the fork's length (Figure 22-43).



Figure 22-43. Use a suitable resolution, but don't exaggerate.

Now, we are ready for the fun part. Parent the fork to the lattice, by selecting the fork and the lattice and pressing **CTRL-P**. Enter Edit Mode for the lattice and start selecting and scaling vertices (Figure 22-44). You might want to scale in X or Y axis separately to have more control over the lattice depth (to avoid making the fork thicker or thinner).



Figure 22-44. Deforming the lattice is a pleasure!

Note that if you move the fork up ad down inside the lattice, the deformation will apply in different parts of the mesh.

Once you're done in Front View, switch to Side View. Select and move different vertices sections to give the fork the suitable bends (Figure 22-45).



Figure 22-45. Bending things.

You can get rid of the lattice now if you're not adding any other child object. But before doing it, you might want to keep your deformations! Just select the fork and press **CTRL-SHIFT-A** and click on the Apply Lattice Deform? menu entry.

**Mad vertices:** On rare occasions, for fairly complex meshes, application of **CTRL-SHIFT-A** will *look like* it has screwed your mesh completely. This is false. Just step in and out of EditMode (**TAB**) and the mesh will be back nicely deformed as you expected.



Figure 22-46. A nice fork.

You can use a lattice to model an object following another object's shape. For instance take a look at the following scene. I have modelled a bottle, and now I would like to confine a character inside it. He deserves it (Figure 22-47).



Figure 22-47. Poor guy...

Add a lattice around the character. I didn't use a too high resolution for the lattice. I scaled it in X and Y to fit the lattice to the character (Figure 22-48).



Figure 22-48. Bending things.

Parent the character to the lattice, and then scale the lattice again to fit the dimensions of the bottle (Figure 22-49).



Figure 22-49. Scale the lattice to fit the bottle.

Now enter Edit Mode for the lattice. Press the Outside button in the Lattice Panel in the Editing Context to switch off the inner vertices of the lattice. We will switch them on later. Move and scale the vertices in front and side views until the character perfectly fits the bottle's shape (Figure 22-50).



Figure 22-50. Edit Lattice so that the poor guy is comfortable in his bottle.

You can select the lattice and do the modelling in one 3D window using Local View and see the results in another window using Global View to make your modelling comfortable (Figure 22-51).



Figure 22-51. Claustrophobic?

Hadn't we used a lattice it would have taken a lot more of vertex picking-and-moving work to deform the character (Figure 22-52).

Since lattices also supports RVK for vertex animation, quite interesting effects can be achieved with this tool.



Figure 22-52. Final Render. Believe me, he deserved it!

Lattices can be used in many applications which require a "liquid-like" deformation of a mesh. Think of a genie coming out of his lamp, or a cartoon character with its eyes popping out exaggeratedly. And have fun!

# Notes

1. and also Object Mode, however scaling in Object Mode could bring up some problems when applying Rotation to DupliVerts as we will see soon

# **Chapter 23. Volumetric Effects**

### Relevant to Blender v2.31

Although Blender exhibits a very nice Mist option in the World Settings to give your images some nice depth, you might want to create true volumetric effects; mists and clouds and smoke which really looks like they occupy some space.

Figure 23-1 shows a set-up with some columns placed in a ring, with some nice materials of your choice for the columns and soil, and a World defining sky color.



Figure 23-1. Columns on a plane.

Figure 23-2 shows the relative rendering, whereas Figure 23-3 shows a rendering with Blender's built-in Mist. Mist setting in this particular case are: Linear Mist, Sta=1, Di=20, Hig=5.



Figure 23-2. A plain rendering.



Figure 23-3. A rendering with built-in Blender Mist.

But we want to create some truly cool, swirling, and most importantly, non-uniform mist. Blender's built-in procedural textures (clouds for example) are intrinsically 3D, but are rendered only when mapped onto a 2D surface. We will achieve a 'volumetric'-like rendering by 'sampling' the texture on a series of mutually parallel planes. Each of our planes will hence exhibit a standard Blender texture on its 2D surface, but the global effect will be of a 3D object. This concept will be clearer as the example proceeds.

With the camera at z=0, looking forward, turn to front view and add a plane in front of the camera, with its centre aligned with the camera's viewing direction. In side view move the plane where you want your volumetric effect to terminate. In our case somewhere beyond the furthest column. Scale the plane so that it encompasses the whole of the camera's field of view (Figure 23-4). It is important to have a camera pointing along the y axis since we need the planes to be orthogonal to the line of sight. Anyway, we will be able to move it later on.



Figure 23-4. The plane set-up.

After having checked that we're at frame 1, let's place a Loc KeyFrame (**IKEY**). We should now move to frame 100, move the plane much nearer to the camera, and set another Loc KeyFrame. Now, in the Object Context Anim Settings Panel (**F7**) Press the DupliFrame button.

The 3D window, in side view, will show something like Figure 23-5. This is not good because the planes are denser at the beginning and end of the sweep. With the plane still selected change a window to an IPO window (SHIFT-F6). There will be three Loc IPOs, only one of which is non-constant. Select it, switch to Edit Mode (TAB) and select both control points. Now turn them from smooth to sharp with (VKEY) (Figure 23-6).



Figure 23-5. The Dupliframed plane.



Figure 23-6. Reshaping the Dupliframed Plane IPO.

The planes will now look as in Figure 23-7. Parent the DupliFramed planes to the camera (select the plane, **SHIFT** select the camera, **CTRL-P**). You now have a series of planes automatically following the camera, always oriented perpendicularly to it. From now on you could move the camera if you so wish.

### Chapter 23. Volumetric Effects



Figure 23-7. Reshaping the DupliFramed Plane IPO.

> MA:Material	Lambert = Ref 0.800 Halo
ME:Plane.001 OB ME 1 Mat 1	Shadow
VCol Light VCol Paint TexFace Shadeless	CookTorr  Spec 0.500 Radio Wire Wire
Sne G 0.800	ZTransp
B 0.800	Amb 0.5
Alpha 0.200         Image: Control of the second secon	Add 0.0 Add 0.0 Zinvert

Figure 23-8. Basic Material settings.

Now we must add the Mist material itself. The material should be Shadeless and cast no shadows to avoid undesired effects. It should have an small Alpha value (Figure 23-8). A material like this would basically act like Blender's built in mist, hence we would have no advantage in the resulting image. The drawback is that computing 100 transparent layers is very CPU intensive, especially if one desires the better results of the Unified Renderer.

**Quick previews:** You can use the DupOff: Num Button in the Anim Settings Panel to turn off some of the planes and hence have a faster, lower quality preview of what you are doing. For the final rendering you will then turn DupOff back to 0.

Pay attention to the Alpha value! The fewer planes you use the thinner the mist will be, so your final rendering will be much more 'Misty' than your previews!

The truly interesting stuff comes when you add textures. We will need at least two: One to limit the Mist in the vertical dimension and keep it on the ground; The second to make it non-uniform and with some varying hue.

As a first texture Add a Blend texture of "linear" type, with a very simple colorband, going from pure white, Alpha=1 at a position 0.1 to pure white, Alpha=0 at a position

0.9 (Figure 23-9). Add this only to the Alpha channel and as a multiplying (Mul Button) texture (Figure 23-10). To make our mist consistent as the Camera moves, and the planes follow, we have to set it Global. This will be true also for all other textures and will make the planes sample a fixed 3D volumetric texture. If you are planning an animation you will see a static mist, with respect to the scene, while the camera moves. Whichever other texture setting would show a Mist which is static with respect to the camera, hence being always the same while the camera moves, which is highly unrealistic.



Figure 23-9. Height limiting texture.

Texture Map Input Map To	Texture Map Input Map To	Texture Map Input Map To
Fading TE:Threads	UV Object Glob Orco Stick Win Nor Refl	Col Nor Csp Cmir Ref Spec Hard Alpha Emit
Clear 1	Flat         Cube         ofsX 0.000           Tube         Sphe         ofsY 0.000           ofsZ 0.000         ofsZ 0.000	Stenc Neg No RGB Mix Mul Add Sub
	X         Y         Z           X         Y         Z           X         Y         Z           sizeY         1.00           X         Y           Z         sizeY           X         Y	R 1.000         G 0.000         G 0.000         G 0.000         I           B 1.000         I         Nor 0.500         I         I           DVar 1.00         I         Var 1.000         I         I

Figure 23-10. Basic Material settings for cloud texture.

Anyway, if you want to have a moving, swirling, changing mist you can do so by animating the texture, as will be explained later on.

The Blend texture operates on X and Y directions, so if you want it to span vertically in the Global coordinates you will have to remap it (Figure 23-10). Please note that the blending from Alpha=1 to the Alpha=0 will occur from global z=0 to global z=1 unless additional offsets and scalings are added. For our aims the standard settings are OK.

If you now do a rendering, it doesn't matter where your camera, and planes, are. The mist will be thick below z=0, non-existent above z=1 and fading in between. If you're puzzled by this apparent complexity, think of what you would have got with a regular Orco (ORiginal COordinate) texture and non-parented planes. If you had to move the camera, especially in animations, the results would become very poor as soon as the planes were no longer perpendicular to the camera. You'd end up with no mist at all if the camera were to become parallel to the planes!

The second texture is the one giving the true edge on the built-in mist. Add a Cloud texture, make its Noise Size=2, Noise Depth=6 and Hard Noise On (Figure 23-11). Add colorband to this too, going from pure white with Alpha=1 at Position 0 to a pale bluish-grey with Alpha=0.8 at a position of about 0.15, to a pinkish hue with Alpha=0.5 around position 0.2, ending in a pure white, Alpha=0 colour at position 0.3. Of course, you might want to go to a greenish-yellow for swamp mists etc.

	▼ Texture Co	lors
Mat World	Colorband Add <	Cur: 2   Del
Lamp	4Pos 0.219 E L S A	x 0.500
Default Vars	Bright1.000 Con	tr1.000

Figure 23-11. Cloud texture settings.

Use this texture on both Col and Alpha as a Mul texture, keeping all other settings as default. If you now render the scene the bases of your columns will now be masked by a cool mist (Figure 23-12). Please note that the Unified Renderer gives much better results here.



Figure 23-12. Final rendering.

**Note:** If you are planning an animation and want your Mist to be animated as if it were moved by wind, it is this latter texture you must work on. Add a Material texture IPO, be sure to select the correct texture channel and add some IPO to the <code>ofsx, ofsy</code> and <code>ofsz</code> properties.

# **Chapter 24. Sequence Editor**

An often underestimated function of Blender is the Sequence Editor. It is a complete video editing system that allows you to combine multiple video channels and add effects to them. Even though it has a limited number of operations, you can use these to create powerful video edits (especially when you combine it with the animation power of Blender!) And, furthermore, it is extensible via a Plugin system quite alike the Texture plugins.

# Learning the Sequence Editor

Relevant to Blender v2.31

This section shows you a practical video editing example exhibiting most of the Sequence Editor built in features. We will put together several Blender made animations to obtain some stunning effects. One frame of the resulting edited animation is in Figure 24-1.



Figure 24-1. Final result.

## First Animation: two cubes

Let's start with something simple and see where it leads. Start a clean Blender and remove the default plane. Split the 3D window and switch one of the views to the camera view with **NUM0**. In the top-view, add a cube and move it just outside of the dotted square that indicates the camera view (Figure 24-2).



Figure 24-2. Moving the cube out of the camera view.

We want to create a simple animation of the cube moving into view, rotating once, and then disappearing. Set the animation end to 61 (set the End: value in the Anim Panel of the Scene Context, Render Buttons F10) and insert a LocRot KeyFrame on frame 1 with IKEY and selecting LocRot from the menu which appears. This will store both the location and the rotation of the cube on this frame.

Go to frame 21 (press **UPARROW** twice) and move the cube closer to the camera. Insert another KeyFrame. On Frame 41, keep the cube on the same location but rotate it 180 degrees and insert another KeyFrame.

Finally on frame 61 move the cube out of view, to the right and insert the last KeyFrame.

We will need two versions of the animation: one with a solid material and one with a WireFrame. For the material, we can use a plain white lit by two bright lamps - a white one and a blue one with an energy value of two (Figure 24-3).

For the WireFrame cube, set the material type to 'Wire' and change the color to green (Figure 24-4).



Figure 24-3. A rendering of the solid cube...



Figure 24-4. ...and a rendering of the WireFrame cube.

Enter an appropriate filename (for example 'cube\_solid.avi') in the Pics field (first text button on top) of the Scene Context Render sub-context Output Panel (F10) (Figure 24-5).

<ul> <li>Output</li> </ul>	
C //render/cube_solid	l.avi
2 11	
6 11	
\$	
Backbuf	Edge Edge Settings
DispView	DispWin Extension

Figure 24-5. Set the animation output filename.

Render the animation with the white solid cube. This will save it to your disk. Save it as an AVI file. Use AVI Raw if possible, because it yelds an higher quality - compression should be the last thing in the editing process - otherwise, if short of disk space use AVI Jpeg or AVI Codec, the first being less compressed and hence often of higher quality.

Now change the material to the green wire frame, render the animation again, saving the result as cube\_wire.avi.

You now have a 'cube\_solid.avi' and 'cube\_wire.avi' on your hard disk. This is enought for our first sequence editing.

### First Sequence: delayed wireframes

The first sequence will use only the wireframe animation - twice - to create an interesting effect. We will create multiple layers of video, give them a small time offset and add them together. This will simulate the 'glowing trail' effect that you see on radar screens. Start a clean Blender file and change the 3D window to a Sequence Editor window by pressing **SHIFT-F8** or by selecting the Sequence Editor icon **Wideo** Sequence Editor from the window header Window Type Menu.

Add a movie to the window by pressing **SHIFT-A** and selecting Movie (Figure 24-6) or by using the Add>>Movie Menu entry. From the File Select Window wich appears select the wireframe cube animation that you made before.

Add sequence
Images
Movie
Scene
Plugin
Cross
GammaCross
Add
Sub
Mul
AlphaOver
AlphaUnder
AlphaOverDrop

Figure 24-6. Adding a video strip

After you have selected and loaded the movie file, you will see a blue strip that represents it. After adding a strip, you are automatically in grab mode and the strip follows the mouse. The start and end frame are now displayed in the bar.

Take a closer look at the Sequence Editor screen now. Horizontally you see the time value. Vertically, you see the video 'channels'. Each channel can contain an image, a movie or an effect. By layering different channels on top of each other and applying effects, you can mix different sources together. If you select a video strip, its type, length and filename will be printed at the bottom of the window.

Move your video strip and let it start at frame 1. Place it in channel 1, that is on the bottom row and press **LMB** to finalize (Figure 24-7).



Figure 24-7. Placing the strip.

**Lead-in, Lead-out and stills:** You can add *lead-in* and *lead-out* frames by selecting the triangles at the start and end of the strip (they will turn purple) and dragging them out. In the same way, you can define the 'length' in frames of a still image.

Duplicate the movie strip with **SHIFT-D**, place the duplicate in channel 2 and shift it one frame to the right. We now have two layers of video on top of each other, but only one will display. To mix the two layers you need to apply an effect to them.

Select both strips and press **SHIFT-A**. Select ADD from the menu that pops up (Figure 24-8). Otherwise use the Add>>Effect>>Add.



Figure 24-8. Mixing two video strips

To see what's happening split the sequence editor window and select the image button in the header (Figure 24-9). This will activate the automatic preview (Figure 24-10). If you select a frame in the sequence editor window with the strips, the preview will be automatically updated (with all the effects applied!).



Figure 24-9. Sequence Editor preview button.

If you press **ALT-A** in the preview window, Blender will play back the animation. (Rendering of effects for the first time takes a lot of processing time, so don't expect a real-time preview!).



Figure 24-10. Adding a preview window.

**Windowless preview:** If you do not like the separate render window, switch to the Render Buttons (**F10**) and select DispView in the bottom left.

Now it's time to add some more mayhem to this animation. Duplicate another movie layer and place it on channel 4. Add it to the existing ADD effect in video channel 3 with a new ADD effect. Repeat this once and you will have four WireFrame cubes in the preview window (Figure 24-11).



Figure 24-11. Sequence with 4 WireFrame cube strips added together.

All the cubes have the same brightness now, but I would like to have a falloff in brightness. This is easily arranged: open an IPO window somewhere (**F6**) and select the sequence icon in its IPO Type Menu (Figure 24-12).



Figure 24-12. Sequence IPO button.

Select the first add strip (the one in channel 3), hold down **CTRL** and click **LMB** in the IPO window on a value of 1. This sets the brightness of this add operation to maximum. Repeat this for the other two add strips, but decrease the value a bit for each of them, say to around 0.6 and 0.3 (Figure 24-13).



Figure 24-13. Defining the brightness of a layer with an IPO

Depending on the ADD values that you have just set, your result should look something like what is shown in Figure 24-14.



Figure 24-14. Four WireFrame cubes combined with fading effects.

Now we already have 7 strips and we have only just begun with our sequencing! You can imagine that the screen can quickly become very crowded indeed. To make your project more manageable, select all strips (AKEY and BKEY work here, too!), press MKEY and press ENTER or click on the Make Meta pop up. Otherwise you can use the Strip>>Make Meta Strip Menu entry. The strips will now be combined into a meta-strip, and can be copied or moved as a whole.

With the meta strip selected, press **NKEY** and enter a name, for example 'Wire/Delay', to better remember what it is (Figure 24-15).



Figure 24-15. Named META strip

## Second Animation: A delayed solid cube

Now it is time to use some masks. We want to create two areas in which the animation plays back with 1 frame time difference. This creates a very interesting glass-like visual effect.

Start by creating a black and white image like the one in Figure 24-16. You can use a paint program or do it in Blender. The easiest way to do this in Blender is to create a white material with an emit value of 1 or a shadeless white material on some bevelled Curve Circles. In this way, you do not need to set up any lamps. Save the image as mask.tga.



Figure 24-16. Animation mask.

Switch to the sequence editor and move the meta strip that we made before out of the way (we will reposition it later). Add the animation of the solid cube (**SHIFT-A**>>Movie). Next, add the mask image. By default a still image will get a length of 50

frames in the sequence editor. Change it to match the length of the cube animation by **RMB** and **GKEY** to dragging out the arrows on the side of the image strip with the right mouse button.

Now select both strips (hold down **SHIFT**), press **SHIFT-A** and add a SUB (subtract) effect (Figure 24-17).



Figure 24-17. Subtracting the mask from the video.

In the preview window you will now see the effect; the areas where the mask is white have been removed from the picture (Figure 24-18).



Figure 24-18. Mask subtracted.

This effect is ready now; select all three strips and convert them into a META strip by pressing **MKEY**.

Now repeat the previous steps, except that you don't use the SUB effect but the MUL (multiply) effect (Figure 24-19). This time you will only see the original image where the mask image is white. Turn the three strips of this effect into a meta strip again.



Figure 24-19. Mask multiplied.

For the final step I have to combine the two effects together. Move one of the meta strips above the other one and give it a time offset of one frame. Select both strips and add an ADD effect (Figure 24-20).



Figure 24-20. Adding the two effects

In the preview window you can now see the result of the combination of the animation and the mask (Figure 24-21).

#### Chapter 24. Sequence Editor

When you are ready, select the two meta strips and the ADD effect and convert them into a new meta strip. (That's right! You can have meta strips in meta strips!)

**Getting into a Meta Strip:** To edit the contents of a meta strip, select it and press **TAB**. The meta strip will 'explode' to show its components and the background will turn yellow-ish/green to indicate that you are working inside a meta strip. Press **TAB** again to return to normal editing.



Figure 24-21. Two time-shifted layers.

### Third Animation: a tunnel

We want a third 'effect' to further enrich our animation; a 3D 'tunnel' to be used as a background effect. This is really simple to create. First save your current work - you will need it later!

Start a new scene (**CTRL-X**) and delete the default plane. Switch to front view (**NUM1**). Add a 20-vertex circle about 10 units under the z=0 line (the pink line in your screen) (Figure 24-22).



Figure 24-22. Adding a 20-vertex circle.

While still in Edit Mode, switch to side view (**NUM3**) and snap the cursor to the origin by locating it roughly at the x,y,z=0 point and pressing **SHIFT-S**. Select Curs>>Grid.

We want to turn the circle into a circular tube, or torus. For this, we will use the Spin function. Go to the Editing Context (F9) and enter a value of 180 in the Degr NumButton and enter '10' in the Steps NumButton in the Mesh Tools Panel. Pressing Spin will now rotate the selected vertices around the cursor at 180 degrees and in 10 steps (Figure 24-23).



Figure 24-23. Spinning the circle around the cursor

Leave Edit Mode (**TAB**). With the default settings, Blender will always rotate and scale around the object's center which is displayed as a tiny dot. This dot is yellow when the object is unselected and pink when it is selected. With the cursor still in the origin, press the Center Cursor button in the Edit Buttons window to move the object center to the current cursor location. Now press **RKEY** and rotate the tube 180 degrees around the cursor.

Now it's time to move the camera into the tunnel. Open another 3D window and switch it to the camera view (**NUM0**). Position the camera in the side view window to match Figure 24-24, the camera view should now match Figure 24-25.

**Missing edges:** If not all of the edges of the tunnel are showing, you can force Blender to draw them by selecting All Edges Tog Button in the Mesh Tools 1 Panel of the Editing Context (F9).



Figure 24-24. Camera inside the tunnel.



Figure 24-25. Camera view of the tunnel interior.

To save ourselves some trouble, I want to render this as a looping animation. I can then add as many copies of it as I like to the final video compilation.

There are two things to keep in mind when creating looping animations. First, make sure that there is no 'jump' in your animation when it loops. For this, you have to be careful when creating the KeyFrames and when setting the animation length. Create two KeyFrames: one with the current rotation of the tube on frame 1, and one with a rotation of 90 degrees (hold down **CTRL** while rotating) on frame 51. In your animation frame 51 is now the same as frame 1, so when rendering you will need to leave out frame 51 and render from 1 to 50.

Please note that the number 90 degrees is not chosen carelessly, but because the tunnel is periodic with period 18, hence you must rotate it by a multiple of 18, and 90 is it, to guarantee that frame 51 is exactly the same than frame 1.

Second, to get a *linear* motion you need to remove the ease-in and ease-out of the rotation. These can be seen in the IPO Window of the tube after inserting the rotation KeyFrames. The IPO smoothly starts and end, much like a cosine function. We want it to be straight. To do so select the rotation curve, enter editmode (**TAB**) and select all vertices (**AKEY**) and press **VKEY** ('Vector') to change the curve into a linear one (Figure 24-26).



Figure 24-26. Tunnel rotation IPO without ease-in and ease-out.

To create a more dramatic effect, select the camera while in camera view mode (Figure 24-27). The camera itself is displayed as the solid square. Press **RKEY** and rotate it a bit. If you now play back your animation it should loop seamlessly.



Figure 24-27. Rotate the camera to get a more dramatic effect

For the final touch, add a blue WireFrame material to the tube and add a small lamp on the location of the camera. By tweaking the lamp's Dist value (attenuation dis-

#### Chapter 24. Sequence Editor

tance) you can make the end of the tube disappear in the dark without having to work with mist. (Figure 24-28).

When you are satisfied with the result, render your animation and save it as 'tunnel.avi'.



Figure 24-28. A groovy tunnel.

### Second Sequence: Using the tunnel as a backdrop

Reload your video compilation Blender file. The tunnel that we made in the last step will be used as a backdrop for the entire animation. To make it more interesting I will modify an ADD effect to change the tunnel into a pulsating backdrop. Prepare a completely black picture and call it 'black.tga' (try pressing **F12** in an empty Blender file. Save with **F3**, but make sure that you have selected the TGA file format in the Render Buttons window). Add both black.tga and the tunnel animation and combine them with an ADD effect (Figure 24-29).


Figure 24-29. Setting up the backdrop effect.

Now with the ADD effect selected, open an IPO window and select the Sequence Editor button in its header. From frame 1-50, draw an irregular line by holding down **CTRL** and left-clicking. Make sure that the values are between 0 and 1 (Figure 24-30).



Figure 24-30. Adding randomnes with a irregular Ipo

When you are ready, take a look at the result in a preview screen and change the animation into a meta strip.

Save your work!

## Fourth Animation: a jumping logo

Let's create some more randomness and chaos! Take a logo (We can just add a text object) and make it jump through the screen. Again, the easiest way to do this is to

#### Chapter 24. Sequence Editor

add vertices directly into the IPO window (select a LocX, LocY or LocZ channel first), but this time you may need to be a bit more careful with the minimum and maximum values for each channel. Don't worry about the looks of this one too much - the next step will make is hardly recognizable anyway (Figure 24-31).



Figure 24-31. Jumping logo

Save the animation as 'jumpylogo.avi'.

### Fifth Animation: particle bars

Our last effect will use an animated mask. By combining this with the logo of the previous step, I will achieve a streaking effect that introduces the logo to our animation. This mask is made by using a particle system. To set one up switch to side view, add a plane to your scene and while it is still selected switch to the Object Context (F7) in the Effects Tab of the Constraints Panel. Select New effect and then change the default effect build to Particles. Change the system's settings as indicated in Figure 24-32.

Constraints	Ef	fects
NEW Effect	Delete RecalcAll	Static Particles 👳
≪ Tot: 100 ≻ ≪Sta	: 1.00 End: 100.0	0 (Life: 40.00) < Keys: 8>
⊲CurMul: 0⊧   ∢ M	at: 1 🔊 🗐 Mult: 0.000	) Life: 50.00 Child: 4
ndlife: 0.000	eed: 0 Face Bsp	lin Vect VectSize 0.000
Norm: 0.000 Ob	:0.000 and:0.000	Tex: 0.000 amp: 0.000
X: 0.614 V: 0.00 Force: Z: 0.00	00 (X: 0.000 ) 00 Texture: (2	(: 0.000 Int RGB Grad) (: 1.000 Int abla: 0.050

Figure 24-32. Particle system settings.

Press **TAB** to enter Edit Mode, select all vertices and subdivide the plane twice by pressing **WKEY** and selecting Subdivide from the pop-up menu.

Next switch to front view and add another plane. Scale it along the X-axis to turn it into a rectangle (press **SKEY** and move your mouse horizontally. Then press **XKEY** or **MMB** to scale along the indicated axis only). Give the rectangle a white material with an emit value of one.

Now you need to change the particles into rectangles by using the dupliverts function. Select rectangle, then particle emitter and parent them. Select only the plane and in the Object Context and Anim Settings Panel, select the Dupliverts Button. Each particle is now replaced by a rectangle (Figure 24-33).



Figure 24-33. DupliVerted rectangles

I now add some mist as a quick hack to give the rectangles each a different shade of grey. Go to the World Buttons window with **F5** to change to Shading Context, then click on the button and select Add New in the World Panel. The world settings will now appear.

By default, the sky will now be rendered as a gradient between blue and black. Change the horizon colors (HoR, HoG, HoB) to pure black (Figure 24-34).



Figure 24-34. Setting up mist.

To activate rendering of mist activate the Mist button in the middle of the screen. When using mist, you have to indicate on which distance from the camera it works. Select the camera, switch to the Editing Context enable ShowMist in the Camera Panel. Now switch to top view and return to the Shading Context (F5) and World Buttons. Tweak the Sta: and Di: (Start, Distance, respectively) parameters so that the mist covers the complete width of the particle stream (Figure 24-34 and Figure 24-35).



Figure 24-35. Setting the mist parameters

Set the animation length to 100 frames and render the animation to disk. Call the file 'particles.avi' (Figure 24-36).



Figure 24-36. Rendered particle rectangles.

## Third Sequence: Combining the logo and the particle bars

By now you know the drill: reload your compilation project file, switch to the Sequence Editor window and add both 'particles.avi' and 'logo.avi' to your project. Combine them together with a MUL effect. Since the logo animation is 50 frames and the particles animation is 100 frames, you'll need to duplicate the logo animation once and apply a second MUL effect to it (Figure 24-37 and Figure 24-37).



Figure 24-37. Use the logo animation twice

Combine these three strips into one meta strip. If you're feeling brave you can make a few copies and give them a small time offset just like with the WireFrame cube.



Figure 24-38. The particles animation combined with the logo animation

### Sixth Animation: zooming logo

If you would combine all your animations so far you would get a really wild video compilation, but if this was your company's presentation you would want to present the logo in a more recognizable way. The final part of our compilation will therefore be an animation of the logo that zooms in very slowly. Prepare this one and save it as 'zoomlogo.avi'. Also prepare a white picture and save it as 'white.tga'.

#### Chapter 24. Sequence Editor

We will now use the CROSS effect to first make a rapid transition from black to white, then from white to our logo animation. Finally, a transition to black will conclude the compilation.

Start off by placing black.tga in channel 1 and white.tga in channel 2. Make them both 20 frames long. Select them both and apply a cross effect. The cross will gradually change the resulting image from layer 1 to layer 2. In this case, the result will be a transition from black to white (Figure 24-39).



Figure 24-39. Black-white transition.

Next, add a duplicate of white.tga to layer 1 and place it directly to the right of black.tga. Make it about half as long as the original. Place the logo zoom animation in layer 2 and add a cross effect between the two. At this point, the animation looks like a white flash followed by the logo zoom animation (Figure 24-40).



Figure 24-40. White-video transition

The last thing that you need to do is to make sure that the animation will have a nice transition to black at the very end. Add a duplicate of black.tga and apply another cross effect. When you are ready, transform everything into a meta strip (Figure 24-41).



Figure 24-41. Video-black transition

### Assembling everything so far

We're at the end of our work! It's time to add some of the compilations that we have made so far and see how our work looks. The most important thing to remember while creating your final compilation is that when rendering your animation, the sequence editor only 'sees' the top layer of video. This means that you have to make sure that it is either a strip that is ready to be used, or it should be an effect like ADD that combines several underlying strips.

The foundation of the compilation will be the fluctuating tunnel. Add a some duplicates of the tunnel meta strip and place them in channel one. Combine them into one meta strip. Do not worry about the exact length of the animation yet; you can always duplicate more tunnel strips.

On top of that, place the delayed wireframe cube in channel 2. Add channel 1 to channel 2 and place the add effect in channel 3 (Figure 24-42).



Figure 24-42. Combining the tunnel and the WireFrame cube

Now we also want to add the solid cube animation. Place it in channel 4, overlapping with the WireFrame animation in channel 2. Add it to the tunnel animation in layer one. This is where things are starting to get a little tricky; if you would leave it like this, the animation in channel 5 (the solid cube together with the tube) would override the animation in channel 2 (the wireframe cube) and the wireframe cube would become invisible as soon as the solid cube shows up. To solve this, add channel 3 to channel 5 (Figure 24-43).



Figure 24-43. Combining the tunnel, WireFrame and solid cube.

You will often need to apply some extra add operations to fix missing parts of video. This will most likely become apparent after you have rendered the final sequence.

Slide the Sequence Editor window a bit to the left and add the meta strip with the particle/logo animation in it. Place this strip in layer 2 and place an add effect in layer 3. For some variation, duplicate the WireFrame animation and combine it with the add in layer 3 (Figure 24-44).



Figure 24-44. Adding the particle/logo animation

Now go to the end of the tunnel animation strip. There should be enough place to put the logo zoom animation at the end and still have some space left before it (Figure 24-45). If not, select the tunnel strip, press **TAB** and add a duplicate of the animation to the end. Press **TAB** again to leave meta edit mode.



Figure 24-45. Adding the logo zoom animation.

If there is still some space left, we can add a copy of the solid cube animation. To get it to display correctly, you will have to apply two add channels to it: one to combine it with the particle logo animation and one to combine it with the logo zoom animation (Figure 24-46).



Figure 24-46. Adding one last detail

Figure 24-47 shows the complete sequence.

excore											5-6		5-6						
6			56 ADD: 3-5						3-4 3-4										
		[	62	ADD:	1-4			64 ADI	D: 3-4		62 A	DD: 1-	4						
4		lid Cube Anim							vire/Delay  lid Cube Anim										
2			64 A	DD: 1-	-2			104 AD	D: 1-2						122 A	DD: 1-	2		
2			64 Wi	re/Del	ayı		104 T	hree P	arti <b>cle</b> a	nims					122 Z	oom Lo	gol	-	
0								1 3	50 Multi	ple tur	inelstrip	s							
	0.0	0.2	1.1	2.1	3.0	4.0	42	52	6.1	7.1	8.0	82	9.1	10.1	11.1	12.0	12.2	13.1	14.1

Figure 24-47. The complete sequence

### Conclusion

We are now ready to render our final video composition! To tell Blender to use the Sequence Editor information while rendering, select the Do Sequence button in the Render Buttons window. After that, rendering and saving your animation works like before (be sure not to overwrite any of your AVI of the sequence!).

# **Sound Sequence Editor**

Relevant to Blender v2.31

Since Blender 2.28 there is a (still limited) Audio sequencing toolbox. You can add WAV files via the **SHIFT-A** menu and selecting the Sound entry.

A green audio strip will be created. No 'high level' mixing features are present currently. You can have as many Audio strips as you wish and the result will be the mixing of all them.

You can give each strip its own name and Gain (in dB) via the NKEY menu. This also let you set a strip to mute or 'Pan' it; -1 is hard left, +1 is hard right.

A 'Volume' IPO can be added to the strip in the IPO Window as it is done for effect strips. The Fac channel is the volume here. IPO frames 1-100 correspond to the whole sample length, 1.0 is full volume, 0.0 is completely silent.

Blender cannot yet mix the sound in the final product of the Sequence Editor. The output is therefore a video file, if the ANIM button in the Anim Panel of the Scene Context/Render Sub-context is used as described before, or a *separate* WAV file, containing the full audio sequence, in the same directory of the video file and with the same name but with a .WAV extension. This audio file is created via the MIXDOWN button in the Sequencer button of the Scene Context, Sound Sub-context.

You can mix Video and Audio later on with an external program. The advantage of using Blender's sequence editor lies in the easier synchronization attainable by sequencing frames and sound in the same application.

# **Sequence Editor Plugins**

Relevant to Blender v2.31

As said before Blender is extensible via a plugin system, and two kind of plugins may be found: Texture and Sequence plugins.

#### Chapter 24. Sequence Editor

Sequence plugins works on strips in a way similar to that of conventional ADD, CROSS etc. operation. You must have at least a strip selected and press **SHIFT-A**>>Plugin or Add>>Effect>>Plugin Menu entry. This opens a File Selection Window in which you can select the desired plugin.

Plugin functionalities varies so much that it is not possible to describe them here. Differently than Texture Plugins Sequence Plugins do not have a Buttons in any Button Window, but their parameters are usually accessed via **NKEY**.

# **Chapter 25. Python Scripting**

Relevant to Blender v2.31

Blender has a very powerful yet often overlooked feature. It exhibits an internal fully fledged Python interpreter.

This allows any user to add functionalities by writing a Python script. Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It was expressly designed to be usable as an extension language for applications that need a programmable interface, and this is why Blender uses it.

Of the two main ways of extending Blender, the other one being binary plugins, Python scripting is more powerful, versatile yet easier to comprehend and robust. It is generally preferred to use Python scripting than writing a plugin.

Actually Python scripting had somewhat limited functionalities up to Blender 2.25, the last of NaN releases. When Open Sourcing Blender many of the new developers gathered around the Foundation elected to work on it and, together with UI change, Python API is probably the single part of Blender which got the greatest development. A full reorganization of what existed was carried out and many new modules added.

This evolution is still ongoing and even better integration is expected in forthcoming Blender versions.

Blender has a Text Window among its windows types accessible via the **E** Text Editor button of the Window Type menu or via **SHIFT-F11**.

The newly opened Text window is grey and empty, with a very simple toolbar (Figure 25-1). From left to right there are the standard Window type selection button and the Window menu. Then the full screen button, followed by a toggle button which shows/hides the line numbers for the text and the regular Menu Button.

#### Figure 25-1. Text Toolbar.

The Menu Button ( ) allows you to select which Text buffer is to be displayed, as well as allowing you to create a new buffer or load a text file.

If you choose to load a file the Text Window tempoarily becomes a File Selection Window, with the usual functions. Once a text buffer is in the Text window, this behaves as a very simple text editor. Typing on the keyboard produces text in the text buffer. As usual pressing, **LMB** dragging and releasing **LMB** selects text. The following keyboard commands apply:

- ALT-C or CTRL-C Copy the marked text into the text clipboard;
- ALT-X or CTRL-X Cut out the marked text into the text clipboard;
- ALT-V or CTRL-V Paste the text from the clipboard to the cursor in the Text Window;
- ALT-S Saves the text as a text file, a File Selection Window appears;
- ALT-O Loads a text, a File Selection Window appears;
- **ALT-F** Pops up the Find toolbox;
- SHIFT-ALT-F or RMB Pops up the File Menu for the Text Window;

- **ALT-J** Pops up a Num Button where you can specify a linenumber the cursor will jump to;
- ALT-P Executes the text as a Python script;
- ALT-U Undo;
- ALT-R Redo;
- CTRL-R Reopen (reloads) the current buffer;
- ALT-M Converts the content of the text window into 3D text (max 100 chars);

Blender's cut/copy/paste clipboard is *separate* from Window's clipboard. So normally you *cannot* cut/paste/copy out from/into Blender. To access your Windows clipboard use **SHIFT-CTRL-C SHIFT-CTRL-V** 

To delete a text buffer just press the 'X' button next to the buffer's name, just as you do for materials, etc.

The most notable keystroke is **ALT-P** which makes the content of the buffer being parsed by the internal Python interpreter built into Blender.

The next section will present an example of Python scripting. Before going on it is worth noticing that Blender comes with only the bare Python interpreter built in, and with a few Blender-specific modules, those described in \*\*REF\*\*.

**Other usages for the Text window:** The text window is handy also when you want to share your .blend files with the community or with your friends. A Text window can be used to write in a README text explaining the contents of your blender file. Much more handy than having it on a separate application. Be sure to keep it visible when saving!

If you are sharing the file with the community and you want to share it under some licence you can write the licence in a text window.

to have access to the standard Python modules you need a complete working Python install. You can download this from http://www.python.org. Be sure to check on http://www.blender.org which is the *exact* Python version which was built into Blender to prevent compatibility issues.

Blender must also be made aware of *where* this full Python installation is. This is done by defining a PYTHONPATH environment variable.

#### Setting PYTHONPATH on Win95,98,Me

Once you have installed Python in, say, C:\PYTHON22 you must open the file C:\AUTOEXEC.BAT with your favourite text editor, add a line:

SET PYTHONPATH=C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB-TK

and reboot the system.

#### Setting PYTHONPATH on WinNT,2000,XP

Once you have installed Python in, say, C:\PYTHON22 Go on the "My Computer" Icon on the desktop, **RMB** and select Properties. Select the Advanced tab and press the Environment Variables button.

Below the System Variables box, (the second box), hit New. If you are not an administrator you might be unable to do that. In this case hit New in the upper box.

Now, in the Variable Name box, type PYTHONPATH, in the Variable Value box, type:

C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB-TK

Hit OK repeatedly to exit from all dialogs. You may or may not have to reboot, depending on the OS.

#### Setting PYTHONPATH on Linux and other UNIXes

Normally you will have Python already there. if not, install it. You will have to discover where it is. This is easy, just start a Python interactive shell by opening a shell and by typing python in there. Type the following commands:

```
>>> import sys
>>> print sys.path
```

and note down the output, it should look like

```
[", '/usr/local/lib/python2.2', '/usr/local/lib/python2.2 /plat-linux2', '/usr/local/lib
tk', '/usr/lo
cal/lib/python2.0/lib-dynload', '/usr/local/lib/python2.0/
site-packages']
```

Add this to your favourite rc file as an environment variable setting. For example, add in your .bashrc the line

```
export PYTHONPATH=/usr/local/lib/python2.2:/usr/local/lib/
python2.2/plat-linux2:/usr/local/lib/python2.2/lib-tk:/usr
/local/lib/python2.2/lib-dynload:/usr/local/lib/python2.0/
site-packages
```

all on a single line. Open a new login shell, or logoff and login again.

### A working Python example

#### Relevant to Blender v2.31

Now that you've seen that Blender is extensible via Python scripting and that you've got the basics of script handling and how to run a script, before smashing your brain with the full python API reference let's have a look at a quick working example.

We will present a tiny script to produce polygons. This indeed duplicates somewhat the **SPACE**Add>>Mesh>>Circle toolbox option, but will create 'filled' polygons, not just the outline.

To make the script simple yet complete it will exhibit a Graphical User Interface (GUI) completely written via Blender's API.

#### Headers, importing modules and globals.

The first 32 lines of code are listed in Example 25-1.

#### Example 25-1. Script header

```
002 #
003 # Demo Script for Blender 2.3 Guide
004 #
006 # This script generates polygons. It is quite useless
007 # since you can do polygons with ADD->Mesh->Circle
008 # but it is a nice complete script example, and the
009 # polygons are 'filled'
011
013 # Importing modules
015
016 import Blender
017 from Blender import NMesh
018 from Blender.BGL import *
019 from Blender.Draw import *
020
021 import math
022 from math import *
023
024 # Polygon Parameters
025 T_NumberOfSides = Create(3)
026 T_Radius
             = Create(1.0)
027
028 # Events
029 EVENT_NOEVENT = 1
030 EVENT_DRAW
          = 2
031 EVENT_EXIT
            = 3
032
```

After the necessary comments with the description of what the script does there is (lines 016-022) the importing of Python modules.

Blender is the main Blender Python API module. NMesh is the module providing access to Blender's meshes, while BGL and Draw give access to the OpenGL constants and functions and to Blender's windowing interface, respectively. The math module is Python's mathematical module, but since both the 'math' and the 'os' modules are built into Blender you don't need a full Python install for this!

The polygons are defined via the number of sides they have and their radius. These parameters have values which must be defined by the user via the GUI hence lines (025-026) create two 'generic button' objects, with their default starting value.

Finally, the GUI objects works with, and generates, events. Events identifier are integers left to the coder to define. It is usually a good practice to define mnemonic names for events, as is done here in lines (029-031).

### Drawing the GUI.

The code responsible for drawing the GUI should reside in a draw function (Example 25-2).

#### Example 25-2. GUI drawing

```
034 # GUI drawing
036 def draw():
037 global T_NumberOfSides
038 global T_Radius
   global EVENT_NOEVENT, EVENT_DRAW, EVENT_EXIT
039
040
041
    ######### Titles
042 glClear(GL_COLOR_BUFFER_BIT)
   glRasterPos2d(8, 103)
043
044 Text("Demo Polygon Script")
045
046
   ######### Parameters GUI Buttons
047
    glRasterPos2d(8, 83)
048
    Text("Parameters:")
049
    T_NumberOfSides = Number("No. of sides: ", EVENT_NOEVENT, 10, 55, 210, 18,
                    T_NumberOfSides.val, 3, 20, "Number of sides of out polygon");
050
                  = Slider("Radius: ", EVENT_NOEVENT, 10, 35, 210, 18,
051
    T_Radius
                    T_Radius.val, 0.001, 20.0, 1, "Radius of the polygon");
052
053
    ######### Draw and Exit Buttons
054
055
   Button("Draw", EVENT_DRAW , 10, 10, 80, 18)
056
    Button("Exit", EVENT_EXIT , 140, 10, 80, 18)
057
```

Lines (037-039) merely grant access to global data. The real interesting stuff starts from lines (042-044). The OpenGL window is initialised, and the current position set to x=8, y=103. The origin of this reference is the lower left corner of the script window. Then the title Demo Polygon Script is printed.

A further string is written (lines 047-048), then the input buttons for the parameters are created. The first (lines 049-050) is a Num Button, exactly like those in the various Blender Button Windows. For the meaning of all the parameters please refer to the API reference. Basically there is the button label, the event generated by the button, its location (x,y) and its dimensions (width, height), its value, which is a data belonging to the Button object itself, the minimum and maximum allowable values and a text string which will appear as a help while hovering on the button, as a tooltip.

Lines (051-052) defines a Num Button with a slider, with a very similar syntax. Lines (055-056) finally create a Draw button which will create the polygon and an Exit button.

### Managing Events.

The GUI is not drawn, and will not work, until a proper event handler is written and registered (Example 25-3).

#### Example 25-3. Handling events

```
058 def event(evt, val):
059 if (evt == QKEY and not val):
060 Exit()
061
062 def bevent(evt):
063 global T_NumberOfSides
064 global T_Radius
065 global EVENT_NOEVENT,EVENT_DRAW,EVENT_EXIT
066
```

```
067 ######### Manages GUI events
068 if (evt == EVENT_EXIT):
069 Exit()
070 elif (evt== EVENT_DRAW):
071 Polygon(T_NumberOfSides.val, T_Radius.val)
072 Blender.Redraw()
073
074 Register(draw, event, bevent)
075
```

Lines (058-060) define the keyboard event handler, here responding to the **QKEY** with a plain Exit() call.

More interesting are lines (062-072), in charge of managing the GUI events. Every time a GUI button is used this function is called, with the event number defined within the button as a parameter. The core of this function is hence a "select" structure executing different codes according to the event number.

As a last call, the Register function is invoked. This effectively draws the GUI and starts the event capturing cycle.

#### Mesh handling

Finally, Example 25-4 shows the main function, the one creating the polygon. It is a rather simple mesh editing, but shows many important points of the Blender's internal data structure.

#### Example 25-4. Script header

```
077 # Main Body
079 def Polygon(NumberOfSides,Radius):
080
081 ######### Creates a new mesh
082 poly = NMesh.GetRaw()
083
084 ######## Populates it of vertices
085 for i in range(0,NumberOfSides):
086 phi = 3.141592653589 * 2 * i / NumberOfSides
087
   x = Radius * cos(phi)
    y = Radius * sin(phi)
880
089
     z = 0
090
091
     v = NMesh.Vert(x,y,z)
092
    poly.verts.append(v)
093
094 ########## Adds a new vertex to the center
095 v = NMesh.Vert(0., 0., 0.)
096 poly.verts.append(v)
097
098 ########## Connects the vertices to form faces
099 for i in range(0,NumberOfSides):
100 f = NMesh.Face()
101 f.v.append(poly.verts[i])
102 f.v.append(poly.verts[(i+1)%NumberOfSides])
103 f.v.append(poly.verts[NumberOfSides])
104 poly.faces.append(f)
105
106 ########## Creates a new Object with the new Mesh
107 polyObj = NMesh.PutRaw(poly)
108
```

109 Blender.Redraw()

The first important line here is number (082). Here a new mesh object, poly is created. The mesh object is constituted of a list of vertices and a list of faces, plus some other interesting stuff. For our purposes the vertices and faces lists are what we need.

Of course the newly created mesh is empty. The first cycle (lines 085-092) computes the x,y,z location of the NumberOfSides vertices needed to define the polygon. Being a flat figure it is z=0 for all.

Line (091) calls the NMesh method Vert to create a new vertex object of co-ordinates (x,y,z). Such an object is then appended (line 096) in the poly Mesh verts list.

Finally (lines 095-096) a last vertex is added in the centre.

Lines (099-104) now connects these vertices to make faces. It is not required to create all vertices beforehand and then faces. You can safely create a new face as soon as all its vertices are there.

Line (100) creates a new face object. A face object has its own list of vertices v (up to 4) defining it. Lines (101-103) appends three vertices to the originally empty f.v list. The vertices are two subsequent vertices of the polygon and the central vertex. These vertices must be taken from the Mesh verts list. Finally line (104) appends the newly created face to the faces list of our poly mesh.

### Conclusions

If you create a polygon.py file containing the above described code and load it into a Blender text window, as you learned in the previous section, and press **ALT-P** in that window to run it, you will see the script disappearing and the window turn grey. In the lower left corner the GUI will be drawn (Figure 25-2).

Demo Polygon Script Parameters:	
No. of side	es: 3 🔹 🕨
Radius: 1.00	]
Draw	Exit
♥ File Edit	

Figure 25-2. The GUI of our example.

By selecting, for example, 5 vertices and a radius 0.5, and by pressing the Draw button a pentagon will appear on the xy plane of the 3D window (Figure 25-3).



Figure 25-3. The result of our example script.

# **Python Reference**

Relevant to Blender v2.31

The Full Python Application Programmer Interface of Blender has a reference documentation which is a book by itself. For space reason it is not included here.

Here it is :)<sup>3</sup>

# **Python Scripts**

Relevant to Blender v2.31

There are more than one hundred different scripts for Blender available on the net. As with plugins, scripts are very dynamic, changing interface, functionalities and web location fairly quickly, so for an updated list and for a live link to them please refer to one of the two main Blender sites, www.blender.org<sup>4</sup> or www.elysiun.com<sup>5</sup>.

# Notes

- 1. http://www.python.org
- 2. http://www.blender.org
- 3. http://www.blender.org/modules/documentation/228PythonDoc/Blendermodule.html
- 4. http://www.blender.org/
- 5. http://www.elysiun.com/

# Chapter 26. Blender's Plugins System

#### by Kent Mein

This section reports an in-depth reference for coding Blender's Texture and Sequence plugins.

# Writing a Texture Plugin

#### Relevant to Blender v2.31

In this Section we will write a basic texture plugin and then go through the steps to use a texture plugin. The basics behind a texture plugin is that you are given some inputs; position, and normal values as well as some other info. Then you return intensity, colour and/or normal information depending on the type of texture plugin.

All the files necessary to develop plugins as well as a few sample plugins can be found in the blender/plugins. You can alternately get a bunch of plugins from http://www.cs.umn.edu/~mein/blender/plugins<sup>1</sup>

Plugins are supported (loaded/called) in Blender using the dlopen() family of calls. For those unfamiliar with the dlopen system it allows a program (Blender) to use a compiled object as if it were part of the program itself, similar to dynamically linked libraries, except the objects to load are determined at runtime.

The advantage of using the dlopen system for plugins is that it is very fast to access a function, and there is no overhead in interfacing to the plugin, which is critical when as (in the case of texture plugins) the plugin can be called several million times in a single render.

The disadvantage of the system is that the plugin code works just like it is part of Blender itself, if the plugin crashes, Blender crashes.

The include files found in the plugin/include/ subdirectory of the Blender installation document the Blender functionality provided to the plugins. This includes the Imbuf library functions for loading and working with images and image buffers, and noise and turbulence functions for consistent texturing.

## **Specification:**

#### Relevant to Blender v2.31

• *#include <plugin.h>* 

Every Blender plugin should include this header file, which contains all of the structures and defines needed to properly work with Blender.

• char name[]="Tiles";

A character string containing the plugin name, this value will be displayed for the texture's title in the Texture Buttons window.

• #define NR\_TYPES 2 char stnames[NR\_TYPES][16]= {"Square", "Deformed"};

Plugins are allowed to have separate subtypes for minor variations on algorithms - for example the default clouds texture in Blender has the "Default" and "Color" subtypes.

NR\_STYPES should be defined to the number of subtypes required by your plugin, and a name for each subtype should be given. Every plugin should have at least 1 subtype and a subtype name.

• *VarStruct varstr[]*= {...};

The varstr contains all of the information Blender needs to display buttons for a plugin. Buttons for plugins can be numerical for input data, or text for comments and other information. Plugins are limited to a maximum of 32 variables.

Each VarStruct entry consists of a type, name, range information, and a tool tip.

*The type* defines the data type for each button entry, and the way to display the button. For number buttons this value should be a combination (ORed) of INT or FLO for the number format, and NUM, NUMSLI, or TOG, for the button type. Text buttons should have a type of LABEL.

*The name* is what will be displayed on (or beside) the button. This is limited to 15 characters.

*The range information* consists of three floats that define the default, minimum, and maximum values for the button. For TOG buttons the minimum is set in the pressed state, and the maximum is set in the depressed state.

*The tip* is a string that will be displayed when the mouse is over this button (if the user has tool tips on). This has a limit of 80 characters, and should be set to the NULL string ("") if unused.

typedef struct Cast {...};

The cast structure is used in calling the doit function, and serves as a way to simply access each plugin's data values.

The cast should contain, in order, an integer or float for every button defined in the varstr, including text buttons. Typically these should have the same name as the button for simple reference.

• float result[8];

The result array is used to pass information to and receive information from the plugin. The result values are mapped as follows:

Result Index	Significance	Range
result[0]	Intensity value	0.0 to 1.0
result[1]	Red color value	0.0 to 1.0
result[2]	Green color value	0.0 to 1.0
result[3]	Blue color value	0.0 to 1.0
result[4]	Alpha color value	0.0 to 1.0
result[5]	X normal displacement value	-1.0 to 1.0
result[6]	Y normal displacement value	-1.0 to 1.0
result[7]	Z normal displacement value	-1.0 to 1.0

The plugin should always return an intensity value. Returning RGB or a normal are optional, and should be indicated by the doit() return flag "1" (RGB) or "2" (Normal).

Before the plugin is called, Blender includes the current rendering-normal in result[5], result[6] and result[7]. • float cfra

The cfra value is set by Blender to the current from before every render pass. This value is an the frame number +/-.5 depending on the field settings.

• *plugin\_tex\_doit prototype* 

The plugin\_tex\_doit function should be prototyped for use by the getinfo function. You do not need to change this line.

• plugin\_tex\_getversion

This function must be in each plugin for it to be loaded correctly. You should not change this function.

• plugin\_but\_changed

This function is used to pass information about what buttons the user changes in the interface. Most plugins should not need to use this function, only when the interface allows the user to alter some variable that forces the plugin to do recalculation (a random hash table for example).

plugin\_init

If needed plugins may use this function to initialize internal data. NOTE: This init function can be called multiple times if the same plugin texture is copied. Do not init global data specific to a single instance of a plugin in this function.

plugin\_getinfo

This function is used to communicate information to Blender. You should never need to change it.

plugin\_tex\_doit

The doit function is responsible for returning information about the requested pixel to Blender.

The Arguments

• int stype

This is the number of the selected subtype, see the *NR\_TYPES* and *char stypes* entries above.

Cast \*cast

The Cast structure which contains the plugin data, see the Cast entry above.

• float \*texvec

This is a pointer to 3 floats, which are the texture coordinates for which a texture value is to be returned.

• float \*dxt float \*dyt

If these pointers are non-NULL they point to two vectors (two arrays of three floats) that define the size of the requested texture value in pixel space. They are only non-NULL when OSA is on, and are used to calculate proper anti aliasing.

The doit function should fill in the result array and return 0, 1, 2 or 3 depending on what values have been filled in. The doit function should *always* fill in an intensity value. If the function fills in a color value it should return 1, if it fills in a normal value it should return 2, if it fills in everything it should return 3.

#### Texture/Material Interaction

Blender is somewhat different from most 3D packages in the logical separation between textures and materials. In Blender textures are objects that return certain values, signal generators in fact. Materials control the mapping of textures onto objects, what is affected, how much, in what way, etc. Properly designed plugins should only include variables to affect the signal returned not the mapping of it. Buttons to control scale, range, axis, etc. are best only included when they make the texture easier to use (in the case of the size button in the Tiles plugin) or they speed up the calculation (the Intensity/Color/Bump subtypes in the Clouds2 plugin). Otherwise the Material Buttons make these buttons redundant, and the interface becomes needlessly complex.

### **Generic Texture Plugin:**

```
Relevant to Blender v2.31
#include "plugin.h"
/* Texture name */
char name[24] = "";
#define NR_TYPES 3
char stnames[NR_TYPES][16] = {"Intens", "Color", "Bump"};
/* Structure for buttons,
 * butcode name default min max 0
 */
VarStruct varstr[]= {
 {NUM | FLO, "Const 1", 1.7, -1.0, 1.0, ""},
};
typedef struct Cast {
 float a;
} Cast;
float result[8];
float cfra;
int plugin_tex_doit(int, Cast*, float*, float*, float*);
/* Fixed Functions */
int plugin_tex_getversion(void) {
 return B_PLUGIN_VERSION;
}
void plugin_but_changed(int but) { }
void plugin_init(void) { }
void plugin getinfo(PluginInfo *info) {
 info->name= name;
```

```
info->stypes= NR_TYPES;
 info->nvars= sizeof(varstr)/sizeof(VarStruct);
 info->snames= stnames[0];
 info->result= result;
 info->cfra= &cfra;
 info->varstr= varstr;
 info->init= plugin_init;
 info->tex_doit= (TexDoit) plugin_tex_doit;
 info->callback= plugin but changed;
}
int plugin_tex_doit(int stype, Cast *cast, float *texvec, float *dxt,
float *dyt) {
 if (stype == 1) {
 return 1;
 } if (stype == 2) {
 return 2;
 return 0;
}
```

# **Our Modifications:**

Relevant to Blender v2.31

The first step is to come up with a game plan. What is this plugin going to do, how are the users going to interact with it. For this example we will create a simple texture that creates a simple brick/block pattern.

Now we'll copy our generic plugin to cube.c and will fill in the gaps.

Its always a good idea to add some comments. First off tell users what the plugin does, where they can get a copy, who they should contact for bugs/improvements, and any licensing restrictions on the code. When using comments make sure you use /\* \*/ style comments. The plugins are in C and some cCcompilers do not accept // style comments.

```
/*
  Description: This plugin is a sample texture plugin that creates a simple
  brick/block pattern with it.
  It takes two values a brick size, and a mortar size.
  The brick size is the size of each brick.
  The mortar size is the mortar size in between bricks.
  Author: Kent Mein (mein@cs.umn.edu)
  Website: http://www.cs.umn.edu/~mein/blender/plugins
  Licensing: Public Domain
  Last Modified: Tue Oct 21 05:57:13 CDT 2003
*/
```

Next we need to fill in the Name, you should really keep this the same as your .c file, preferably descriptive, less than 23 chars, no spaces, and all lowercase.

char name[24] = "cube.c";

We are going to keep this plugin simple, and only have one type that deals with intensity. So we need the following:

#define NR\_TYPES 1

```
char stnames[NR_TYPES][16]= {"Default"};
```

For our user interface we are going to allow people to change; The size of the brick and mortar, as well as the intensity values returned for the brick and mortar. For that we need to edit the varstr and Cast. The Cast should have a variable for each entry in varstr.

```
/* Structure for buttons,
   butcode
                 name
                                default min max Tool tip
 * /
VarStruct varstr[]= {
   {NUM|FLO, "Brick",
                                           1.0, "Size of Cell"},
                             .8,
                                    0.1,
   NUM FLO,
               "Mortar",
                                    0.0,
                                           0.4, "Size of boarder in cell"},
                             .1,
                                           1.0, "Color of Brick"},
   {NUM|FLO,
               "Brick Int",
                            1,
                                    0.0,
                                           1.0, "Color of Mortar"},
               "Mortar Int", 0,
   {NUM FLO,
                                    0.0,
};
typedef struct Cast {
        float brick,mortar, bricki, mortari;
} Cast;
```

Now we need to fill in plugin\_tex\_doit, we basically want to break down our texture into "cells" which will consist of a brick and the mortar along the bottom edges of that brick. Then determine if we are in the brick or the mortar. The following code should do that.

```
int plugin tex doit(int stype, Cast *cast, float *texvec, float *dxt,
   float *dyt) {
   int c[3];
  float pos[3], cube;
   /* setup the size of our cell */
  cube = cast->brick + cast->mortar;
   /* we need to do is determine where we are inside of the current brick. */
  c[0] = (int)(texvec[0] / cube);
  c[1] = (int)(texvec[1] / cube);
  c[2] = (int)(texvec[2] / cube);
  pos[0] = ABS(texvec[0] - (c[0] * cube));
  pos[1] = ABS(texvec[1] - (c[1] * cube));
  pos[2] = ABS(texvec[2] - (c[2] * cube));
   /* Figure out if we are in a mortar position within the brick or not. */
  if ((pos[0] <= cast->mortar) || (pos[1] <= cast->mortar) ||
       (pos[2] <= cast->mortar))
      result[0] = cast->mortari;
   } else {
      result[0] = cast->bricki;
   }
  return 0;
}
```

One thing to note, the ABS function is defined in a header in plugins/include. There are some other common functions there as well be sure to take a look at what's there.

## **Compiling:**

Relevant to Blender v2.31

bmake is a simple utility (shell script) to aid in the compilation and development of plugins, and can be found in the plugins/ sub-directory of the Blender installation directory. It is invoked by: bmake (plugin\_name.c) and will attempt to link the proper libraries and compile the specified C file properly for your system. If you are trying to develop plugins on a windows machine bmake may not work for you in that case you should look into using lcc. You can use the following to compile a plugin with lcc: Assuming you have your plugins in c:\blender\plugins. Here is an example of how you would compile the texture plugin sinus.c Open a dos prompt and do the following:

(Note: You'll want to make sure the lcc\bin directory is in your path)

```
cd c:\blender\plugins\texture\sinus
lcc -Ic:\blender\plugins\include sinus.c
lcclnk -DLL sinus.obj c:\blender\plugins\include\tex.def
implib sinus.dll
```

# Writing a Sequence Plugin

Relevant to Blender v2.31

In this Section we will write a basic sequence plugin and then go through the steps use a sequence plugin. The basics behind a sequence plugin are you are given some inputs; 1-3 input image buffers as well as some other information and you output a resulting image buffer.

All the files necessary to develop plugins as well as a few sample plugins can be found in the blender/plugins directory. You can alternately get a bunch of plugins from http://www.cs.umn.edu/~mein/blender/plugins<sup>2</sup>

### **Specification:**

Relevant to Blender v2.31

• #include <plugin.h>

Every Blender plugin should include this header file, which contains all of the structures and defines needed to properly work with Blender.

• char name[]="Blur";

A character string containing the plugin name, this value will be displayed for the texture's title in the Texture Buttons window.

• VarStruct varstr[]= {...};

The varstr contains all of the information Blender needs to display buttons for a plugin. Buttons for plugins can be numerical for input data, or text for comments and other information. Plugins are limited to a maximum of 32 variables.

Each VarStruct entry consists of a type, name, range information, and a tool tip.

*The type* defines the data type for each button entry, and the way to display the button. For number buttons this value should be a combination (ORed) of INT or FLO for the number format, and NUM, NUMSLI, or TOG, for the button type. Text buttons should have a type of LABEL.

*The name* is what will be displayed on (or beside) the button. This is limited to 15 characters.

*The range information* consists of three floats that define the default, minimum, and maximum values for the button. For TOG buttons the minimum is set in the pressed state, and the maximum is set in the depressed state.

*The tip* is a string that will be displayed when the mouse is over this button (if the user has tool tips on). This has a limit of 80 characters, and should be set to the NULL string ("") if unused.

• typedef struct Cast {...};

The cast structure is used in calling the doit function, and serves as a way to simply access each plugin's data values.

The cast should contain, in order, an integer or float for every button defined in the varstr, including text buttons. Typically these should have the same name as the button for simple reference.

• float cfra

The cfra value is set by Blender to the current from before every render pass. This value is an the frame number +/-.5 depending on the field settings.

plugin\_seq\_doit prototype

The plugin\_seq\_doit function should be prototyped for use by the getinfo function. You do not need to change this line.

plugin\_seq\_getversion

This function must be in each plugin for it to be loaded correctly. You should not change this function.

plugin\_but\_changed

This function is used to pass information about what buttons the user changes in the interface. Most plugins should not need to use this function, only when the interface allows the user to alter some variable that forces the plugin to do recalculation (a random hash table for example).

plugin\_init

If needed plugins may use this function to initialize internal data. NOTE: This init function can be called multiple times if the same plugin sequence is copied. Do not init global data specific to a single instance of a plugin in this function.

plugin\_getinfo

This function is used to communicate information to Blender. You should never need to change it.

plugin\_seq\_doit

The sequence doit function is responsible for applying the plugin's effect and copying the final data into the out buffer.

The Arguments

Cast \*cast

The Cast structure which contains the plugin data, see the Cast entry above.

• float facf0

The value of the plugin's IPO curve for the first field offset. If the user hasn't made an IPO curve this ranges between 0 and 1 for the duration of the plugin.

• float facf1

The value of the plugin's IPO curve for the second field offset. If the user hasn't made an IPO curve this ranges between 0 and 1 for the duration of the plugin.

• *int x int y* 

The width and height of the image buffers, respectively.

• Imbuf \*ibuf1

A pointer to the first image buffer the plugin is linked to. This will always be a valid image buffer.

• Imbuf \*ibuf2

A pointer to the second image buffer the plugin is linked to. Plugins using this buffer should check for a NULL buffer, as the user may not have attached the plugin to two buffers.

• Imbuf \*out

The image buffer for the plugin's output.

• Imbuf \*use

A pointer to the third image buffer the plugin is linked to. Plugins using this buffer should check for a NULL buffer, as the user may not have attached the plugin to three buffers.

#### ImBuf image structure

The ImBuf structure always contains 32 bits ABGR pixel data.

ImBuf structs are always equal in size, indicated by the passed *x* and *y* value.

#### User Interaction

There is no way for Blender to know how many inputs a plugin expects, so it is possible for a user to attach only one input to a plugin that expects two. For this reason it is important to always check the buffers your plugin uses to make sure they are all valid. Sequence plugins should also include a text label describing the number of inputs required in the buttons interface.

### **Generic Sequence Plugin:**

```
#include "plugin.h"
char name[24] = "";
/* structure for buttons,
 * butcode name default min max 0
 */
VarStruct varstr[]= {
 { LABEL, "In: X strips", 0.0, 0.0, 0.0, ""},
};
/* The cast struct is for input in the main doit function
Varstr and Cast must have the same variables in the same order */
typedef struct Cast {
 int dummy; /* because of the 'label' button */
} Cast;
/* cfra: the current frame */
float cfra;
void plugin_seq_doit(Cast *, float, float, int, int,
ImBuf *, ImBuf *, ImBuf *, ImBuf *);
int plugin_seq_getversion(void) {
return B_PLUGIN_VERSION;
}
void plugin_but_changed(int but) {
}
void plugin_init() {
}
void plugin_getinfo(PluginInfo *info) {
info->name= name;
 info->nvars= sizeof(varstr)/sizeof(VarStruct);
 info->cfra= &cfra;
 info->varstr= varstr;
 info->init= plugin_init;
 info->seq_doit= (SeqDoit) plugin_seq_doit;
 info->callback= plugin_but_changed;
}
void plugin_seq_doit(Cast *cast, float facf0, float facf1, int xo, int yo,
 ImBuf *ibuf1, ImBuf *ibuf2, ImBuf *outbuf, ImBuf *use) {
 char *in1= (char *)ibuf1->rect;
 char *out=(char *)outbuf->rect;
}
```

#### **Our Modifications:**

The first step is to come up with a game plan. What is this plugin going to do, how are the users going to interact with it. For this example we will create a simple filter that will have a slider for intensity from 0-255. If any of the R,G, or B components of a pixel in the source image are less then our chosen intensity, it will return black and alpha, otherwise it will return whatever is in the image. Now we'll copy our generic plugin to simpfilt.c and will fill in the gaps.

Its always a good idea to add some comments. First off tell users what the plugin does, where they can get a copy, who they should contact for bugs/improvments, and any licensing restrictions on the code. When using comments make sure you use /\*\*/ style comments. The plugins are in c and some c compilers do not accept // style comments.

```
/*
Description: This plugin is a sample sequence plugin that filters out lower
intensity pixels. I works on one strip as input.
Author: Kent Mein (mein@cs.umn.edu)
Website: http://www.cs.umn.edu/~mein/blender/plugins
Licensing: Public Domain
Last Modified: Sun Sep 7 23:41:35 CDT 2003
*/
```

Next we need to fill in the Name, you should really keep this the same as your .c file. Preferably descriptive, less than 23 chars, no spaces, and all lowercase.

char name[24] = "simpfilt.c";

The Cast and varstr need to be in sync. We want one slider so we'll do the following:

```
varStruct varstr[]= {
    { LABEL, "In: 1 strips", 0.0, 0.0, 0.0, ""},
    { NUM|INT, "Intensity", 10.0, 0.0, 255.0, "Our threshold value"},
};
typedef struct Cast {
        int dummy; /* because of the 'label' but-
ton */
int intensity;
} Cast;
```

Now we need to fill in plugin\_seq\_doit. We basically want to loop through each pixel and if RGB are all less than intensity set the output pixel to: 0,0,0,255 else set it to the input values for that position.

Chapter 26. Blender's Plugins System

So we wind up with simpfilt.c

### **Compiling:**

bmake is a simple utility (shell script) to aid in the compilation and development of plugins, and can be found in the plugins/ sub-directory of the Blender installation directory. It is invoked by: bmake (plugin\_name.c) and will attempt to link the proper libraries and compile the specified C file properly for your system. If you are trying to develop plugins on a windows machine, bmake may not work for you. In that case you should look into using lcc. You can use the following to compile a plugin with lcc: Assuming you have your plugins in c:\blender\plugins. Here is an example of how you would compile the sequence plugin sweep.c Open a dos prompt and do the following: (Note: You'll want to make sure the lcc\bin directory is in your path).

```
cd c:\blender\plugins\sequence\sweep
lcc -Ic:\blender\plugins\include sweep.c
lcclnk -DLL sweep.obj c:\blender\plugins\include\seq.def
implib sweep.dll
```

### Notes

- 1. http://www.cs.umn.edu/~mein/blender/plugins
- 2. http://www.cs.umn.edu/%7Emein/blender/plugins

# Chapter 27. Yafray as an Integrated External Renderer

Relevant to Blender v2.34

by Gaurav Nawani

## Part 1

Yafray integration is one of the best features added to Blender. The current release of Blender 2.35 has a very neat integration with Yafray from within Blender, and is fairly stable to use. Unfortunately, Yafray usage is limited to those who are already comfortable with the Blender interface or can hack their way through. The first part of the tutorial deals with the basic steps needed to render from Yafray and later parts serve as a guide for the rest of the Yafray's available features.

### Interface

Blender has two of its own rendering engines built-in and that includes it's own raytracer and an older scan-line rendering engine. Yafray however is a standalone raytracer. Its functionality is accessed through Blender's interface by exporting the scene parameters to a Yafray readable format by one of two options. For the first one Blender has the required support built-in which allows Blender to use Yafray as a plug-in, virtually as if Yafray were an inbuilt renderer, and the other option is where Blender exports the scene data to Yafray format in an XML file called YBtest.xml, and then Yafray renders it as a standalone program. Both methods require Yafray to be first installed on the system and it is assumed that you have both Blender 2.34 and Yafray 0.07 installed on your system.

The Yafray integration in Blender can be broadly categorized in two parts. One relates to the interface for light or lamp settings. The other one to the core Yafray rendering features.

### Step-1

It is to be noted that this step might not be necessary for Windows. Before we proceed, we first need to configure Yafray preferences. Drag down the top menu bar to un-hide Blender's User Preferences window. Here click on File Paths button to open up the Path preferences menu. On the top left part is the YFexport text entry box. Enter the path where you would like to save the exported blend file in Yafray format (XML file) while rendering with the second option. This is necessary if you want to save the exported file and later edit it manually.

	Fonts: /	home/g	Textures: /home/gau							
ļ	Render	:			Ø	Pythor	n: /usr/lib/pyth			
[	V	iew &	Controls		Edit N	/lethods		Langua		
1:	✓ File	Add	Timeline	Game	Render	Help	SCR:2-Model			

Figure 27-1. Path selection in User Preferences menu.

### Step-2

As you are now aware, Blender allows you to choose between its own internal raytracer and Yafray. To use Yafray you first have to instruct Blender. To do that press F10 for Rendering Options window, now go to the *Render* tab in the Render Options window (Figure 27-2). In the Render tab, select Yafray from the Render engine drop down list (Figure 27-3), by default Blender internal is selected.

**Note:** Please do not get confused with the Ray button. This has nothing to do with enabling Yafray's ray-tracing. It does not effect in any way the Yafray renderings if Yafray is chosen as the raytracer from the rendering engine drop down list.



Figure 27-2. The default options in Render Tab.

Notice as soon as you choose Yafray two more tabs appear beside the Render tab (Figure 27-3). These two extra tabs are Yafray and Yafray GI and these two tabs have rendering and other features and parameters of Yafray.

•	Rend	ler		YafRay	YafRay GI				
		REI	Shadow EnvMa				ар		
YafRa	у		Pano Ray Radi			dio			
	0	SA	100%						
5	8	∜ Bf: 0.50 ▶	75%	50% 25%			5%		
<b>•</b>	(parts:	1 🕨	Fields		0	Odd X			
			Gauss		4 1.00 ▶		) ⊳		
Sky	Pre	mul	Border G			amma			

Figure 27-3. Yafray raytracer selection and its two sub-tabs.

Now select the Yafray tab from the two new tabs, there will be some functions visible, right now we are interested in the button named XML (pressed by default) (Figure 27-4). You go ahead and turn it off right now. I will explain why, as I had told earlier that Blender has two options for rendering with Yafray, either as plug-in interface or to call Yafray as command line program.



Figure 27-4. Default Yafray tab with Yafray file export enabled. (XML button).



Figure 27-5. Disabled Yafray export.

Choosing the plug-in or first option (XML button off) allows you to see the rendering progress in the Render window (Figure 27-6), much like the Blender's own rendering, and this is one of the reasons why Yafray requires and uses more memory than before. Using the second option which is by default (XML button pressed) active, will first export the active scene to the YFexport path, which you have set in your File Paths preferences (the Section called *Step-1*). The Yafray is then called as a command line program, and it takes over the processing of the exported XML file, and only after completion provides the image back to the Render window, so there is no other interaction in between, save for the textual output in the Blender's terminal window.



Figure 27-6. The rendering in progress for Yafray XML disabled.

### Step-3

After going through the first two steps you have virtually done every thing to make a Yafray render. At this time you have your scene ready with lamps of your choice, the best thing to proceed from here is to press **F12** to render. Here is where the *problem* crops up for almost everyone. I will explain... Depending on the light settings in your scene, you either will see a blank screen or will see very faint outlines of the objects in the scene. Or if your lamps have higher light intensity (value) then you might see the scene properly. Alternatively, in extreme case, Yafray and Blender do a crash thingy.

In case you came up with black render the chances are good that you can render through Yafray, all you need now is to adjust the light's parameters, and you will have your own Yafray render within minutes.

#### Points to check for problems

- The first problem for Yafray renders is almost every time a lack of sufficient light intensity in the scene. This is not actually a problem of Yafray, but an implementation issue from within Blender. So, in this case, you need to increase the light intensity, or value, of every lamp in the scene (more information in the Section called *Part 2*).
- If every thing fails you might need to check for the distance value of the lamps, since the light attenuation falloff is mostly sharper in CG (to reduce computations), your objects in the scene might appear black because their ray casting distance just might not be reaching the objects in the scene. You can get around this by adjusting distance value in the Lamps tab.
- In case of crash please check the elysiun forums for possible answers.

### Part 2

Instead of instructing you to modify the settings here and there, I thought it might be better to explain the Yafray light types one by one. This will enable you to make informed decisions.

#### Let there be light... And there was Yafray everywhere

There are five light types used in Blender and of them only four are directly supported in Yafray. On enabling Yafray raytracer from the Render tab, you initially see six lamp types (in the lamp options window, F5), of which Yafray makes use of Blender's lamp, area, spot and sun. I will come later to the other two lamps listed, Hemi and Photons later.


Figure 27-7. The scene.

The scene description: The current scene has two light sources. One is a helper light (Sun, without shadow and at value .200), which is used to provide non-directional light to increase the ambiance for the scene. The other is the lamp source (or light source), which is a placeholder for every light type explained. The lamp positions are static through out, and Ray Shadow (shadow casting) is enabled for all the lamp sources that are being used (see Figure 27-8).



Figure 27-8. The scene as seen in the Viewport.

**Note:** The suggested uses for the light types defined here are general in the usability approach. They are not to be taken as the absolute word for lighting your scene. The lighting varies depending on the scene and the artists own vision for the scene. For best results experiment with the lights in your scene to find which works best for you.

**Important:** The mention of render times in the images is rounded off, it is relative and is used just to give the idea of typical render times for same scene by different lights and settings.

## Lamp

This is the simplest light source available, in some ways it is like sun, as it is omnidirectional (i.e. shoots light in all directions from the point of its origin) and is spherical in nature. Since it shoots light rays in every direction, that means the ray going away from the scene will not make any visible changes in the scene, the light rays will be lost and thus are waste of computation, Although Ray tracers use a variety of hacks by cutting off the unwanted computations, it is a bit slower than other directional light sources if bigger parts of the light shot come inside the scene view.

The general use of Lamp is for indoor lighting, like rooms and halls, but it is not limited to this use, it is also more commonly used as *filler light* or *helper light* in the scene.

*Values:* When a single lamp is used for the scene in Yafray (through-out the tutorial when I say Yafray I mean Blender-Yafray), it will generally not provide sufficient lighting to light up the scene unless it is kept at a higher value. For the sample scene, the value of Lamp was set at 10 (maximum) but still there seems to be the need for additional lighting (but we are sticking to it for the sake of understanding). See Figure 27-9.



Figure 27-9. Rendered with Lamp at default settings.

The Lamp will cast sharper shadows at default (zero) radius. The radius setting in Lamp is used to increase the area size for shooting the light. If the size of Lamp is bigger than the objects, then the light shot from some parts of the lamp could directly reach to the parts of the scene where the other part of the lamp casts a shadow. This

intersection results in the shadow being diluted for that part, thus getting blurred (also called *partial shadows*). You can observe this phenomenon in real world (sun light).

*LINK:* for more info on *shadow dilution* or Partial shadows go to Ditto head's Light tutorial<sup>1</sup>.



Figure 27-10. The noise comes when sampling values is low.

The scene above (Figure 27-10) was rendered at radius 2 with a sampling value at 1, as can be seen the scene has grainy partial shadows. The use of samples button is only to reduce those grains in shadows, the sampling buttons control the number of samples used in shadow calculations, increasing the sampling results in smoother shadows. The sampling button is only available in lamp and area light and functions the same for both.



Figure 27-11. The increase in sampling to 5 removed the noise.

### Area

The Area Light is a directional source. The shape of Area Light can be varied from square to rectangular from the drop down list in the Lamp tab. Below the shape selection there is also an option to increase the size of the Area Light.



Figure 27-12. The basic render from area light at default setting and value of 4.

*Values:* For Area Lamp you will need lower light intensity values for rendering a scene with Yafray. The sample scene is lit by light value at 4, it is brighter than the one rendered by lamp at value 10. This is because it is shooting all the light towards a direction from a plane while the lamp shoots the light in every direction distributing energy where it may not be required.



Figure 27-13. Although the scene looks similar to Figure 27-11 but is brighter and have better shadows.

**Note:** One important point to remember is that in the official Build of Blender2.34 using two or more area lights results in error in the renderings, it is a recognized bug and have been patched in some development builds.

### Spot

The spot is also a directional light source and as the name suggests it provides a circular spot of light or more appropriately a cone of light. The size of cone can be controlled by the SpotSi or angle of spotlight beam. The higher the angle the more nearer it behaves like an area light, but with one major difference. It cannot cast partial shadows. Spot light has a parameter to control the SpotBi or Spot's edge smoothness only. It can be seen only if the spot light is inside the scene view, otherwise you will not notice any difference in spot or other lights.



Figure 27-14. The Spot lamp render with default settings.



Figure 27-15. Spot lamp with spherical light attenuation.

Notice the increased brightness at the portion nearer to the lamp in the Figure 27-15 while using the Spherical light attenuation (quad is not supported in Yafray). Increase in the distance value the spherical attenuation also increases in radius, for example the Figure 27-15 has a default distance value of 20 while Figure 27-16 has the distance value at 30. The effect looks like that of typical household lamps where the attenuation is sharper.

**Note:** The light attenuation in Yafray has only quadratic falloff, while Blender can have linear, cubic and the mix of these also. That explains relatively sharper light intensity falloff in Yafray renders.



Figure 27-16. Spot lamp with Spherical attenuation enabled.

## Sun

This is also a directional light source. It tries to emulate sunlight by shooting light of the same intensity everywhere in the scene without attenuation (Figure 27-17). This results in the environment being lit up with constant ambiance. Its value must be kept lower. The sample scene uses a light intensity value of .100. It is obviously good for out door lights, especially for a sun.

In outdoor scenes, if you do not want sharper shadows you can disable shadow casting for sun, and use other lamps for shadow generation. But make sure they have sufficient light intensity to cast a shadow.



Figure 27-17. The sunlight at value 2. No wonder why this lamp is called Sun.

### More lamps?

*Hemi* - Yafray does not support Blenders implementation of Hemi light internally; right now it just uses Yafray's implementation of sun light instead, so you can use Sun light instead.

*Photon* - Last of all, the Photon light source or photons lamp button is not to be confused with any light source. It does not cast any shadow or light, it is only used in Caustics\* calculations and requires being placed or directed where you need the caustic calculation. Photon lamp shoots photons in an area, the photons are used by Yafray as the specialized ray elements only to calculate the caustics on passing through the objects like glass or mirrors, which have have the property of bending light when passed through them, known as *total internal reflection*. Basically the placement of photon lamp is to allow the user to optimize the rendering. Caustics is one of the most computation intensive jobs in Ray tracing.

**Note:** This photon lamp and photons are not to be confused with the photon option in the GI method. Which is explained in the the Section called *Part 3*.

# Part 3

## GI and other features

Yafray supports *Global Illumination*\*. We have to go back to the Render tab to learn more about Yafray's implementation of GI. The first look at the Yafray GI tab and you might think. Oh! It is so easy, you are right the features available are simplified for your use. The available options are *Method* and *Quality*, lets have a look at them one by one. The quality is the same for both and is explained later.



Figure 27-18. The Yafray GI tab.

# The available Methods of GI

## Skydome

While Skydome is simply a method to have lighting from the sky or more appropriately the atmosphere, it does not provides full GI in the true sense, as it does not takes into account the indirect light bounces on the surfaces of objects. Rather it affects the light in the scene by the colour of the atmosphere and also the diffusion of light in the atmosphere is controlled by it. For example in the default scene the colour of atmosphere was set at reddish (chosen just to clearly identify the affect). See Figure 27-19 for how the diffusion effects the scene.



Figure 27-19. The left part of image have the power of diffusion set at 2 while the right one at 4.

The colour of the background or atmosphere can be changed in the Word Panel (F8). You have to enable the world for Skydome to work fully. The Skydome offers no other features. The Skydome is faster method of GI, while its results may not be physically more accurate, its results are not too bad to ignore for the speed advantage it offers.

### Full

On the other hand, the Full option takes into consideration the reflected and/or refracted light bounces (or indirect light) on neighboring surfaces. This method is a very close simulation of actual lighting in the real world, and that is why the GI produces more photo-realistic images than any other method used in CG, that is also why it requires higher computations and is slower.



Figure 27-20. Using Full GI increases the time as well as realism.

Notice the difference in realism between the two parts in Figure 27-20. The light bounces provide enough diffusion in the second part of the image, which lights up the shadow portion in the scene according to the distance of the surfaces in contact. While the first part has approximately similar shadow depth through out.

Reno	ler	YafRay	YafRay GI
	Full	\$	◀ Depth: 3 🕨
	None	\$	CDepth: 1
	Cache		Photons
- Sha	adQu: 0.940	÷	✓ Count: 200000 ►
Prec: 1	Gradie	nt	Radius:1.00
Refinement: 0.200			- MixCount: 100
▲ Power: 2.00 ▶			Tune Photons

Figure 27-21. The Yafray GI tab. Options for full method are visible.

# Features for Full method of GI

## Depth:

This refers to the total number of light bounces for one reflected or diffuse ray. Generally the depth 3 works best for normal scenes. Increasing the depth results in higher amounts of calculation per ray.

## Cdepth:

The Cdepth is nothing but the bounce depth for caustics, or for the transparent materials like the glass, gems, liquids etc. Higher computations are required for caustics calculations, that is why the GI in CG keeps separate the normal lights with photons, even though in the real world the photons are the actual light particles. For better quality of caustics you will need to have as much as 3-5 Cdepth or more for detail.

## Photon:

This photon button should not be confused with the photon lamp. This is only used as a helper in global illumination for Yafray and bears no relation with the caustic photon lamps or caustic photons.

## Count:

The count refers to the total number of photons to be made available in the scene for helper in GI. The number of photons will will vary for the scene, but higher numbers in the scene will provide smoother results, again its at your disposal to find time/quality limit.

## Radius:

This refers to the distance within which the photons calculated have higher precision value to effect the GI, outside which the photons do not effect the scene much. So the Idea is to optimize the radius to the size of the area where better GI calculation is required. Keep the radius half the size of area.

## MixCount:

This allows you to choose the number of photons which should be kept inside the Radius. The Radius and Mix Count make up your *photon map*. The photon-map is nothing but the optimized area for photons used in GI calculation and Yafray sufficiently processes the sampling and photon gathering on the photons inside the photon-map and leaves the photons outside the photon-map.

## **Tune Photons:**

This allows the re-use of successful gathering of photons and positions from cache, which helped in GI calculation on previous render, and only re-compute the other photons for faster result. It is may not necessarily faster in every rendering but can provide good speed improvements generally. Due to speed improvement it is good to use it during test renders once you come close to what you wanted. You can disable it after you are satisfied with the test renders and want to create the final version of the image to do the full computation again.

#### Quality:

The quality dropdown list allows you to choose the quality level of GI for both the Skydome and Full GI methods. Quality setting allows control over the number of samples used to sample in GI. Since its use is automatic by default, it will try to used maximum value for MultiPasses and samples per pass depending upon the quality level. If you want, you can disable the AutomaticAA manually and can instruct Yafray accordingly.

## Other features of Yafray

#### Anti-Aliasing:

The Yafray tab in Rendering options window lists the options for manual control over anti-aliasing. Press the AutoAA button to un-hide the manual settings.

Render	YafRa	у	YafRay GI	
Auto AA			xml	
AA Passes 0	⊩	4	AA Samples 0	Þ
Psz 1.500		Thr 0.0	50	
Raydepth 5	Þ			
Bi 0.001		4	Processors: 1	Þ
Gam 1.000		Exp 0.0	00	

Figure 27-22. The Yafray tab.

- *AAPasses:* It is used for the number of passes to be used for Anti-Aliasing. The higher the passes, the better the results, longer render times also. While using passes of more than one, every pass samples the total number of samples chosen in AASamples for every pass, thereby achieving better AA of pixels in noisier areas. If the results are noisier in a single pass you can increase the number of passes for better control and also a higher number of samples per pass. Single pass antialiasing with higher sampling, is not necessarily better than lesser samples with multipass.
- *AA Samples:* The number of samples per pass. The total AA sampling can be calculated by multiplying AApasses with AAsamples. In our case for the the final blurred image for the DoF used below, we have used 4x4=16 anti-aliasing samples.
- *Psz:* The psz is the *>Pixel filter size>*, that is used during the Aliasing calculations for aliasing the overlapping of neighbouring pixels.
- *Threshold:* This sets the threshold or the maximum brightness difference with the neighbouring pixel, when this is above the chosen level, extra samples are taken until the result is below the threshold limit or maximum samples/passes are reached. Lower threshold means more pixels will be antialiased, so at 0 all pixels in the render will be anti-aliased, when 1 no anti-aliasing is done.

*Raydepth:* This effects the maximum number of *bounces* a reflected/refracted ray can make. This is only important for glass and mirrors, higher ray depth will improve the quality of the reflection/refraction in the glasses and or mirror.

## The Depth of filter

Yafray supports the real camera like DoF. The DoF of filer setting can be accessed easily. First select camera in view port then press F9. You will see a tab along with the Camera tab named Yafray DoF. In the Camera tab enable Show limits, doing this will allow you to see a yellow cross on the Show limits line of the camera in Viewport. The yellow cross is what we need to put at or near the objects we want to have sharp focus on. That distance can be set by the DoFDist numeric button. The Aperture setting is the value you need to change to get the required DoF you want. Normally a setting in between .100 to .500 will suffice for most scenes.



Figure 27-23. Notice the noise on the wooden boards.

How to remove the artifacts in DoF rendered scenes. DoF filter requires more samplings per scene to get the right amount of blurring. To do that you need to disable Automatic AA in Yafray tab in Render panel. Here try to adjust the AAPasses and AA Sample accordingly. If you have large aperture settings then you need to have multiple passes plus higher samplings per pass. Increasing anti-aliasing will also increase the render times.



Figure 27-24. Increasing the sampling manually, and aperture size solves the problem.

## HDRI or High Dynamic Range Illumination

Yafray has HDRI support. To use this go to World buttons F8. Go to Texture and Input tab and add a new texture. Then go to texture selection F6 select Texture type as image, press Load image button, locate the HDR file, select and press Enter. Blender will not show the HDRI image in the World tab, however it is loaded automatically during rendering. You can also increase or reduce the exposure of the HDRI from the Texture brighteners button in the colour tab in texture window (F6). The possible exposure settings are -1,0,1, for brightness' sliders at 0,1,2. This is because HDRI exposure can be modified in integer values only. For the final step, use any one of the GI methods. Generally Skydome will work fine.



Figure 27-25. HDRI render using Uffizi probe. It need more quality level.

## **Closing comments**

I would like to thank eeshlo, to an large extent I was motivated to learn about Yafray because of him, and thus was able to write this tutorial, and also for his help in solving my queries. Thanks also goes to Dreamsgate for helping in the editing and support for this document.

This tutorial is not finished, as there are several aspects which I have not tried myself, like caustics, and GI techniques for certain types of renderings. So I call this tutorial version 1. Later improvements will not necessarily come at regular intervals. If you have any queries or have some point that I overlooked or misunderstood in the matter written above. I would be glad to hear from you. Also welcome are the suggestions for improvements. My mail-id

# **Glossary for the geeks**

*Global Illumination:* It is a method (algorithm) of computation for light calculation in the scene which, takes in to account the light bounces from the neighboring surfaces, along with the normal illumination of direct lights. In Other words GI calculates the Indirect light also, thus it makes the renders more photo-realistic. Examples of GI

methods are Radiosity and Ambient Occlusion in Blender and on a general scale Radiosity, Ray tracing and Caustics all use different GI algorithms.

*Ray tracing:* A method in CG which uses an algorithm to calculate the effect of lights on the surface of objects in the scene. In CG the ray-tracer. works by calculating the light effect on the scene by *tracing* the light photons back to the point of origin, from the scene or the camera. It uses the reverse of what is in real life, the sun shoots photons and we receive them through reflection/refraction from the objects, the photon energy is also modified by the objects by absorption or adsorption to form a particular texture or colour of the object.

The reverse way of ray-tracing is done so as to reduce the amount of calculations, as it is faster to take just the photons or lights which reach the scene or our eye or the camera, than calculating everything what is outside the view.

*Photon:* The photon is also referred as Ray of light. And it is the smallest unit of light energy.

*Caustics:* The caustics are referred to as the refraction pattern formed by highly transparent objects such as a glass of fluids which have a certain degree of Total internal reflection, for example the light falling on a glass filled with wine will form some strange patterns of different colours and intensity which are referred to as light caustic. The computer method for Caustic calculation is also referred to as photon mapping. The photon lamp in Blender/Yafray is for this purpose only.

*HDRI:* Or High Dynamic Range Illumination. This method is relatively new in GI. This uses the actual light probe value of a real scene in the real world taken through with special equipment to produce a 360 view of a scene, and stores the information of the light from all areas in the scene in a spherically mapped image called HDR. The renderer uses that information to shoot light from and provides even more photorealistic rendering.

*Anti aliasing:* It is refers to a method to reduce the brightness levels between two neighboring pixels by overlapping the colours in the difference level to neighboring pixels. This make the images appear smoother.

*Photonmap:* An assumed area or bound with-in which more density of photons are kept for calculations of indirect illumination.

# Glossary for the geeks

#### **Global Illumination**

It is a method (algorithm) of computation for light calculation in the scene which, takes in to account the light bounces from the neighboring surfaces, along with the normal illumination of direct lights. In Other words GI calculates the Indirect light also, thus it makes the renders more photo-realistic. Examples of GI methods are Radiosity and Ambient Occlusion in Blender and on a general scale Radiosity, Ray tracing and Caustics all use different GI algorithms.

#### **Ray tracing**

A method in CG which uses an algorithm to calculate the effect of lights on the surface of objects in the scene. In CG the ray-tracer. works by calculating the light effect on the scene by *tracing* the light photons back to the point of origin, from the scene or the camera. It uses the reverse of what is in real life, the sun shoots photons and we receive them through reflection/refraction from the objects, the photon energy is also modified by the objects by absorption or adsorption to form a particular texture or colour of the object.

The reverse way of ray-tracing is done so as to reduce the amount of calculations, as it is faster to take just the photons or lights which reach the scene or our eye or the camera, than calculating everything what is outside the view.

#### Photon

The photon is also referred as Ray of light. And it is the smallest unit of light energy.

#### Caustics

The caustics are referred to as the refraction pattern formed by highly transparent objects such as a glass of fluids which have a certain degree of Total internal reflection, for example the light falling on a glass filled with wine will form some strange patterns of different colours and intensity which are referred to as light caustic. The computer method for Caustic calculation is also referred to as photon mapping. The photon lamp in Blender/Yafray is for this purpose only.

#### HDRI

See: High Dynamic Range Illumination

#### **High Dynamic Range Illumination**

Or *HDRI*. This method is relatively new in GI. This uses the actual light probe value of a real scene in the real world taken through with special equipment to produce a 360 view of a scene, and stores the information of the light from all areas in the scene in a spherically mapped image called HDR. The renderer uses that information to shoot light from and provides even more photo-realistic rendering.

See Also: HDRI.

#### Anti aliasing

It is refers to a method to reduce the brightness levels between two neighboring pixels by overlapping the colours in the difference level to neighboring pixels. This make the images appear smoother.

#### Photonmap

An assumed area or bound with-in which more density of photons are kept for calculations of indirect illumination.

Glossary for the geeks

# Notes

1. http://www.blenderman.org/modules.php?name=Content&pa=showpage&pid=33

Chapter 27. Yafray as an Integrated External Renderer

# Chapter 28. From Blender to YafRay Using YableX

Relevant to Blender v2.31

by Manuel Bastioni

# What is Yable?

Yable is a Python script, originally devised by Andrea Carbone, allowing to export the Blender scene in the XML format of YafRay, so to be able to exploit that engine for highly photorealistic renderings. However, Yable is not a mere "format converter", but is a true scene processing lab that allows you to assign and change the lights, materials and environmental settings taking full advantage of YafRay features. From another point of view we could consider Yable script as a GUI, able to visualize and manage with great simplicity the large quantity of parameters used by YafRay.

## Which Yable?

The first versions of Yable (end of 2002) have been realised entirely by Andrea and, afterwards, as a result of the success of the script among the users, many different versions of the script have been issued, to correct bugs and add new features. It is important to point out the contribution of Alejandro Conty Estévez, Alfredo "Eeshlo" de Greef, Christoffer Green, Leope, Johnny "guitargeek" Matthews, Jean-Michel "jms" Soler.

After the official release of Yable 0.30 many non-official patches has been issued, generically called YableX, and published on the Yable forum on www.Kino3d.com. For the purpose of this Chapter we have examined all those versions to come up with a new "official" version and decided to base it on the latest YableX release that, thanks to modifications carry out by Jms, works with the last release of Blender.

## Where to get YableX?

The two main Blender sites, which you should know by now, have links to it, anyway the last version has been realized by Jms and can be downloaded from his site Zoo-Blender (http://www.zoo-logique.org/3D.Blender<sup>1</sup>). I recommend, however, that you have a look at the official YafRay forum, in the exporters section: http://www.YafRay.org<sup>2</sup>

# Installing the script

### Relevant to Blender v2.31

Yable is a script, that is a simple text file that can be loaded in Blender, hence we cannot talk properly of an installation. As a matter of fact it is sufficient to load it in Blender's Text Window and press **ALT-P** to launch it. Before doing so we must anyway pay attention to two fundamental points:

- You need a full Python installation (the right one for your Blender version), down-loadable at www.python.org;
- You must edit line 81 of the Yable script.

The first is a necessary condition so that the script could find the required Python modules, while the second is needed to set up the directory in which the settings and the XML generated from Yable will be saved. Setting, for example, line 81 to:

Implies that every time that you export a scene from a file foo.blend, a new folder called foo will be created in C:/ containing all the elements of the scene. In this way, even if you close Blender, and then you reopen the file and restart the script, Yable will be able to retrieve the settings, since it search them in a folder that it has the same name of the current .blend file.

**Note:** This automatic naming is very handy, but unfortunately implies also that if we want to save the .blend file with a different name, you must re-export it from Yable at least once, so that the settings will be written again. Otherwise you can copy the contents of the old folder into the new one.

If you want, you can also set up an external viewer that start automatically in order to show the result of the rendering: If you so desire you must edit line 90, setting the VIEWERAPP variable the path to the chosen application.

As said before, all the data will be saved in the directory defined as YABLEROOT, all *except* the textures. Is important that all the images used for the scene are in the same directory, and that such directory will be indicated correctly to Yable, but this is done at run time in the pertinent Text Button appearing as soon as the script is launched (Figure 28-1). The *full* path to the textures is required.

Texture dir:		
	Start YaBle	

Figure 28-1. Texture Dir setting and startup button.

# The Interface

Relevant to Blender v2.31

The functions of Yable have been divided in three main screens, accessible by pressing the three upper buttons that you can see in Figure 28-2.



Figure 28-2. Yable header Buttons.

## Workflow philosophy.

Once we have created a Blender scene, with objects, materials and lights, we can load and start the script, possibly splitting the main 3D Viewport in two, and turning one of the two halves into a Text Window. This way we will be able to see at the same time the scene and Yable GUI.

The Yable Workflow is as follows:

- Select an object of the scene;
- Go to the Material or Light part of the interface, depending on what you are defining, and press the Get Selected button. This way Yable retrieves the settings for the object (if any);

- Edit the attributes. These are completely independent from those of Blender!
- Press the Assign button to assign the parameters you entered to the object. Don't forget this step! A button Assign All can be used to assign the data to all selected objects.

# **Global Settings**

This part of the GUI allows us to access the functions of general scene settings Figure 28-3.

Global Settings Material Settings	Light Editor Exit
Texture dir:	
Hemi Light 6 Path Light	Const BkGd SunSky BkGd HDRI BkGd Normal BkGd
De <mark>pi<b>g</b>rField</mark>	Fog: 0.000 Red ( 8 en ) Blu R: 1.000
<b>10</b> °	No Default expos: 0.000 o Gamma: 1 Resolution 320x240
Lavers Anim	AntiAlias Settings
INC Export 1 Pename: untitled Render Image: untitled Time View Output alphaTGA Yab	AAPW 1.50 3AAT 0.040 Ray Depth 5 Cpu 1 Bias: 0.300 0 IndSamples le-0.30 Yet Another Elender Exporter

Figure 28-3. Yable headerGeneral Setting Buttons.

- *Texture path* (Figure 28-3#5) The path of the texture can be redefined anytime.
- *Global Illumination* (Figure 28-3#6) Adds to the scene the global illumination, that is the simulation of the diffused light, originated in nature as an effect of the infinite mutual reflections and diffusions between the objects. Its effect is added to that any possible direct lights.

Path light and Hemi light are finalized to yield of the same effect, but using different algorithms that have advantages and disadvantages, for the description of which we send back to the specific YafRay Chapter.



Figure 28-4. Yable Global Illumination settings.

Figure 28-4 shows the various options for Hemi or Path light. In the former case we can set up the colour using the Red, Green and Blue NumButton, or take the color from the background with the button Use Background color. It is possible to use this last feature when we use backgrounds based on images, even if the Use Background Button disappears by setting all the three RGB slider to 0.

In the Path Light case, on the other hand, the background image is used by default. Another difference between hemi and path light is the Depth parameter, that is referred to the number of bounces to be considered in the calculation of the exchange of reflections between the objects. To get a minimum of radiosity effect it is necessary to have at least two light interchanges.

Since the Path Light computation is rather complex a Cache option is provided, allowing to optimize and diminish the rendering times. It basically acts as a preprocess used to determine the zones of the image that need more samplings; as an example on a great flat surface we can presume that the diffuse lighting is quite uniform, and therefore we can carry out less calculations.

Parameters shared by both the global lights are the Power, the QMC, and the Samples. The Power indicates the power of the luminous emission, while the Samples indicates the accuracy of the sampling during the rendering: high values improve the clearness of the light (the hemi light have the tendency to become grainy) but this increase vary much the times required for calculation. QMC refers to the use of the Quasi MonteCarlo method for the determination of the zones to compute: it is based on sequences of quasi random numbers, and accelerates the rendering, even if, sometimes, it generate granular pattern of the image.

• *Background (Figure 28-3#7)* - There are four options. Depending on the choice made some additional buttons appear, almost all are of immediate understanding (Figure 28-5).

Const BkGd SunSky BkGd	Const BkGd SunSky BkGd
HDRI BkGd Normal BkGd	HDRI BkGd Normal BkGd
Red 0.800	Sun Set Sun Pos Turbid: 4.00
	BrightH: 1.00 SpreadH: 1.00 BSGIow: 1.00 SSGIow: 1.00 BackScattLight: 1.00
Const BkGd SunSky BkGd	Const BkGd SunSky BkGd
HDRI BkGd Normal BkGd	HDRI BkGd Normal BkGd
Exposure Adjust : 0	Power : 1.000
Probe Name: EMPTY	Image Name: EMPTY

Figure 28-5. Background settings.

The Const BkGd, is the easier to use (Figure 28-5 top left): it is an homogenous colour defined by its RGB values.

The Normal BkGd (Figure 28-5 bottom right) allows us to use an image (the last version of YafRay supports JPG and TGA); the only parameter is the Power, that indicates the brilliance of the image.

The HDRI BkGd (Figure 28-5 bottom left) is, perhaps, the one allowing for the maximum realism. The HDR (High Dynamic Range) Images, by storing pixel colours as floating point numbers contain much more data than other formats. Moreover, they are usually available as *probes*, that is as full 360 horizontal, 180 vertical backgrounds. After the we have obtained the appropriate images, is necessary to put them in the same folder as the textures, to write the name in the Probe Name button, and to set up the exposure that we want to use (positive means brighter).

Finally the SunSky BkGd (Figure 28-5 top right) uses a sophisticated algorithm for the simulation of the conditions of Sun light. The position of the sun can be set by selecting an object in the Blender scene (usually an empty) and pressing the Set Sun Pos button and confirming. It is important to note that the dimension of the sun will depend also from the distance between the chosen object and the camera. A particularly important parameter for the construction of the scene is Turbid, that allows us to regulate the value of the density of the atmospheric layers that envelop the planet: dense layers let to go through only determined wavelengths of the solar light, hence it changes both the color and the power of the light. The other buttons control the halo and the spread of the beams. The SunSky background, if used with Path Light or Hemi Light, is able to emit light in extremely realistic ways. In the case in which we don't want to use Global Illumination we can press the Sun button and Yable will add a Sun type light in the exported scene, that will simulate, more roughly but faster, the effect of the solar light.

• Fog (Figure 28-3#8) - With the Fog slider we choose the amount of fog present in the scene (zero by default), while the color is chosen by selecting one of the three Red, Green and Blue button and by using the single Num Button below them (Figure 28-6).

Fog: 0.000		
Red ][	Green	Blu
R: 1.000		

Figure 28-6. Fog Settings.

• *Depth of field* (Figure 28-3#9) - This is required to mimic a real camera's focal blur. This is a feature that YafRay will render in much shorter times than other rendering engines, as it is performed as a post-processing operation - however, this means that it has some shortcomings in the precise accounting of reflections.

The regulation is made by selecting an object whichever in the scene of Blender and pressing the button Set Focus (Figure 28-7). The point chosen in this way will be perfectly sharp. With the others two Num Buttons we can regulate the amplitude of the field depth: Near Blur influences on how much will the objects that are between the camera and the point of focus be blurred, while Far Blur affects the objects further than the focus from the camera.

Depth of Field
Set Focus
Near Blur: 10.06
(Far Blur: 10.00
(Scale: 2.00

Figure 28-7. Depth of Field Settings.

• *Anti-noise Filter* (Figure 28-3#10) - This too is a filter applied as post processing. It works in an iterative manner by taking some points within of a circular area and assign the same color if their colours differ more than a given threshold.

Anti Noise
(Radius: 1.00 -
Max Delta: 0.10

Figure 28-8. Anti Noise Filter Settings.

The amplitude of the circular area is determined by the Radius parameter (Figure 28-8), while the threshold is given by Max Delta. This is a very useful filter, but to use with care, because it has also a blurring effect that might compromise the quality of the result. Higher values of the delta tend to unify all.

• *Gamma correction, exposure, resolution* (Figure 28-3#11) - A simple group of buttons that allows you to set the brightness and the gamma of the complete rendering. The No Buttons exclude completely both post-processes. The Default Button that bring back the expos Num Button to the default value. The Gamma Num Button allows the regulation of the gamma (Figure 28-9).



#### Figure 28-9. Resolution Settings.

The resolution Menu Button allows you to choose the dimension of the rendered image. Pressing it, we can choose between the most common formats:320x240, 480x320, 640x480, 640x512, 768x470, 1024x576, 1024x768 and 1280x960. Choosing the Custom option, two new buttons are visualized, that allow to set up any resolution.

• *Rendering setting* (Figure 28-3#12) - This group of buttons allows you to set details of the exported file and let to launch YafRay directly from Blender (Figure 28-10).

Lave	rs	Anim				
Path	gz _	<u>Fr</u> <u>Update Meshes</u>				
IINIC	Export	Filename: untitled				
INC	Render	Image: untitled				
Time	View Output	alphaTGA				

Figure 28-10. Rendering Settings.

The fundamental keys are Export, Render, Filename and Image. The last two are needed in order to choose the name that will have the XML file and the rendered

image. Export produces only, within the YABLEROOT directory all the necessary XML files while, if used with Render, will also execute YafRay and will produce the final image; finally, if the button View Output is also selected, at the end of the rendering Yable will launch also the application specified in VIEWERAPP, in order to see the result. Note that you must have YafRay in your path to be able to launch it within Blender.

The Layers button open a new panel for the choice of the layer to export (Figure 28-11).

11	2	3 13	4 14	5 15	6 16	7 17	8 18	9 19	10 20
		All O	ff			A	All On		
Back									

Figure 28-11. Layer Selection.

The Path button forces the description of the scene to be exported using separate files (one main file, and sub-files to be saved in suitable folders, subdivided by materials and meshes). The location of such files will be indicated to YafRay by with the use of a full path.

To the contrary, the !INC Button will force Yable to produce a "monolithic" file. If at the moment of the export neither Path or !INC are used, Yable will use automatically the Path option.

**Rendering problems:** Sometimes, by using different files, Yable incurs in some problems and may mix the objects created in previous rendering. In such case is worthwhile to use the single file, that is surely overwritten every time, or to delete the old XMLs.

The Anim Button forces a different XML file to be exported for *each* frame of the Blender scene. Once the frames are rendered you can compose them into an animation by using Blender's Sequence Editor. The Anim button implies that the Fr (frame) button is also pressed, which appends to the chosen name for XML and images files a number suffix that indicates the rendered frame. All the XMLs are saved in a separate subdirectory named as the blender file with the \_MOVIE suffix.

An example: Suppose our YABLEROOT is C:/bar/ and we are working with the file robot.blend, when we press Export, Yable will create, first of all, a folder C:/bar/robot/; then, if we have specified the Path option, inside this directory the main XML file (called robot.xml) will be created, and two folders: Materials and Meshes, from which YafRay, reading the paths in robot.xml, will draw the data for the materials and objects.

In the case in which we choose to use the <code>!INC</code> button, no folders will be created within <code>C:/bar/robot/</code>, but only a single file <code>robot.xml</code>, that contains all data.

Finally, if the Anim button has been utilized, another folder will be created, C:/foo/robot/robot\_MOVIE/ containing as many XML robot.001.xml, robot.002.xml, robot.003.xml,... as there are frames in the animation.

Note that if the files of the animation are obtained with the Path option, it will be necessary to copy the Meshes and Materials folders in the robot\_MOVIE folder. If the animation does not include transformations of morphing (as an example RVK), is safe to leave de-activated the button Update Mesh. Otherwise for every exported frame every mesh of the scene will be exported too.

The last three buttons are GZ, Time, and alphaTGA: the first enable the creation of gzipped files, the second let the time employed for the rendering appear on the YafRay console and the third modifies the XML so that YafRay saves TGA images with the alpha channel.

• Antialiasing setting (Figure 28-3#13) - AAP Num Button indicates the number of passes of antialiasing; putting it equal to zero indicates no antialias. AAMS adjust the number of samplings to use for every AA pass. AAPW adjust the pixel width parameter, that is the overlap of pixels; the range vary between 0 and 2, and using high values, we can obtains a better smoothness, even if sometimes too much accentuate. AAT establishes the value of threshold (AA\_threshold) beyond which the pixel will be processed from the antialiasing: the value can vary between 0 (all points will be processed) and 1 (no pixel are processed). CPU indicates, if multiple CPU are present, how many should be used for rendering.

# **Material Setting**

The second GUI panel contains the Material Setting. Here it is possible to assign to every object the material that will be used in the YafRay rendering. The Materials assigned with Yable and Blender materials are two different things: the script draws from the scene only some values, like the UV coordinates and the diffuse color. This latter only upon request of the user. The rest is all independent; from this point of view Yable is a sort of laboratory: it not only exports passively the scene, but it allows us to study and to apply new materials.

Once an object is selected we must press the Get Selected Button: new buttons will appear; containing YafRay settings if the Object already has them, or empty otherwise(Figure 28-12).



Figure 28-12. Material Buttons.

• *Shader Type* (Figure 28-12#14) - This button allows you to choose the Type of Material Shader to apply.

The Constant shader is the simplest, characterized only from the Red, Green and Blu Num Buttons. Crafter is a distinct case: it is an interface used to loading the Crafter shader, that is a stand alone program for the visual composition of materials. Generic is the more versatile material, and includes also the characteristics of

the others, hence we will describe it deeply, using it like a paradigm for the general understanding.

• Object Attributes (Figure 28-12#14) - Pressing this button some new buttons appear (Figure 28-12 on the right). They are very important characteristics, which are linked to the *Object* and not to the *Material* itself. The Cast Shadow toggles if the object projects shadows or not. The Caustic IOR button enables the calculation of the caustics for the light beams that will pass through the object; these will be deflected according to the value of the refractive index indicated from CausIOR NumButton. High IOR values produce more sharp caustics (think, as an idea, to the lens that concentrates the solar beams in a point). The Receive Radio and the Emit Radio buttons, if pressed, will force the object to participate to the calculation of the global illumination, receiving and re-emitting energy. The Caustic Tcolor Num Buttons allows you to specify the transmitted colour, that is the colour assumed from the light passing through the Object. The Caustic Rcolor Num Buttons, on the other hand, refers to the light reflected by the Object.

**Note:** Note that even if we set correctly a material, it will participate to the effects of caustics and radiosity only if it is illuminated with appropriate lights, like the Path Light, or the Photon Light.

- *Diffuse Colour* (Figure 28-12#16) Is the basic color, and corresponds to the Diffuse Colour in Blender materials. The Bl button on the side is used take RGB values directly from the Blender Material. Pressing the button Add Specular Color new buttons appear similar to those just seen, used to set the specular color. Also in this case the meaning of this Colour is the same to that of Blender.
- *Reflection and Transmission Colours* (Figure 28-12#17 and #18) It is possible to set the Reflected and Transparency colours of the material. Note that also the transparency is set using the RGB Num Buttons to define a Colour, not just a plain "alpha" value.

The Transmit parameter does not decide the degree of transparency, but only which color of the light pass through (or is blocked by) the material. To make some example, using the black colour we impose that no color passes through the material, using the red one, we would mean that the object is transparent only for the red component of the light, using the white we let all the light to pass. Pressing also the buttons Refl2 and Transm2 which appears we can (and new RGB Num Buttons are created) define a different behaviour of the material at grazing light incidence.

• *Hardness, Index of Refraction* (Figure 28-12#19) - The hard parameter governs the sharpness of specular highlights exactly as in Blender. The refractive index IOR is fundamental in transparent objects, and is used in order to calculate the deviation of the light beams that crosses the material. As a result of this effect, the bodies immersed in a transparent medium appears to us distorted (think to a paddle immersed in the limpid water). Table 28-1 shows some IORs of common materials:

Material	IOR		
Void	1.0		
Air	1.00029		
Ice	1.31		

#### Table 28-1. Sample IORs

Material	IOR
Water (at 20C)	1.33
Ethylic Alcohol	1.36
Glycerine	1.473
Glass	1.52
Sapphire	1.77
Diamond	2.417

- *Get Selected* (Figure 28-12#20) The Get Selected, to be pressed every time *after* you have selected the object and *before* starting to modify the material.
- *Get Selected, Loading and saving the materials* (Figure 28-12#21) The Load Material and Save Material buttons allows you to save material settings and to quickly call them back. A name can be given to each material.
- *Preview of the material, autosmooth and modulators* (Figure 28-12#22) The AutoSmooth Button is used to regulate the appearance of the surfaces. Pressing it an additional Num Button appears that regulated the angle under which the corner of the two faces is considered smooth, exactly as in Blender.

The Mat preview Button creates a preview of the material using a sample scene. The tga is saved in the current directory (for example, under Windows, it is saved in the folder in which the Blender executable resides.) The button phlightprv indicate "Photon Light Preview" and it is used to put a photonic light during the materials preview.

The Modulators Button allows you to access a separate panel for the composition of advanced shaders formed by overlying layers. Every layer can be an image or a procedural texture. Obviously is possible to set the modality with which the layers must be mixed and also the percentage of transparency.

The panel at the beginning refers to the default modulators, that is used automatically if the Object has UV coordinates and is mapped with an image in Blender. Unfortunately, because of a bug, these formulations are not maintained by Yable, that continues to use the default settings. However all the successive layers of modulators that are added work correctly. In order to add a new component is sufficient to click on the Others button, and to choose the kind of modulator that we want. In Figure 28-13 we see (starting from the top left, clockwise) the main panel, the panel adding new members, the menu Others, before and after the addiction of a new member:

Mix  Mul  Add    Col  color: 1.000	UV Texture Settings These values will affect only the Blender Texture! SizeX: 1.000 SizeY: 1.000 Size: 1.00
-	
Others:	Image: Style MB & NAME    Mix  Mage: Style MB & NAME    Mix  Mage: Style MB & NAME    Mix  Mage: Style MB & NAME    Style MB & NAME  Style MB & NAME    Style MB & NAME  Style MB & NAME    Mix  Name
	Ok! Cano

Figure 28-13. Material Modulators.

It is possible to add an Image layer, a Clouds layer and a Marble one. The first thing is to assign a name at the modulator created, to do this it will be sufficient to write something of meaningful in place of GIVE\_ME\_A\_NAME.

If the Modulator is an image it is necessary to insert the name of the image itself: only the name and extension, without path, which has been defined once for all before!

The parameters refer, by default, to bump mapping, that is to the relief effect that will be given to the object: using positive values rises clearer zones and lowers darker. The various size refer to the scaling of the UV coordinates (the three X, Y, Z axis may be scaled independently, or all together) while the various buttons Col, Spec, Ref, Hard and Trans have the usual meanings and indicates which characteristic, and by how much, is modified. At the end of the settings, press the Ok! button, to return to the main Material panel. At the end of the procedure the new added modulator will be in the Menu and can be selected and cancelled anytime, using the button Del that appear next to the Ok! Canc and Back buttons.

**Note:** The only type of mapping supported up to now is the UV type. It follow that the object must possess these coordinates, for which we sends back to the relevant Section of this Guide. If all is correctly executed, Yable will export automatically (without the adding of an image type modulator) both the UV coordinates and the used image. About this image we must pay attention to this: Yable does not use the image loaded in the texture of Blender's materials, but the one loaded in the Image Window (that is the one on which calculations for the positioning of the UV are made). Obviously, the images must all be in the usual folder specified at the start of the script.

Clouds and Marble Modulators insertion is similar to that for the Images, except that these panels have few additional, self explanatory, specific parameters for these two types of procedural texture.

• Assign (Figure 28-12#23) - The Assign button finalizes the material and assigns it to the object. Don't forget this! The button Selected All allows you to assign the settings to more than one selected objects.

**Note:** Saving a material and Assigning a material are two separate actions. If you assign it the Object acquires that material, if you save it, it will be available later on.

## **Light Settings**

The lights setting in Yable is made with the same modality of the materials assignation: a light is selected in Blender, Get Selected is pressed and the characteristics that we want to to export in YafRay are chosen and assigned definitively with Assign.

The type of light used in the scene of Blender *do not* have any relationship with the type of light that will be exported: the only parameters that will be surely conserved are the positional coordinates of the lamp; all the rest, included the pointing direction can be assigned independently with Yable. In Figure 28-14 we have represented a point light with the buttons Diffuse and Caustic activated, so to have an example that include the greater part of the available options.



Figure 28-14. Light GUI Panel.

• Light Types (Figure 28-12#24) - The menu lets us choose between various types of direct light: Point Light, Spot Ligth, Soft light, Area Light and Photon Light . Accordingly to the light type the options immediately beneath varies. Figure 28-15 shows them all.

| DIRECT Light Type |
|-------------------|-------------------|-------------------|-------------------|-------------------|
| Spot Light 🚽      | Point Light 🖃     | Sun Light -       | Soft Light -      | Area Light -      |
| Power: 11         | Power: 1.00       | Power: 1.00       | Power: 1.00       | Power: 25         |
| Red 1.000         |
| Green 1.000       |
| Blu 1.000         |
Width: 45.00	Cast Shadow	Cast Shadow	Resol: 200	Samples: 64
Blend: 20.00			Radius: 5	PSamples: 64
Falloff: 2.00			Bias: 0.300	Side: 5.00
Cast Shadow				
Halo				

Figure 28-15. Direct Light Options.
The Point light is a point source that emits light in all the directions. The power of the lamp is chosen with the Power Num Button, while its colour is set via RGB Num Buttons and is possible to choose if it must project or not a shadow with the Cast Shadow Button.

The Spot light is very similar to the Blender spot: the parameters Blend and Falloff have the same meaning, while Width represents the angular width of the light cone.

Halo indicates the presence of the halo (volumetric light), by pressing it we add some new buttons: three Num Buttons for the Halo colour, Res = resolution of the shadow map, Density = quantity of fog contained in the halo, Blur = blur applied to shadow map, Samples = number of samplings used in the rendering.

The direction of the Spot is given via a target. We must press the Select Target button, select an Object to be the target of the spot in the Blender scene and finally press the Confirm Button to complete the operation.

Both the Point and the Spot light decreases following the physical law of the inverse square of the distance.

The Soft light is similar to the Pointlight, with the difference that it produces soft shadows. Shadows which appear to be too "clean" are one of the disadvantages of the raytrace engines, but this type of light, using a shadow map, safely resolves the problem. Besides the usual parameters, the Radius parameter, defines the width of the transition between shadow and light. The Bias parameter controls the proximity of the shadow to the object that produces it, while the Resol parameter controls the resolution of the shadow map: the higher is this parameter the better is the accuracy of the shadow.

The Sun light simulates the characteristic of the solar light, it seems not to decay with increasing of the distance (it is an impression due to the enormous power of the Sun). It is, therefore, a much simple light, in which we can only set up the color, and whose intensity remains constant.

The Area light is an extended luminous source. While all those already seen emit light from a point (in reality it corresponds, as an example, to the small filament of a light bulb), the area light is produced from a whole surface. YafRay admit also quadrilateral surfaces but Yable is limited to use only squares. The specific parameters are Samples, Psamples and Side, they are the number of general samplings, the number of samplings in the penumbra zone, and the length of the side of the light casting square.

Photon Lights (Diffuse and Caustic) (Figure 28-12#25 and #26) - Photonlight is a very peculiar kind of lamp: it behave in a more realistic way, referring itself to the theory of the light composed by a bundle of photons. These photons must literally be "shoot" towards the Objects, so as to calculate their behaviour when they travel through transparent bodies or are diffused by opaque ones. It is a very complex calculation, for which is not always desirable that it will be executed on all the elements of the scene; therefore we can specify it on a material to material and object by object basys via the parameters Receive and Emit.

*Diffuse light*: Enabling this button a Photon Light of Diffuse type will be add to the exported scene, overlapping the corresponding Direct light. The photons of this light have the capability to be reflected on diffusive surfaces. In this way calculations of the radiosity 'colour leakage' will be added in the rendering, to realize a realistic effect of Global Illumination. This can be done also via Path Light, but a carefully tuned Photon Light is much faster.

*Caustic light*: Caustic are concentrations of light caused from the refraction of the transparent objects. The Photon Light of Caustic type allows you to account for this phenomenon correctly. Before going ahead, it is necessary to say that if targeted to objects with inadequate material, this light does not generate any effect. In fact the scene, to have caustics, must contain a source light A, an opaque object C *receiving* 

the caustics and a transparent object B placed between these two *generating* the caustics (Figure 28-16).



Figure 28-16. Caustics setup.

- A must be a caustic photonlight;
- B must have a Material able to produce Caustics (exhibiting *at least* assigning at least Caustic IOR and Caustic Tcolor);
- C must have a Material able to receive the radiosity effects (for this the default material could be good, because this material export a simple shader that, without specify the received and emitted values of radiosity leave the choice to YafRay, that usually hold this values activated. If, indeed, a generic material is used, we must remember to activate the Reiceve Radio Toggle Button).

The Photon Light of Caustic and Diffuse type have the same parameters. The arrows button is used to impose the same value to more than one variables (for example the same color ).

The Photons button set the number of photons that will be shot from the lamp, in common scene a few thousands of photons is enough but, to obtain a better results in terms of quality is recommended to set 50000 photons or more. The Depth button defines the number of bounces/transmissions the photons can do before YafRay ray tracing stops to handle it, and it is a particularly important data, mostly in the calculation of the Global illumination. 3 is a good value for acceptable results. Search indicates the numbers of photons that can be used to illuminate a single point of the Object surface, higher values is used to take in consideration also zones that receive few photons, with a shaded effect of illumination, while low value is used to give light only to the points really hit by photons completely, with more definite and "hard" boundaries. Angle is the angle of the projection cone with which the photon are shot: high values are used to cover a wide area, but with power fading when we depart from the center. Fixed is the abbreviation of fixedradius, and represent the radius in which the number of photons defined by Search must fall in order to consider the point of the surface illuminated. cluster is the smallest portion of lit surface able to contain a photon. The higher is this number the larger is the width of the cluster and, consequently, the less defined is the illumination effect. QMC is, again, the quasi-MonteCarlo method.

# Yable Juicy example

Relevant to Blender v2.31



Figure 28-17. Caustics setup.

What you see in (Figure 28-17) is a completely ic image realized by Xavier 'richie' Ligey, modelled with Blender and rendered with YafRay. The export has been made with Yable. There is no light at all in the scene, only a Hemi Light with an HDRI background. Xavier has been so kind as to give us the screenshots of the settings he has used in Yable.

Figure 28-18 shows the car paint, Figure 28-19 shows the chrome material and Figure 28-20 shows the glass material.

Chapter 28. From Blender to YafRay Using YableX

Global Settings Material Settings Light Editor
Shader Type Generic - Object Attributes
Diffuse Color Red 0.070 Green 0.137 Green 0.137 Green 0.290 Green 0.137 Green
BI Ref 1 Ref 2 R 0.900 B 0.900 Transm 1
Load Material Save Material
Hard 25 Add IOR AutoSmooth Degr:90 MAT Preview phlightprv Fast Fresnel Modulators
Get Selected Name: Plane.015 Assign All Assign Type: Mesh Yable-0.30 Yet Another Biender Exporter

Figure 28-18. Car Paint.

Global Settings Material Setti	ngs Light Editor
Shader Type Generic 🖃	Object Attributes
Diffuse Color Red 0.100 Green 0.100 Diffuse Color Blu 0.100 Add Specular Color	
B1 Ref 1 Ref 2 R 0.700 B 0.700 B 0.700 Transm 1	
	Load Material Save Material
Hard 25 Add IOR	AutoSmooth Degr:90
Get Selected Name: Pla Type: Me	ane.110 sh /able-0 au yet another stenger =>oorter

Figure 28-19. Chrome parts.

Global Settings Material Settings	Light Editor
Shader Type Generic 🖃	Object Attributes
Diffuse Color Red 0.200 Green 0.200 Blu 0.200 Add Specular Color	
BI Ref 1 Ref 2 R 0.900 G 0.900 Transm 1 Transm 2 R 0.800 B 0.800 B 0.800	Load Material Save Material
Hard 25 Caracteria Add IOR IOR 1.05 Caracteria International Add IOR Min Ref 0.192 Caracteria International Address Stressel	AutoSmooth Degr:90 MAT Preview phlightprv Modulators
Get Selected Name: Plane Type: Mesh Yabl	Assign All Assign =-0.30 Yet Another Blender Exporter

Figure 28-20. Windshield and other glasses.

A special thank to Alessandro Braccili, who helped me in Yable/YafRay understanding, and in the writing of this chapter.

# Notes

- 1. http://www.zoo-logique.org/3D.Blender/
- 2. http://www.yafray.org/

Chapter 28. From Blender to YafRay Using YableX

# Chapter 29. YafRay

by Alejandro Conty Estevez, Chris Williamson, Johnny Matthews.

## Introduction

Relevant to Blender v2.31

by Alejandro Conty Estevez,

By the time I started working with YafRay, I was checking out some blender exporters like BMRT and Lightflow. While I was writing some exporting and shading code, I began to be interested in how a raytracer could be written. So when the exams season was in full swing, I became bored (as weird as it may sound) and began to write the main program structure. Once I got a few test renders, I put it off for a year, till the next summer. Then, I wrote the XML loader and YafRay, called "noname" by that moment, began to be an usable program.

Alfredo joined the development almost at the same time. That was of great help. A month later a lot of necessary stuff, like acceleration, were finished and Alfredo ported a lot of his code to YafRay. As the famous hemilight.

Then Luis Fernando Ruiz, a friend of mine and classmate joined to give us a good web site. So we said good bye to that boring plain text web site. We also had the chance to see YafRay rendering on several computers concurrently when Luciano Campal wrote his hack to make YafRay able to work in a distributed way thanks to mosix. It was very exciting when we got access to a 20 computers room for testing. Things started to look very promising when Andrea came with Yable. An experimental export script for an experimental renderer that resulted in a very long thread of cool images at elYsiun. We saw the first nice images done with blender and rendered with YafRay thanks to him.

We didn't expect that boom. Neither Alfredo nor me. Of course it was the cool export script what was catching people, exporting easily from blender to a raytracer. We got very excited with all that support from the community. I still get impressed by what people can do with a simple tool like this.

Now more people are getting involved and helping. We begin to have a good documentation section and resources, most of which have been written by Chris Williamson. Basically, it's what you'll see in this chapter. But he is not the only one. YafRay is also getting very easy to use from blender thanks to Johnny Matthews. I think he spends almost every minute writing Extractor: a new export script for blender. It makes the exporting much more easy by getting all the data directly from blender with nearly no user interaction.

The current power of Extractor and its fast development point out that this could be the future official export scheme for exporting from blender. Anyway, efforts are being made to write a built-in exporter in blender. Alfredo contributed with a lot of shading compatibility code and did some experiments. So it seems we will be able to compare both python and built-in solutions at some point.

YafRay started as an experiment and still is. It's not finished and lacks a lot of features if you compare it with other render engines. I always think is not good enough and that it is hard to imagine what do people see in it. Since people like it for some reason, we now want to really convert it into a full rendering engine that deserves to be called "renderer". This will take some time to have fun coding. We want to add what YafRay lacks (particles, effects, etc...) and to improve global illumination. But only Alfredo De Greef and me are coding YafRay right now, so in order to keep the development up to an acceptable rate, we should get more people to code, more developers. I hope this happens sooner or later.

Finally, I want to thank all the blender community that supported this project. All those beautiful pictures are what really bring people to YafRay. Likewise, thanks to

all the people who give ideas and thoughts on the forums to improve YafRay, and to Juan David G. Cobas for his very appreciated math support.

## Installation

Relevant to Blender v2.31

YafRay is available for Linux, Windows and Mac OSX. Download the package suitable for your OS at www.yafray.org<sup>1</sup>.

### YafRay for Windows

Run the installer program. It will create a directory called "yafray" on your c:\ drive. This directory contains the yafray.exe executable and the grammar file which are used by the loader. Also the installer copies three dll's into your Windows system directory. These dll's are for cygwin support. Finally, a batch file (yafray.bat) is copied into the Windows directory (we need this file in the PATH).

To run YafRay is easy. Just open a MS-DOS window, go to you working directory and type "yafray file.xml" or "yafray file.xml.gz". For example, if you want to work in e:\raytracing\work on an XML file which resides in C:\Docs\xmls named test.xml, you open a MS-DOS windows and:

```
c:\windows\>
c:\windows\> e:
e:> cd e:\raytracing\work
e:\raytracing\work> yafray c:\Docs\xmls\test.xml
```

One or more targa file, the output of the render, will be created in the e:\raytracing\work directory.

### YafRay for Mac OSX

Expand the tarball. (stuffIt expander can expand tarball also). Double click on the expanded package to run installer. YafRay must be installed on the Root device (the one on which it has been installed MacOSX), *you cannot choose any other disk*.

Installed files and location are: /usr/sbin/yafray /usr/etc/gram.yafray

YafRay utilization does not differ noticeably from the three OS, so you can refer to the previous section.

YafRay has just 2 files ' /usr/sbin/yafray ' and ' /usr/etc/gram.yafray'. But a common user cannot usually access these directories via the Mac OSX GUI, so the *OS X Package Manager* (OSXPM) can help you to uninstall packages from your disk.

#### YafRay on Linux

Expand the tarball.

tar xvzf yafray-#.#.#.tar.gz

Go into the newly created directory and configure it for your machine.

./configure

Make sure that zlib and jpeg support is enabled. If not, you need install devel packages for libjpeg and libgz (check your distribution for it).

Build it!

make

If this fails you can try

cd src make yafray

The executable is yafray and is a command Line program, whose usage is analogous to the what described In the "Windows" section.

## Scene Description Language Overview

#### Relevant to Blender v2.31

A YafRay Scene description file is an XML file complying to the definitions of this section. The renderer parses the XML from top to bottom. So if Block1 is referenced before Block2, it must be defined before Block2 (it must be above it in the XML).

<scene>

```
<shader type = "generic" name = "Default">
        <attributes>
                <color r="0.750000" g="0.750000" b="0.800000" />
                <specular r="0.000000" g="0.000000" b="0.000000" />
                <reflected r="0.000000" g="0.000000" b="0.000000" />
                <transmitted r="0.000000" g="0.000000" b="0.000000" />
        </attributes>
</shader>
<transform
m00 = "8.532125" m01 = "0.000000" m02 = "0.000000" m03 = "0.000000"
m10 = "0.000000" m11 = "8.532125" m12 = "0.000000" m13 = "0.000000"
m20 = "0.000000" m21 = "0.000000" m22 = "8.532125" m23 = "0.000000"
m30 = "0.000000" m31 = "0.000000" m32 = "0.000000" m33 = "1.000000"
>
<object name = "Plane" shader_name = "Default" >
        <attributes>
        </attributes>
                <mesh>
                <include file = ".\Meshes\Plane.xml" />
                </mesh>
</object>
</transform>
depth type="pathlight" name="path" power= "1.000000" depth "2" samples = "16" use_QMC
gle_threshold="0.200000" shadow_threshold="0.200000" >
</light>
<camera name="Camera" resx="1024" resy="576" focal="1.015937" >
        <from x="0.323759" y="-7.701275" z="2.818493" />
        <to x="0.318982" y="-6.717273" z="2.640400" />
        <up x="0.323330" y="-7.523182" z="3.802506" />
</camera>
<filter type="dof" name="dof" focus = "7.97854234329" near_blur "10.000000" far_blur "1</pre>
</filter>
```

Don't worry! It's not as complex as it looks. Concentrate on the bold highlighted tags.

The Tags work similar to HTML tags (also like brackets) each tag must have an opposite closing tag. Two tags together, with settings inside, is one block. A block can tell the renderer how to shade something, how big to render the image, what the shape of an object looks like, where it is etc etc.

In the example above, first a shader is defined, then an object (which is wrapped in its Transform Matrix), then a light is added then a camera, a filter, a background and finally the render settings (notice the closing </scene> tag).

## Shaders

Relevant to Blender v2.31

### **Base Shaders**

These shader blocks determine the BiDirectional Reflectivity Function (BDRF) or Illumination Model that the object is shaded with. Each different base shader type has various inputs that can receive the outputs from other shader blocks, altering the surface characteristics.

### Constant

A uniformly constant shader

#### Generic

The most versatile shader

```
<reflected2 r="1.000000" g="1.000000" b="1.000000" />
                   <transmitted r="0.197183" g="0.197183" b="0.225352" /><transmitted2 r="1.000000" g="1.000000" b="1.000000" />
                   <hard value = "25.000000"/>
                   <IOR value = "1.592105"/>
                   <min refle value = "0.200000"/>
                   <fast_fresnel value = "off"/>
          </attributes>
</shader>
```

## Phong

Classic Phong shader

```
<shader type="phong" name="phongshader">
                <attributes>
                        <environment value="fresnel"/>
                        <color value="rgb"/>
                </attributes>
</shader>
```

### **Procedural**

These Shading blocks create various procedural patterns with inline values. No inputs are needed.

#### Marble

```
<shader type="marble" name="Marble" size="4.00" depth="4" hard="off" tur-</pre>
bulence="5" sharpness="5.00">
        <attributes>
        </attributes>
</shader>
```

- size: Size of the marble effect, lower numbers = less veins, higher numbers = more veins.
- depth: Controls the number of iterations (number of noise frequencies added to the swirl).
- hard: Controls the noise type, when set to 'off' the noise varies smoothly while setting it to 'on' will show more abrupt changes in color.
- turbulence: Controls the amount of noise turbulence.
- sharpness: Controls the sharpness of color 1 compared to color 2, the higher this value, the thinner the color band of color1. This effect is similar to the soft/sharp/sharper switches of the Blender marble texture, the difference is that it is more controlable here. The value must be at least 1 or higher.

### Chapter 29. YafRay



## Wood

- size: Size of the wood effect, lower numbers = less wood grain, higher numbers = more wood grain.
- depth: Controls the number of iterations (number of noise frequencies added to the swirl).
- hard: Controls the noise type, when set to 'off' the noise varies smoothly while setting it to 'on' will show more abrupt changes in color.
- turbulence: Controls the amount of noise turbulence.
- ringscale\_x: Controls the width of the wood rings in the x axis.
- ringscale\_y: Controls the width of the wood rings in the y axis.



# Clouds

- size: Size of the cloud effect.
- depth: Controls the number of iterations (number of noise frequencies added to the swirl).



# **Meta Shaders**

These allow the modification of other shaders and the building of "chains" of simple shaders to build a complex shader.

### Color2float shading block

Takes a color as input and outputs a float <shader type="color2float" name="c2f" input="input" >

```
<attributes>
</shader>
```

• input: Input (color) to convert to float.

### **Colorband shading block**

Builds a color from a value input and a gradient. An unlimited number of modulators add nodes to the gradient. The shader interpolates the color values of the nodes at the given input value. In the example below, an input value of 0.12 would generate a colour between the first and second node, which are black and orange. So a dark orange would be the result.

```
<shader type="colorband" name="Colorband" >
   <attributes>
        <input value="Wood" />
        </attributes>
        <modulator value="0.00"><color r="0.00" g="0.00" b="0.00" /></modulator>
        <modulator value="0.26"><color r="1.00" g="0.36" b="0.00" /></modulator>
        <modulator value="0.66"><color r="1.00" g="1.00" b="0.00" /></modulator>
        <modulator value="1.00"><color r="1.00" g="1.00" b="1.00" /></modulator>
        <modulator value="1.00"><</pre>
```

### **Conetrace block**

It can be used to get reflections or transmitted color from environment. But it could also be used to get blurry ones.

```
<shader type="conetrace" name="envl" reflect="on/off" angle="number"
samples="number" IOR="number">
    <attributes>
        <color ... />
        </attributes>
        </attributes>
        </shader>
```

- reflect: 'on' will reflect the ray, 'off' will refract the ray.
- angle: Angle of the cone (around the ray) to be sampled, 0 for a simple sharp reflection/refraction.
- samples: Number of samples to take inside the cone.
- IOR: Index of Refraction.
- color: Color to filter the incoming light.



Spheres with varying levels of blurry reflections and refractions and an HDRI background.

# **Coords shading block**

Outputs a float based on object coords.

• coord: Coordinate to use, either X, Y or Z.



Clouds shader and coords shader (z) into multiply shader.

## Float2color shading block

Takes a float as input and outputs a color.

• input: Input (float) to convert to color.

## **Fresnel shading block**

Can be used to get realistic reflections/refractions based on the angle of incidence.

```
<shader type="fresnel" name="fresnel1" reflected="..." transmitted="..."
IOR="number" min_refle="number">
    <attributes>
    </attributes>
</shader>
```

- reflected: The input to use as reflected color (usually the conetrace output).
- transmitted: The input to use as transmitted color (usually another conetrace output).
- IOR: Index of Refraction.
- min\_refle: Minimal reflection amount.



Mixing fresnel and conetrace blocks.

### **HSV** shading block

Builds a color from either any inputs or inline values for HSV components.

```
<shader type="HSV" inputhue="..." inputsaturation="..." inputvalue="..."
hue="number" saturation="number" value="number" >
</shader>
```

As in the RGB color, if the inputs are omitted, inline hue/saturation/value values are used.

#### Image shading block

Assigns a bitmap image to the object according to its UV co-ordinates (outputs Color).

```
<shader type = "image" name = "bitmap">
<attributes>
<filename value = "c:\filename.tga" />
</attributes>
</shader>
```

• filename: Path and name of bitmap to apply.

## Mix shading block

Mixes x2 inputs in different ways, depending on the mode used.

</shader>

- input1: First input to mix.
- input 2: Second input to mix.
- mode: Possible mix modes (note, some modes output different results depending on the order of the inputs).

These are: Add, Average, Color Burn, Color Dodge, Darken, Difference, Exclusion, Freeze, Hard Light, Lighten, Multiply, Negation, Overlay, Reflect, Screen, Soft light, Stamp, Subtractive.

### Multiply shading block

Multiplies (float) input values or input value and const value, outputs a float.

- input1: First input to multiply.
- input2: Second input to multiply (if null, input1 is multiplied with the inline 'value' setting).
- value: Value to multiply if input2 is null.



Multiply shader with wood and marble as input.

## **RGB shading block**

Builds a color from either any inputs or inline values for RGB components.

```
<shader type="RGB" inputred="..." inputgreen="..." inputblue="..." >
  <color ...>
</shader>
```

If one of the inputs is omitted, then the default color given by "color" tag is used for that input.

## Sin shading block

Generates float values based on sine wave and input.



Wood shader is input for sin, which in turn is an input for the Hue channel of an HSV shader - the color value of a phong shader.

# **Renderable Objects**

Relevant to Blender v2.31

YafRay currently supports only Mesh Objects. An example of a simple triangular planar plate is:

```
<transform
m00 = "0.997525" m01 = "0.070303" m02 = "0.001329" m03 = "0.115816"
m10 = "-0.018745" m11 = "0.284097" m12 = "-0.958612" m13 = "1.522439"
m20 = "-0.067771" m21 = "0.956215" m22 = "0.284711" m23 = "3.272361"
m30 = "0.000000" m31 = "0.000000" m32 = "0.000000" m33 = "1.000000"
>
<object name = "Plane.002" shader_name = "Plane.002.mat" caus_IOR = "1.500000" recv_rad</pre>
       shadow = "on">
       <attributes>
               <caus_tcolor r = "1.000000" g = "1.000000" b = "1.000000"/>
               <caus_rcolor r = "1.000000" g = "1.000000" b = "1.000000" />
       </attributes>
               <mesh
                       autosmooth = "30.0" >
<points>

 </points>
<faces>
       <f a="0" b="2" c="1" />
                       </faces>
               </mesh>
</object>
</transform>
```

### Transform tag

The Transform tag defines the transform matrix for the enclosed object (position, scale and rotation from the world origin point).

### Object tag

The object tag defines the object geometry, it has a series of parameters:

- caus\_IOR: Index of Refraction for Caustic Photons.
- recv\_rad: Whether to receive radiosity (bounced light generated from Photon Lights) ('on' or 'off').
- emit\_rad: Whether to emit radiosity ('on' or 'off').
- shadow: Whether or not to cast a shadow ('on' or 'off').
- caus\_tcolor: The colour to tint transmitted photons (colours the refractive caustics).
- caus\_rcolor: The colour to tint reflected photons (colours the reflective caustics).

## The Mesh tag:

The key tag within an object definition is the Mesh tag, which defines its geometry:

autosmooth: Threshold angle for the smoothing algorithm (omit to get 'faceted' objects).

Within the mesh tag a <points> block defines the vertices of the mesh, while a <faces> block defines triangular faces by the indices of their three vertices.

## Lights

Relevant to Blender v2.31

YafRay provides several kind of lights:

### Spot light

This is fairly similar to Blender's Spot Light

- size: Angle of the cone (half of Blender's one!).
- blend and beam\_falloff: same as in Blender.
- halo: Whenever to cast volumetic light and shadows or not.

- res: Resolution of the shadowmap (only for volumetric shadows by now).
- blur: Blur applied to the volumetric shadows range from 0.0 to 1.0
- fog\_density: Amount of fog in the halo.
- samples: Number of samples to use for halo rendering. By default the same as res. The more samples, the less noise.
- from: Position of the light.
- to: Target of the light.
- color: Color of the light.
- fog: Color of volumetric light.



## **Point light**

This is fairly similar to Blender's Lamp... but it casts shadows!

- power: Light intensity.
- cast\_shadows: Whether or not to cast (raytraced) shadows ('on' or 'off').
- from: Position of the light.

### Chapter 29. YafRay

• color: RGB color of the light.



Pointlight, with Hemilight (white dot represents light position).

# Sun light

This is again similar to Blender's Sun... but it casts shadows!

- power: Intensity of the sunlight.
- cast\_shadows: Whether or not to cast shadows ('on' or 'off').
- from: Position of the light (direction is automatically towards the origin!)
- color: Color of the light.



sunlight, with hemilight to provide some diffuse shading

# Soft light

- power: Intensity of the light.
- res: Resolution of the shadowmap.
- radius: Radius of the blur (between shadowed and non-shadowed areas, creates the 'soft' look).
- bias: Bias of the shadow map. 'Closeness' of the shadow to the object, if you have shadow that 'leaks' into areas it shouldn't, try decreasing the shadow bias.



# Area light

This is a light shed uniformly by a quadrilateral

- power: Intensity of the light.
- samples: Samples across area light surface.
- psamples: Penumbra prediction (quality of blurred shadow edges).
- a, b, c & d: Positions of the 4 corners of the rectangle that makes up the area light. (Beware of orientation!)



Arealight, with Hemilight (white dot represents light position).

## Path Light

This indirect lighting system performs 'global illumination' by taking light from background and diffuse objects. It uses either a Monte Carlo raytracing algorithm (MC), or a Quasi Monte Carlo raytracing algorithm (QMC). The results from either system can be rendered using an Irradiance Cache.

Since MC uses random sampling the results can be quite noisy. The more samples you take the less noise you'll see. Of course this results in a longer render time. QMC sampling on the other hand produces less noise, but sometimes can result in discernible patterns in the shading of objects. Both noise and patterns can be reduced with yafray built-in Anti Noise Filter.

Path Light will produce nice radiosity effects. It can also produce caustics, however, as the photons that produce the caustics are not focused in a specific direction like the photon light, the caustic patterns will be softer unless a huge number of samples are taken.

```
type = "pathlight" name = "path" power = "1.000000" depth = "2" caus_depth = "4"
ples = "16" use_QMC = "on" cache = "on" cache_size = "0.008000" angle_threshold = "0.20
</light>
```

- power: Intensity of the light.
- samples: Number of samples to take per pixel to get a quick and dirty preview of your render, you can set this number low, then raise it to get the final render.
- depth: Number of ray bounces for each sample, at least 2 to get indirect lighting.
- caus\_depth: Number of ray bounces when passing through caustic objects.
- use\_QMC: Whenever this is set 'on' will use quasi montecarlo sampling.
- cache: When this is set to on, Yafray will perform a prepass render to generate an irradiance cache.
- cache\_size: The size of the grid in the irradiance cache. Smaller values will mean
  a higher resolution irradiance cache (and longer prepass times).

- angle\_threshold: The angle between surface normals that determine whether the caching algorithm considers the surface 'flat', if the surface normal variation is higher than this, the caching algorithm takes more samples.
- shadow\_threshold: The minimum distance from the sample point an object can be before the caching algorithm takes more samples.

### Using the Cached Pathlight

Using the Irradiance Cache feature can be tricky, the results are well worth it, as you can usually get the same quality image in a fraction of the render time.

The cache size is the size of a grid that the scene is divided up into. As the rays are shot into the scene, they intersect objects, the point at which the ray intersects the object will therefore fall into one of the boxes formed by the cache grid. At the time the ray hits this point the renderer first asks:

1. "Are there any other sample points within this box?" If the answer is no, a sample is taken, if there are other samples in the box it moves on to the next question:

2. "Are the surface normals of the other samples different to my current point?" (the angle of difference is defined by "angle\_threshold"). If the answer is yes, a sample is taken, if the surface normals are all the same it moves on to the next question:

3. "Is the Intersection Point close to any other object?" (the distance threshold is defined by "shadow\_threshold" and the distance between the intersection point and the existing sample points plays a part). If the answer is yes, more samples are taken, if no, the sample point is skipped and the renderer moves on to the next intersection point.

By doing this, the renderer finds areas of the image that need more samples (areas of high detail), and areas that need less samples (areas that have low detail, such as flat walls). Taking samples is the time consuming part of Global Illumination, by only taking samples where they are necessary the cached pathlight can produce fantastic images in a comparatively short time.

It does rely on some manual tweaking to find the 'sweet spot' for the settings for any given scene. If the cache size is too small, practically every sample point will be taken anyway, as the answer to question 1 will almost alway be 'no'. On the other hand if the cache size is too big, the distance in between sample points (which plays a part in determining whether another object is considered 'close') will be such that the answer to question 3 will almost always be 'yes'. Both of these situations will result in more samples being taken and the render taking longer.

### Hemi Light

This indirect lighting system performs what is commonly called an 'Occlusion pass'. This produces a fast diffuse light in the scene by ignoring objects surface properties (colour) and just determining whether the point in question is in shadow or not. Because of this, the Hemilight will not produce colour bleeding between objects (unlike the pathlight). It uses either a Monte Carlo raytracing algorithm (MC), or a Quasi Monte Carlo raytracing algorithm (QMC). Since MC uses random sampling the results can be quite noisy. The more samples you take the less noise you'll see, of course this results in longer render time. QMC sampling on the other hand produces less noise, but sometimes can result in descernable patterns in the shading of objects. Both noise and patterns can be reduced with yafray built-in Anti Noise Filter.

The Hemilight will assume that the scene is evenly lit, as if a huge sphere surrounded the scene, lighting it with the color you specify in the 'color' tag. If you omit the color tag altogether, the hemilight will sample the render background if available (you can use this with HDRI backgrounds to get fast and realistic lighting simulations for compositing into a real scene).

```
light type="hemilight" name="sky" power= "0.500000" samples = "20" use_QMC = "on">
<color r ="0.800000" g ="0.900000" b ="1.000000" />
</light>
```

- power: Intensity of the light.
- samples: Number of samples to take. Higher samples will mean a smoother result, but longer render times.
- use\_QMC: Whenever this is set 'on' will use quasi montecarlo sampling.
- color: Color of the diffuse light.



Hemilight with light blue colour value (model from www.amazing3d.com)



## **Photon light**

This is a focused light to produce radiosity and caustics effects.

```
type= "photonlight" name="Lamp.002caus" power = "100.000000" pho-
tons = "50000" depth = "3" search = "100" angle = "15.000000"
mode = "caustic" fixedradius = "0.100000" cluster = "0.010000" use_QMC = "off"
<from x="6.372691" y="3.340035" z="2.815973" />
<to x="0.285646" y="0.149627" z="1.397566" />
<color r="1.000000" g="1.000000" b="1.000000" />
</light>
```

- name: Photonlight name.
- power: Scales the effect of the photon light, whether it be caustic or diffuse.
- mode: Sets the photon light to either diffuse, or caustic as detailed below.
- photons: Number of photons to trace, the more photons, the more information to generate the photonmap from. Generally speaking, you should need less photons for diffuse photonlights.
- depth: Amount of reflections (bounces) or refractions the photons will perform.
- search: Number of photons to gather while shading. higher values will soften the effect (when increasing the search, you should also increase the fixed radius).
- fixedradius: Search radius when looking for photons (number of photons looked for is defined by 'search'.
- cluster: This defines the smallest unit in the photonmap created. The smaller the number, the finer the photonmap.

- use\_QMC: Whenever this is set 'on' will use quasi monte carlo raytracing<sup>2</sup>.
- from: Position of the light.
- to: Target of the light.
- angle: Similar to size value for the spotlight, angle of the photon 'beam'.
- color: Color of the light.

Photon lights have two modes, "caustic" and "diffuse". In the first mode the light will draw reflected and transmitted photons, causing light to form caustic patterns of light that travel through transparent objects (ie objects that have the appropriate settings in their *Object* tag. In "diffuse" mode photons are reflected by diffuse surfaces in random directions to perform "radiosity" or "Global Illumination". In both modes only indirect light is stored (photons which have bounced at least once), so direct lighting still has to be done with a normal light.

Why not to work in both modes? You usually put different photon values for caustics than for radiosity. The needs change so is better to have two different lights for each task.

### **Tuning Photonlights**

#### photons

Choose a good value depending on the task, for radiosity you'll need few photons. For the caustics it depends on what resolution you want in the shapes.

#### search, fixed radius and cluster

These settings are closely linked, you need to get the combination of all three settings right to achieve good looking results. The search setting defines how many photons to look for from a point, the fixed radius defines how far from that point to look for the photons. Once the photons have been gathered, the fixed radius area is gridded up into small 'clusters', the size of which is defined by the cluster setting. Any photons within the same cluster are averaged into one result (equivalent to 1 pixel in the photonmap).

If your diffuse or caustic effects look fractured into geometric shapes, the algorithm is not finding the required number of photons (search) within the defined radius (fixed\_radius). To fix this you need to make sure there are enough photons within the search radius to reach the search amount. To achieve this, you could either increase the total amount of photons (photons), increase the search radius (fixed\_radius), or decrease the amount of photons searched for (search). Increasing the number of photons to see much change . Increasing the search radius will show results quickly but is also reliant on there being enough photons in the scene initially. large differences between the fixed\_radius and cluster settings (eg high fixed radius and low cluster) will greatly increase rendering time (which makes sense, as each sample (which will be large due to the high fixed\_radius) is being split into a lot of tiny clusters (because of the small cluster size). A good rule of thumb is:

fixed\_radius/cluster = sqrt(search)

This means that if you are trying for 100 photons (search = 100) then fixed\_radius divided by cluster should equal 10 (sqrt(100)=10) so if we set a cluster size of .01 then fixed\_radius should be around .1 ( $(10^*.01 = .1) = (.1/.01 = 10)$ )

## Background

Relevant to Blender v2.31

Adds a background image (environment map), colour or sky to your render.

Pathlight and Hemilight can sample the background colour and intensity to simulate real world lighting.

## Normal Image Background

```
<background type="image" name="envnorm" power="1.000000">
    <filename value="C:\directory\image.jpg"/>
</background>
```

- type image: Allows you to use a bitmap image of one of the supported texture formats as a background. The image is mapped around the scene as a sphere, so the image should be in latitude/longitude format (ratio 2:1).
- name: Background Name.
- Power: Level of brightness of the bitmap. 1.0 is default, greater numbers will increase brightness, lower numbers will decrease brightness.
- filename: Full path and filename of the image, including the file extension.

### **HDRI Background**

```
<background type = "HDRI" name = "envhdri" exposure_adjust = "0">
<filename value = "C:\directory\image.hdr" />
</background>
```

- type HDRI: Allows you to use an HDR image as a background. The HDRI is mapped around the scene as an angular map, not latitude longitude like the normal image background.
- name: Background Name.
- exposure\_adjust: Similar to 'power' for a normal image background. 0 is default, increasing this will brighten the HDR, decreasing will darken (equivalent to adjusting the f-stop on a physical camera).
- filename: Full path and filename of the HDRI, including the file extension.

### **Constant Background**

• type - constant: Lets you assign a single colour as the background.

- name: Background Name.
- color: r red value of color (0.000000 1.000000) g green value of color (0.000000 - 1.000000) g - green value of color (0.000000 - 1.000000). 1/1/1 is white, 0/0/0 is black.

### Sun/Sky Background

```
<background type="sunsky" name ="Sunl" turbidity ="4.000000" add_sun="on" sun_power="1.
    a_var="1.000000" b_var="1.000000" c_var="1.000000" d_var="1.000000" e_var="1.000000"
    <from x="-0.007401" y="8.589217" z="3.737965"/>
</background>
```

- type sunsky: Lets you assign a realistic sky background with optional sun.
- turbidity: Atmosphere density (eg mist/fog) the lower the number, the less visible the sky is. A value of 4 is a clear sky.

## Camera

```
Relevant to Blender v2.31
```

```
<camera name="Camera" resx="1024" resy="576" focal="1.015937">
    <from x="0.323759" y="-7.701275" z="2.818493"/>
    <to x="0.318982" y="-6.717273" z="2.640400"/>
    <up x="0.323330" y="-7.523182" z="3.802506"/>
</camera>
```

- Name: Name of the camera.
- resx: Horizontal resolution (rendered image width in pixels).
- resy: Vertical Resolution (rendered image height in pixels).
- res: Resolution of the shadowmap (only for volumetric shadows by now).
- focal: Field of View. Equivalent to lens length in real world camera 1.093 is roughly 35mm, 6.25 is roughly 200mm.
- from: Position of the camera.
- to: Target of the camera.
- up: Camera's 'Up Vector' defines what is considered the 'up' direction of the camera.

## Render

Relevant to Blender v2.31

Outputs an image file, based on the input from a 'Camera' block.

**Multiple views:** You can have as many camera blocks and/or render blocks within the same XML, to save out the same view, with different output settings, or several different views (cameras) at various resolutions, with different backgrounds.

- AA\_passes: Sets the number of anti-alias passes to perform. A value of 0 means no anti-aliasing is done.
- AA\_minsamples: Sets the number of samples per pass.

There are several ways to use these two parameters. You can set 'AA\_minsamples' to a certain number, and set 'AA\_passes' to 1, then after a first render pass all pixels which need it will be anti-aliased using the full number of samples set with 'AA\_minsamples'. The old method is equivalent to setting 'AA\_minsamples' to 1 and 'AA\_passes' to the number of samples, then all pixels will be continually checked if they still need extra samples, this is in fact slower for normal raytracing pictures, but can be faster when rendering with hemi- or path-light. However, due to internal limitations this doesn't work well with high sample settings. You can also combine both methods, check pixels every pass and take more samples per pass at the same time, for instance for 16 samples total, you could try setting both AA\_passes and AA\_minsamples to 4 (4 x 4 = 16).

• AA\_pixelwidth: Sets the amount of overlap of pixels used for AA.

The range is 1 to 2 (common choices are 1.5 or 2.0), the higher, the better and smoother the AA, but depending on your preference you might find the image to look a bit blurred. A value of 1.0 is equivalent to the old method.

 AA\_threshold: Sets the threshold value at which point a pixel is considered for anti-aliasing.

The range is from 0.0 (anti-alias every pixel) to 1.0 (no anti-aliasing).

Since QMC is used, settings with low sample settings like in the example above can produce quite good results nevertheless.

Anti-aliasing is also done on the alpha channel.

save\_alpha: To save a rendered targa image with the alpha channel set value 'on'.

## **Filters**

Relevant to Blender v2.31

## **Anti Noise Filter**

```
<filter type="antinoise" name="Anti Noise" radius = "1.000000" max_delta = "0.100000"> </filter>
```

- type-antinoise: Post processes the rendered image, reducing noise resulting from too few pathlight, hemilight, or conetraced samples.
- name: Name of the filter
- radius: Amount of blur to apply to the areas considered to have noise.
- max\_delta: Tolerance setting for noise. With higher values, more of the image will be considered 'noise' and will have the blur applied to them.

### **Depth of Field Filter**

```
<filter type="dof" name="dof" focus = "12.5" near_blur ="10.000000" far_blur ="10.00000" </filter>
```

- type-dof: Post processes the rendered image, using depth information to apply an out of focus effect.
- name: Name of the filter.
- focus: Distance from the camera that is in focus (objects further away and closer than this point will be out of focus).
- near\_blur: Amount to blur objects in front of the focus point.
- far\_blur: Amount to blur objects behind the focus point.
- scale: Scales the area that is in focus. Higher values will decrease the effect of depth of field as the out of focus areas are pushed away from the focus area.

The Depth of Field filter is a 2D filter, i.e. a post processing technique, and as such, has advantages and disadvantages. It uses the rendered image, plus a Z Buffer (which tells the filter how far away each pixel is from the camera) to figure out which pixels are blurred or not blurred.

Because its a 2D effect it has the advantage of being extremely quick. However there are a few disadvantages:

Reflections are not blurred correctly. If you look at a reflection, you'll notice that the reflection's blur is based on the distance from the camera of the *reflection plane*, not the *object in the reflection*.

Because the DOF is done on a 2D image, rather than a 3D scene, the blur cannot know what is behind any given object, therefore often the edges of an extremely blurred object in the foreground will look smudgy or dirty.

If you keep these limitations in mind, the Depth of Field filter can produce great looking Depth of Field effects very quickly.

## Notes

- 1. http://www.yafray.org
- 2. http://www.coala.uniovi.es/wiki/index.php/YafrayGlossaryQMC

Chapter 29. YafRay

# Glossary

## A-Z

### Active

Blender makes a distinction between *selected* and *active*. Only one Object or item can be *active* at any given time, for example to allow visualization of data in buttons.

An active object is one that is in EditMode, or is immediately switchable to EditMode (usually by **TAB**). No more than one object is active at any moment. Typically, the most recent selected object is active.

See Also: Selected.

#### Actuator

A LogicBrick that acts like a muscle of a lifeform. It can move the object, or also make a sound.

See Also: LogicBrick, Sensor, Controller.

#### Alpha

The alpha value in an image denotes opacity, used for blending and antialiasing.

#### **Ambient light**

Light that exists everywhere without any particular source. Ambient light does not cast shadows, but fills in the shadowed areas of a scene.

#### Anti-aliasing

An algorithm designed to reduce the stair-stepping artifacts that result from drawing graphic primitives on a raster grid.

### AVI

"Audio Video Interleaved". A container format for video with synchronized audio. An AVI file can contain different compressed video and audio-streams.

#### **Back-buffer**

Blender uses two buffers in which it draws the interface. This double-buffering system allows one buffer to be displayed, while drawing occurs on the back-buffer. For some applications in Blender the back-buffer is used to store color-coded selection information.

#### Bevel

Beveling removes sharp edges from an extruded object by adding additional material around the surrounding faces. Bevels are particularly useful for flying logos, and animation in general, since they reflect additional light from the corners of an object as well as from the front and sides.

#### **Bounding box**

A six-sided box drawn on the screen that represents the maximum extent of an object.

#### Bump map

A grayscale image used to give a surface the illusion of ridges or bumps. In Blender bumpmaps are called Nor-maps.

#### Channel

Some DataBlocks can be linked to a series of other DataBlocks. For example, a Material has eight *channels* to link Textures to. Each IpoBlock has a fixed number of available *channels*. These have a name (LocX, SizeZ, enz.) which indicates how they can be applied. When you add an IpoCurve to a channel, animation starts up immediately.

#### Child

Objects can be linked to each other in hierarchical groups. The Parent Object in such groups passes its transformations through to the Child Objects.

See Also: Parent.

#### Clipping

The removal, before drawing occurs, of vertices and faces which are outside the field of view.

#### Controller

A LogicBrick that acts like the brain of a lifeform. It makes decisions to activate muscles (Actuators), either using simple logic or complex Python scripts.

See Also: LogicBrick, Sensor, Python, Actuator.

#### DataBlock (or "block")

The general name for an element in Blender's Object Oriented System.

#### **Doppler effect**

The Doppler effect is the change in pitch that occurs when a sound has a velocity relative to the listener. When a sound moves towards the listener the pitch will rise. When going away from the listener the pitch will drop. A well known example is the sound of an ambulance passing by.

#### **Double-buffer**

Blender uses two buffers (images) to draw the interface in. The content of one buffer is displayed, while drawing occurs on the other buffer. When drawing is complete, the buffers are switched.
# EditMode

The mode for making intra-object graphical changes. Blender has two modes for making changes graphically. EditMode allows intra-object changes (moving, scaling rotating, deleting, and other operations on selected vertices of the active object). By contrast, ObjectMode allows inter-object changes (operations on selected objects).

Switch between EditMode and ObjectMode with Hotkey: TAB.

See Also: ObjectMode, Vertex (pl. vertices).

#### Extend select

Adds new selected items to the current selection (SHIFT-RMB).

## Extrusion

The creation of a three-dimensional object by pushing out a two-dimensional outline and giving it height, like a cookie-cutter. It is often used to create 3D text.

#### Face

The triangle and square polygons that form the basis for Meshes or for rendering.

#### Field

Frames from videos in NTSC or PAL format are composed of two interlaced fields.

#### FaceSelectMode

Mode to select faces on an object. Most important for texturing objects. Hotkey: **FKEY**.

# Flag

A programming term for a variable that indicates a certain status.

#### Flat shading

A fast rendering algorithm that simply gives each facet of an object a single color. It yields a solid representation of objects without taking a long time to render. Pressing **ZKEY** switches to flat shading in Blender.

# Fps

Frames per second. All animations, video, and movies are played at a certain rate. Above ca. 15fps the human eye cannot see the single frames and is tricked into seeing a fluid motion. In games this is used as an indicator of how fast a game runs.

# Frame

A single picture taken from an animation or video.

# **Gouraud shading**

A rendering algorithm that provides more detail. It averages color information from adjacent faces to create colors. It is more realistic than flat shading, but less realistic than Phong shading or ray-tracing. The hotkey in Blender is **CTRL-Z**.

#### **Graphical User Interface**

The whole part of an interactive application which requests input from the user (keyboard, mouse etc.) and displays this information to the user. Blender's GUI is designed for an efficient modelling process in an animation company where time equals money. Blender's whole GUI is done in OpenGL.

See Also: OpenGL.

# Hierarchy

Objects can be linked to each other in hierarchical groups. The Parent Object in such groups passes its transformations through to the Child Objects.

#### lpo

The main animation curve system. Ipo blocks can be used by Objects for movement, and also by Materials for animated colors.

## **IpoCurve**

The Ipo animation curve.

## Item

The general name for a selectable element, e.g. Objects, vertices or curves.

## Lathe

A lathe object is created by rotating a two-dimensional shape around a central axis. It is convenient for creating 3D objects like glasses, vases, and bowls. In Blender this is called "spinning".

# Keyframe

A frame in a sequence that specifies all of the attributes of an object. The object can then be changed in any way and a second keyframe defined. Blender automatically creates a series of transition frames between the two keyframes, a process called "tweening."

# Layer

A visibility flag for Objects, Scenes and 3DWindows. This is a very efficient method for testing Object visibility.

# Link

The reference from one DataBlock to another. It is a "pointer" in programming terminology.

#### Local

Each Object in Blender defines a *local* 3D space, bound by its location, rotation and size. Objects themselves reside in the *global* 3D space.

A DataBlock is *local*, when it is read from the current Blender file. Non-local blocks (library blocks) are linked parts from other Blender files.

#### LogicBrick

A graphical representation of a functional unit in Blender's game logic. LogicBricks can be Sensors, Controllers or Actuators.

See Also: Sensor, Controller, Actuator.

#### Mapping

The relationship between a Material and a Texture is called the 'mapping'. This relationship is two-sided. First, the information that is passed on to the Texture must be specified. Then the effect of the Texture on the Material is specified.

## Mipmap

Process to filter and speed up the display of textures.

## MPEG-I

Video compression standard by the "Motion Pictures Expert Group". Due to its small size and platform independence, it is ideal for distributing video files over the internet.

#### **ObData block**

The first and most important DataBlock linked by an Object. This block defines the Object *type*, e.g. Mesh or Curve or Lamp.

#### Object

The basic 3D information block. It contains a position, rotation, size and transformation matrices. It can be linked to other Objects for hierarchies or deformation. Objects can be "empty" (just an axis) or have a link to ObData, the actual 3D information: Mesh, Curve, Lattice, Lamp, etc.

# ObjectMode

The mode for making inter-object graphical changes. Blender has two modes for making changes graphically. ObjectMode allows inter-object changes (moving, scaling rotating, deleting and other operations on selected objects). By contrast, EditMode allows intra-object changes (operations on selected vertices in the active object).

Switch between ObjectMode and EditMode with Hotkey: TAB.

See Also: EditMode.

# OpenGL

OpenGL is a programming interface mainly for 3D applications. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Blender's whole interface and 3D output in the real-time and interactive 3D graphic is done by OpenGL.

#### **Orthographic view**

An orthographic view of an object makes it appear flat and two-dimensional, like a plan or an elevation. All the points of the object are perpendicular to the viewing plane, and are projected in parallel.

See Also: Perspective view.

## Oversampling

See: Anti-aliasing

#### Overscan

Video images generally exceed the size of the physical screen. The edge of the picture may or may not be displayed, to allow variations in television sets. The extra area is called the overscan area. Video productions are planned so critical action only occurs in the center safe title area. Professional monitors are capable of displaying the entire video image including the overscan area.

#### Parent

An object that is linked to another object, the parent is linked to a child in a parent-child relationship. A parent object's coordinates become the center of the world for any of its child objects.

See Also: Child.

#### **Perspective view**

In a perspective view, the further an object is from the viewer, the smaller it appears.

See Also: Orthographic view.

#### Pivot

A point that normally lies at an object's geometric center. An object's position and rotation are calculated in relation to its pivot-point. However, an object can be moved off its center point, allowing it to rotate around a point that lies outside the object.

# Pixel

A single dot of light on the computer screen; the smallest unit of a computer graphic. Short for "picture element."

# Plug-In

A piece of (C-)code loadable during runtime. This way it is possible to extend the functionality of Blender without a need for recompiling. The Blender plugin for showing 3D content in other applications is such a piece of code.

## Python

The scripting language integrated into Blender. Python<sup>1</sup> is an interpreted, interactive, object-oriented programming language.

# Quaternions

Instead of using a three-component Euler angle, quaternions use a four-component vector. It is generally difficult to describe the relationships of these quaternion channels to the resulting orientation, but it is often not necessary. It is best to generate quaternion keyframes by manipulating the bones directly, only editing the specific curves to adjust lead-in and lead-out transitions.

#### Render

To create a two-dimensional representation of an object based on its shape and surface properties (i.e. a picture for print or to display on the monitor).

#### **Rigid Body**

Option for dynamic objects in Blender which causes the game engine to take the shape of the body into account. This can be used to create rolling spheres for example.

# Selected

Blender makes a distinction between *selected* and *active* objects. Any number of objects can be *selected* at once. Almost all key commands have an effect on *selected* objects. Selecting is done with the right mouse button.

See Also: Active, Extend select.

#### Sensor

A LogicBrick that acts like a sense of a lifeform. It reacts to touch, vision, collision etc.

See Also: LogicBrick, Controller, Actuator.

#### Single User

DataBlocks with only one user.

# Smoothing

A rendering procedure that performs vertex-normal interpolation across a face before lighting calculations begin. The individual facets are then no longer visible.

#### Transform

Change a location, rotation, or size. Usually applied to Objects or vertices.

#### Transparency

A surface property that determines how much light passes through an object without being altered.

See Also: Alpha.

# User

When one DataBlock references another DataBlock, it has a user.

#### Vertex (pl. vertices)

The general name for a 3D or 2D point. Besides an X,Y,Z coordinate, a vertex can have color, a normal vector and a selection flag. Also used as controlling points or handles on curves.

# Vertex array

A special and fast way to display 3D on the screen using the hardware graphic acceleration. However, some OpenGL drivers or hardware doesn't support this, so it can be switched off in the InfoWindow.

# Wireframe

A representation of a three-dimensional object that only shows the lines of its contours, hence the name "wireframe."

# X, Y, Z axes

The three axes of the world's three-dimensional coordinate system. In the FrontView, the X axis is an imaginary horizontal line running from left to right; the Z axis is a vertical line; and Y axis is a line that comes out of the screen toward you. In general, any movement parallel to one of these axes is said to be movement along that axis.

## X, Y, and Z coordinates

The X coordinate of an object is measured by drawing a line that is perpendicular to the X axis, through its centerpoint. The distance from where that line intersects the X axis to the zero point of the X axis is the object's X coordinate. The Y and Z coordinates are measured in a similar manner.

# Z-buffer

For a Z-buffer image, each pixel is associated with a Z-value, derived from the distance in 'eye space' from the Camera. Before each pixel of a polygon is drawn, the existing Z-buffer value is compared to the Z-value of the polygon at that point. It is a common and fast visible-surface algorithm.

# Notes

1. http://www.python.org/

Glossary