Community Experience Distilled

Blender 3D 2.49
Architecture, Buildings, and Scenery

Create photo-realistic 3D architectural visualizations of buildings, interiors, and environmental scenery with Blender

Allan Brito

[PACKT] PUBLISHING open source* community experience distilled

Blender 3D 2.49

Architecture, Buildings, and Scenery

Create photo-realistic 3D architectural visualizations of buildings, interiors, and environmental scenery with Blender

Allan Brito

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Blender 3D 2.49
Architecture, Buildings, and Scenery

First published: August 2010

Production Reference: 2270810

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-849510-48-6

www.packtpub.com

Cover Image by Allan Brito (allanrbs@gmail.com)

Credits

Author
Allan Brito

Reviewers
Ira Krakow
Jonathan Williamson

Acquisition Editor
David Barnes

Development Editor
Swapna Verlekar

Technical Editor
Dayan Hyames

Copy Editor
Janki Mathuria

Indexer
Monica Ajmera Mehta

Editorial Team Leader
Akshara Aware

Project Team Leader
Lata Basantani

Project Coordinator
Shubhanjan Chatterjee

Proofreader
Clyde Jenkins

Graphics
Geetanjali Sawant

Production Coordinator
Aparna Bhagat

Cover Work
Aparna Bhagat

About the Author

Allan Brito is a Brazilian architect specialized in information visualization, who lives and works in Recife, Brazil. He works with Blender 3D to produce animations and still images for visualization and instructional material.

In addition to his work with Blender as an artist, he also has substantial experience in teaching and researching 3D modeling, animation, and multimedia.

He is an active member of the community of Blender users, writing about Blender 3D and its development for web sites in Brazilian Portuguese (`http://www.allanbrito.com`) and English (`http://www.blendernation.com`).

This is his second book about Blender; the first one was Blender 3D – Guia do Usuário, which was published in Brazil. It's a guide on how to use Blender, covering the basics of the tool and more advanced topics like character animation.

He can be reached through his website at `http://www.blender3darchitect.com`, where he covers the use of Blender 3D and other tools, for architectural visualization.

I would like to thank my family for supporting me during the production of this book, especially my wife Érica and my parents Maria and Luiz.

About the Reviewers

Ira Krakow is a Blender 3D trainer, author, and consultant. He is particularly interested in promoting Blender 3D in elementary and secondary schools to help unleash students' creativity and knowledge in artistic and scientific areas. He has over 35 years of experience in software and database development, support, and training.

You can find Ira's Blender 3D tutorials at `http://www.youtube.com/irakrakow`, and you can visit his Blender 3D Forum at `http://forum.irakrakow.com`. Ira can be reached at `ira.Krakow@gmail.com`

Ira would like to acknowledge the outstanding support of the Blender 3D community to all Blender users, from newcomers to advanced animation professionals. He would especially like to thank Kernon Dillon at `http://blendernewbies.blogspot.com`, and all the dedicated Blender users who contribute to `http://www.blenderartists.org` (BlenderArtists).

Jonathan Williamson is a young artist and educator in his twenties, who has a deep passion for art and creativity. More importantly, he has a passion to create and share those creations. Since starting in Blender when he was fourteen, Jonathan has produced a number of tutorials and training resources for Blender, through `http://montagestudio.org` and `http://blendercookie.com`, in an effort to give back to the community that drives it.

Jonathan grew up near a small apple orchard in central Kansas, USA. As a home schooler, and always being surrounded by family, art, and nature, he was instilled with an urge to learn and understand the world around him from a young age. He continues to fuel that urge today, committing himself to being a life-long student.

Throughout the reviewing process of this book, I have had the endless support of many different people and would like to give a brief thanks to them. My brother, more than anyone, has always been there to support and encourage me, as have the rest of my family. My dearest thanks go out to them. I would also like to give a warm thank you to all of my friends. I wouldn't be where I am today without each and every one of you.

Table of Contents

Preface

This book will show you how to generate realistic architectural models quickly using Blender. Blender 3D is an open source 3D graphics suite, capable of modeling, rendering, and animating 3D environments. You can create natural scenery, landscapes, plants, various weather conditions, environmental factors, building materials such as wood, metal, brick, and more, using blender.

What this book covers

Chapter 1, *Introduction to Blender and Architectural Visualization* covers the role of Blender as a tool for architectural visualization artists, and how it can help them to achieve their goals.

Chapter 2, *Blender 3D: Quick Start* covers the basics of Blender and how to use the most important aspects of the user interface.

Chapter 3, *Modeling* shows and guides the reader through the poly-modeling tools of Blender and how to use them to create 3D geometry.

Chapter 4, *Modeling for Architecture* extends the knowledge acquired in Chapter 3, showing examples of architectural modeling with Blender.

Chapter 5, *Modeling Details* deals with small, but important details that could trick an architectural visualization artist.

Chapter 6, *Modeling Furniture* covers how to create 3D furniture to populate the projects created for interior design.

Chapter 7 Materials demonstrates how to add realism to the 3D models using materials and how to create glass, mirrors, and other surfaces.

Chapter 8, *Textures* follows the tips from Chapter 7 to give materials an even better look and realism with bitmap-based textures to create tiles, wood floors, and much more.

Chapter 9, *UV Mapping* is a powerful way to get full control over textures, and this is the subject of this chapter.

Chapter 10, *Light Basics* covers how to use Blender's default light sources and organize them to create better light designs for our visualizations.

Chapter 11, *Radiosity and Ambient Occlusion* adds the power of a global illumination effect to all the projects and raises the quality of the light for all renders.

Chapter 12, *Global Illumination with YafaRay* is about how to install and use advanced GI algorithms to add incredible realism to the light and render of all projects created with Blender.

Chapter 13, *Animation for Architectural Visualization* helps architectural visualization artists to create short films and even interactive animations to show their projects.

Chapter 14, *Post-Production with GIMP* helps users to edit and enhance the images rendered with Blender.

Chapter 15, *Blender 2.50 and Architectural Visualization* is about an expected upgrade for Blender that will change some aspects of the user interface. In this chapter, you will learn some of those aspects and about the most important differences from Blender 2.49.

What you need for this book

Blender 2.49 and YafaRay 0.1.1

Who this book is for

This book is for architects, game designers, artists, or movie makers who want to create realistic buildings, interiors, and scenery using Blender 3D—a free, open source graphics tool. This book is not a general introduction to Blender, but focuses on developing expertise on the architectural aspects of the tool. Readers need not have prior knowledge of Blender.

Conventions

In this book, you will find a number of styles of text that distinguish among different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "There is another way of renaming objects with a small menu called **Transform Properties**".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com`, or e-mail `suggest@packtpub.com`.

If there is a topic in which you have expertise, and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the color images for the book

The printed version of the book is in black and white, but a full color version of the images is available for download at `http://www.PacktPub.com`. If you purchased this book elsewhere, you can visit `http://www.PacktPub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted, and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately, so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Introduction to Blender and Architectural Visualization

As you know, every type of construction, such as building a house, movie set, or virtual set needs a project. These projects are made up of a lot of documents and technical drawings that help in the construction of those buildings and virtual environments, which will be used to show something that doesn't physically exist. For the construction crew of a building, like carpenters and engineers, these technical drawings and documents are just fine; but when you need to make a presentation of these projects for people who can't read technical drawings, things can get a little difficult.

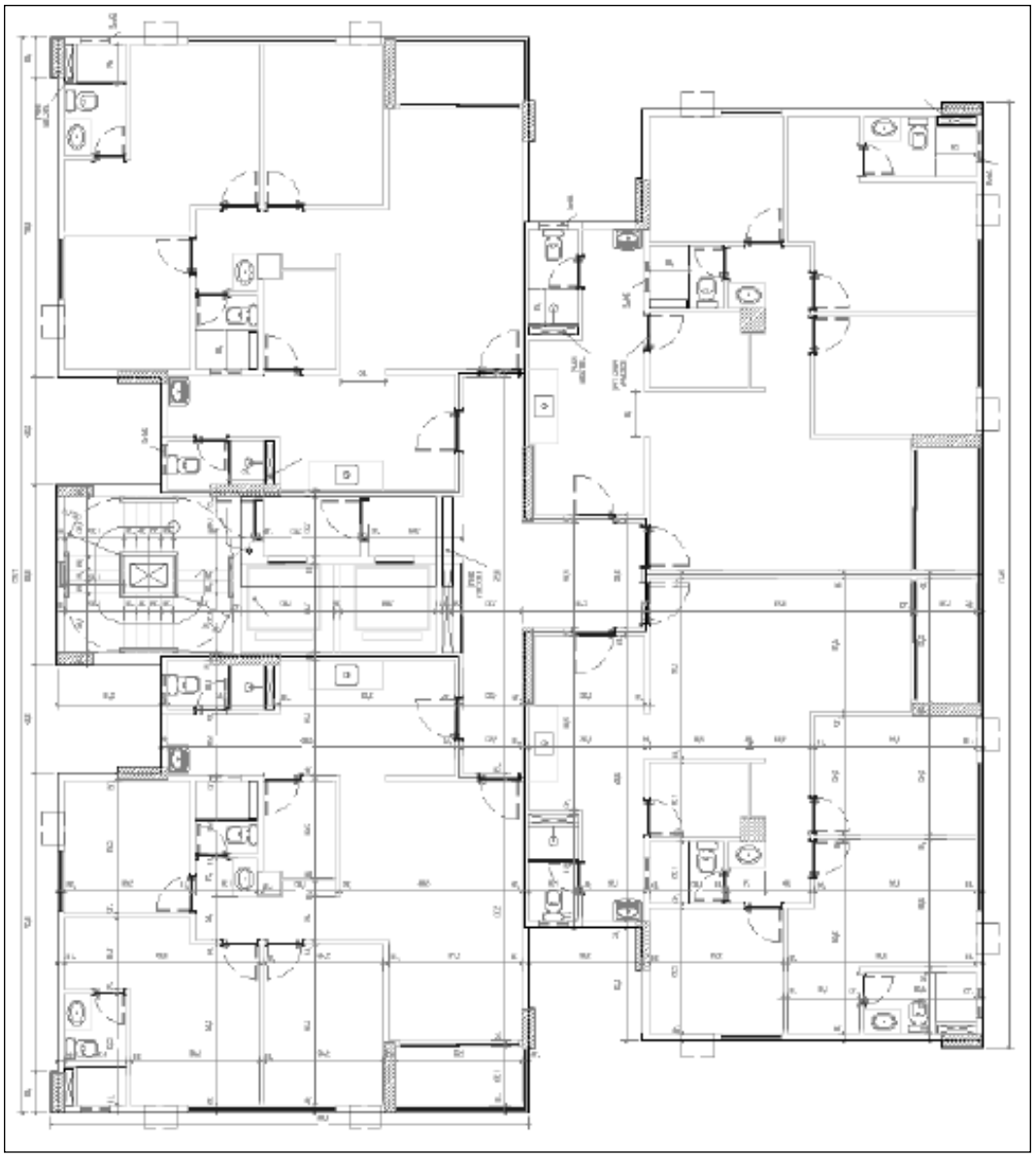
Architectural visualization

The traditional way to show architectural projects is with images that look like a photograph of the project that could be made by hand and painted with watercolor or airbrushes. Another option to create those images is with the use of computers, which we will be doing in this book.

One of the most common types of images used to represent architectural projects are perspective views, which are projections of the construction with one or more vanishing points, like the following image:

It's far easier to understand a picture of a building or environment, than to make decisions based on the reading of a technical drawing, like the next image demonstrates. These kinds of presentations look really great, but they are expensive to create and require a long production time for each view. That's where the computer-generated architectural visualization makes everything easier for everyone involved with the project.

The benefits of using computer-generated visualization for architecture quickly made it a standard for these kinds of presentations. By visualization, we mean the set of techniques used to represent something that doesn't exist yet visually. The traditional visualization techniques, like physical models, take too long to be produced; by the time it's finished, we can't change much on the model. In traditional drawing and painting, like watercolor or airbrush, the artist can't use the same drawing to generate other views. This makes the techniques expensive and time-consuming.

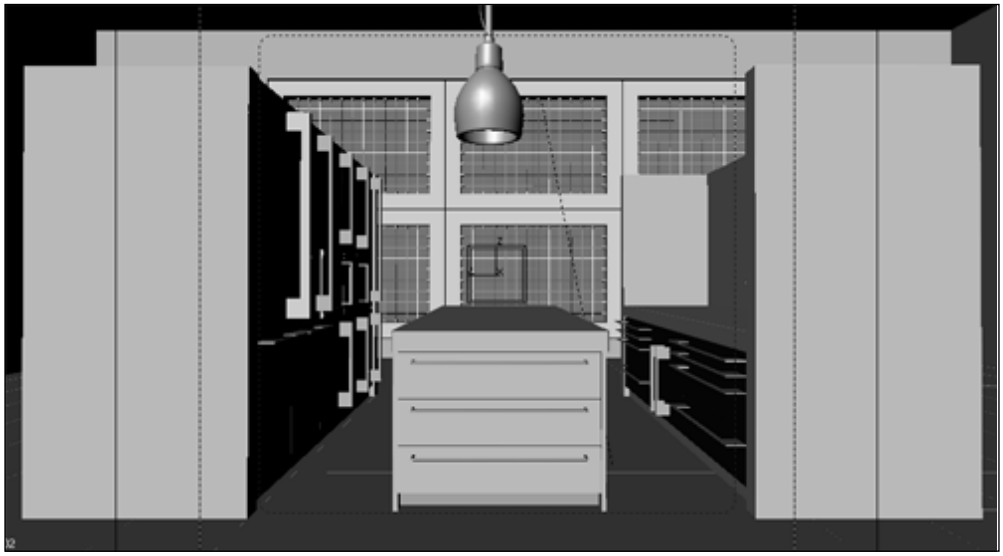


With computer-generated visualization, we have the ability to change the 3D model at any time, and generate several views from the same 3D model with a simple camera move. The use of 3D models brings more options, even in the project stage, because it's possible to visualize all the environments and parts quickly while it's being planned, and make changes to improve the organization and overlook every aspect of the project.

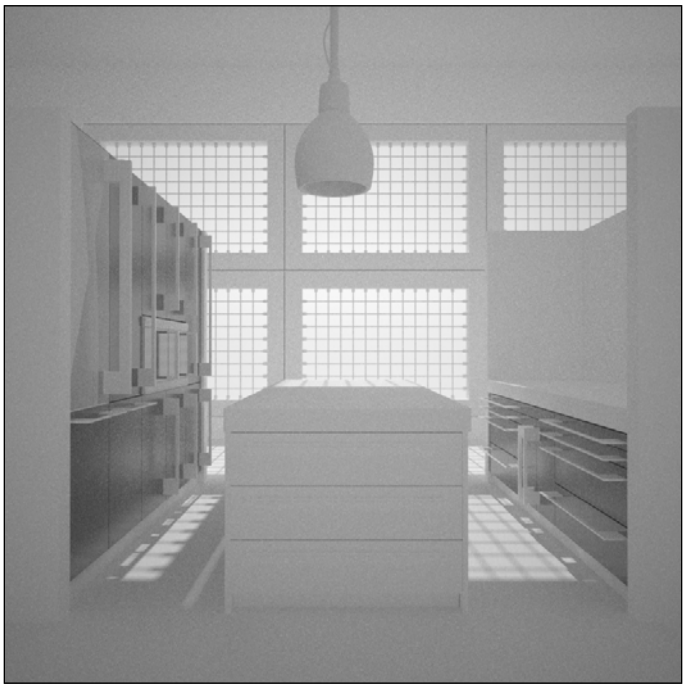
Today almost every project for buildings, sets, or anything involving construction has a 3D visualization for project development or to show the concept to someone who wouldn't understand a presentation based on technical drawings. Even if it's only the benefit of being faster and cheaper to produce, the computer-generated architectural visualization has more benefits. And it's one that can't be beaten by traditional visualization like watercolor. This feature is animation, or the production of a small video showing the visualization with a moving camera, just as if we were walking inside the project. With animation, the project can be presented in a much richer environment than on paper, and it can't be reproduced by traditional artwork.

One of the keys to create a good visualization for architecture is to achieve photorealism. This occurs when the artist is able to simulate the environment and all objects on the scene, including materials and textures, in a way that gets really close to reality. For 3D visualization with a clear objective of selling a project, the creation of a photorealistic image may be the key to sell an idea—in this case a project. We have a comparison between a regular 3D visualization with no special lights or materials and a photorealistic image of the same scene.

The regular 3D realization:



The photorealistic image of the same scene:



How about Blender 3D?

But what's Blender 3D, and how is it related to architectural visualization? Blender 3D is an open source 3D graphics suite, which models, renders, and animates 3D environments. Blender can be developed by anyone with good knowledge of programming and computer graphics, and the willingness to work. By being open source, anyone can download and start to use it immediately in commercial projects, because there aren't any costs related to the download of Blender. It's not shareware with limited tools or time constraints, and you can use it freely. In the last year, the Blender user base has grown significantly, with a total of approximately 1.5 million downloads. Everyday, more students and professionals switch to Blender, as it's tools get better in each new release, with the addition of new sculpting tools, advanced rendering, support from commercial render systems, and more.

One of the aspects that calls attention to Blender is it's size, of only about 15 MB. That's right! Only 15 MB, and we can even run it directly from a portable drive. Another great aspect of Blender is that we can use various operating systems, such as Linux, Microsoft Windows, and Mac OS X, leaving the choice to the user.

When we start to use Blender, we will notice that even being about 15 MB in size, it does not mean lack of power when we start to model and render. A lot of quality work has been done with Blender in the last few months, like the first open movies made entirely with Blender, named *Elephants Dream and Big Buck Bunny.*

Elephants Dream and Big Buck Bunny

If you want to download the movies, along with the production files, visit `http:// www.elephantsdream.org` and `http://www.bigbuckbunny.org`.

As we go through the book, we will see that Blender is a tool designed to give artists high productivity and fast access to tools and menus. This means that Blender is strongly based on keyboard shortcuts and not menus, which can help to create 3D models really fast for 3D visualization. For advanced users, this is great, but it can be hard for new users. But don't worry, with some practice and the examples that we will cover along the book, it will be possible to understand quickly the most important aspects and tricks involving 3D modeling and animation to create great architectural visualization and scenarios with Blender.

Hardware and software requirements for Blender

Blender doesn't require a powerful hardware setup for anyone who just wants to start using it. Is there anything special about the hardware needed for architectural visualization? Well, if you want to produce photorealistic renderings, then I strongly recommend you to upgrade your system with more RAM and CPU power, because these kinds of rendering require a lot of processing. But if you want to create renderings that look more like a sketch or anything that doesn't look photorealistic, it won't be necessary to use a powerful computer, because this kind of rendering demands less resources.

The Blender Foundation recommends these minimum requirements:

- 3-button mouse
- OpenGL graphics card with 16 MB RAM
- 300 MHz CPU
- 128 MB RAM
- 1024 x 768 pixels display with 16-bit color
- 20 MB free hard disk space

However, there is more. If you really want to get maximum performance, there is a more powerful configuration:

- 2 GHz quad core CPU
- 2 GB RAM
- 1920 x 1200 pixels display with 24-bit color
- 3-button mouse
- OpenGL graphics card with 128 or 256 MB RAM

As for the software, you can run Blender on almost any operating system available. Here is the list of systems that support Blender:

- Microsoft Windows XP, Vista or Windows 7
- Mac OS X 10.3 and later
- Linux
- Irix 6.5 MIPS3

Other tools for visualization

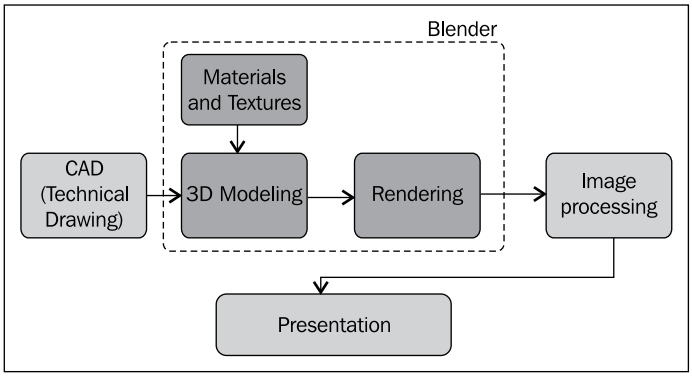
In spite of being a very powerful 3D graphics suite, Blender can't handle all the processes of creating architectural visualization alone. We will need some extra tools like Gimp, for post-processing and image editing. There are some tasks, like texture editing and creation, that need a more specialized tool, and for that, Gimp is the best choice to work together with Blender.

Another great tool that we will be using is YafaRay, which can make awesome renders using a global illumination engine that helps Blender to create photorealistic images. The integration between Blender and YafaRay is really great, with a plugin that allows us to export 3D models from Blender directly to YafaRay for rendering.

The visualization workflow, shown in the next image, requires the use of a whole set of tools. Blender is just one of them, but we could say that it's the tool responsible for the creation of the images and animations. Along with Blender, we could name some other tools that can help a visualization artist to create a good presentation:

- **CAD**: Everything starts with a CAD file, which has the technical drawings for a project. Here is a small list of some tools used for CAD drawings: QCad, VariCAD, AutoCAD, VectorWorks, DoubleCAD XT, and ArchiCAD.
- **3D Modeling and Rendering**: Here is where Blender is used.

- **Image processing**: After generating the images with Blender, we may need to make adjustments or corrections to the files. We could use tools like Gimp or Photoshop to make these adjustments.
- **Presentation**: After the editing process, we can use some other tool to make a presentation. If we choose to print the perspectives, a good choice is to use Inkscape to make a sheet or folder. Another option is to use a slideshow to present the perspectives; this can be done with OpenOffice.org Impress.



CAD and 3D architectural modeling

To work with architectural visualization, you will need to understand how Blender deals with files from CAD software like AutoCAD, ArchiCAD, QCad, and other tools. The reason to work with CAD files for architectural visualization is that sometimes it is easier to start a project based on the dimensions and information contained in a technical drawing. For instance, we can use the representation of the walls of a building in a CAD file, and use it to create 3D walls with the correct measurements without the need to pay much attention to dimensions. The CAD file already contains the correct proportions.

The most common file format used to exchange CAD drawings is DXF, which means Drawing Exchange Format. Therefore, if your CAD software can save your drawings in the DXF file format, Blender will be able to import it. Because most CAD packages can do that, it makes Blender compatible with them. Another common file format to use is 3DS, from the old 3D Studio Max.

Another important source of 3D models is Google SketchUp, which can export models to Blender 3D using a file format named COLLADA. An artist can work on a model using SketchUp to start a project, and finish the visualization with realistic textures, and rendering using Blender 3D.

Improving COLLADA support

There is an ongoing project to improve the ability of Blender 3D to read COLLADA files, which will enable Blender to fully read any file exported from SketchUp. Not only will files exported from SketchUp be readable, but all 3D models available at the 3D Warehouse—a portal with thousands of free models to download, from furniture models to full 3D buildings—will also be readable. This project is part of the Google Summer of Code 2009 and may be released with Blender 2.50 in the middle of 2010.

3D models from the Internet

Creating architectural visualizations and scenarios with computers may be a very quick task, if you only need to model and work on walls, floors and other basic elements. What can really change the look and feel of your scenes are the details. The secret for a good scene is the amount of details and objects that it contains. A good visualization for an office space is filled with desks, computers, chairs, and objects placed over the desks. But don't worry, we won't have to model all those objects every time. For that, we have to create a good library with objects, which we will be able to use for our scenes. This includes cars, people, vegetation, and all other objects that can be interesting.

The easiest way to gather all these files are from the Internet; places like 3D Cafe (`http://www.3dcafe.com`), which allows anyone to download 3D models for free.

Here is a list of places to find models for Blender:

- `http://resources.blogscopia.com` — furniture models in the native Blender file format.
- `http://www.e-interiors.net` — lots of pictures and free models of furniture. Most files are in 3DS or DXF file formats.
- `http://www.linedstudio.com` — more furniture models and scenes already in Blender native file format.
- `http://blender-archi.tuxfamily.org/Models` — collection of models to use in Blender for architectural visualization. All are in the Blender native file format.

Visualization with Blender

If you want to find some good examples of architectural visualization made with Blender, there are some websites that you can visit; most of them are related to some external rendering engine that can be integrated with Blender. But their communities of artists create great examples of what Blender can do for visualization.

For this book, we chose to use YafaRay, which is the external renderer that best integrates with Blender. But this is not the only render engine used to create realistic images with Blender 3D. We could use other great external renderers as well:

- Indigo renderer–`http://www.indigorenderer.com`
- Kerkythea—`http://www.kerkythea.net`
- Sunflow—`http://sunflow.sourceforge.net`
- POV-Ray—`http://www.povray.org`
- YafaRay—`http://www.yafaray.org`
- LuxRender—`http://www.luxrender.net`

From all those renderers, we could easily pick two of them to use in architectural visualization projects, which are YafaRay and LuxRender. Both of them have great and powerful tools to create realistic images. The only proprietary renderer from the list is Indigo Renderer, which still offers a free version to use, but with limitations on the render resolution, and all images generated with it have a watermark.

Take a look into the gallery of those renderers, and you will find some great examples of architectural visualization made with Blender. And, we can't forget about the Blender Gallery (`http://www.blender.org/features-gallery/gallery/art-gallery`) as well, which is updated on a monthly basis. This gallery has images from Blender, and almost every month some great visualization images hit the gallery.

Summary

In this chapter, we took a glance at architectural visualization and Blender, and some other techniques and assets that we will need later in the book. Here is what we have learned:

- What architectural visualization is
- What the process of architectural visualization is
- The workflow of architectural visualization and how Blender can make it easier

2
Blender 3D: Quick Start

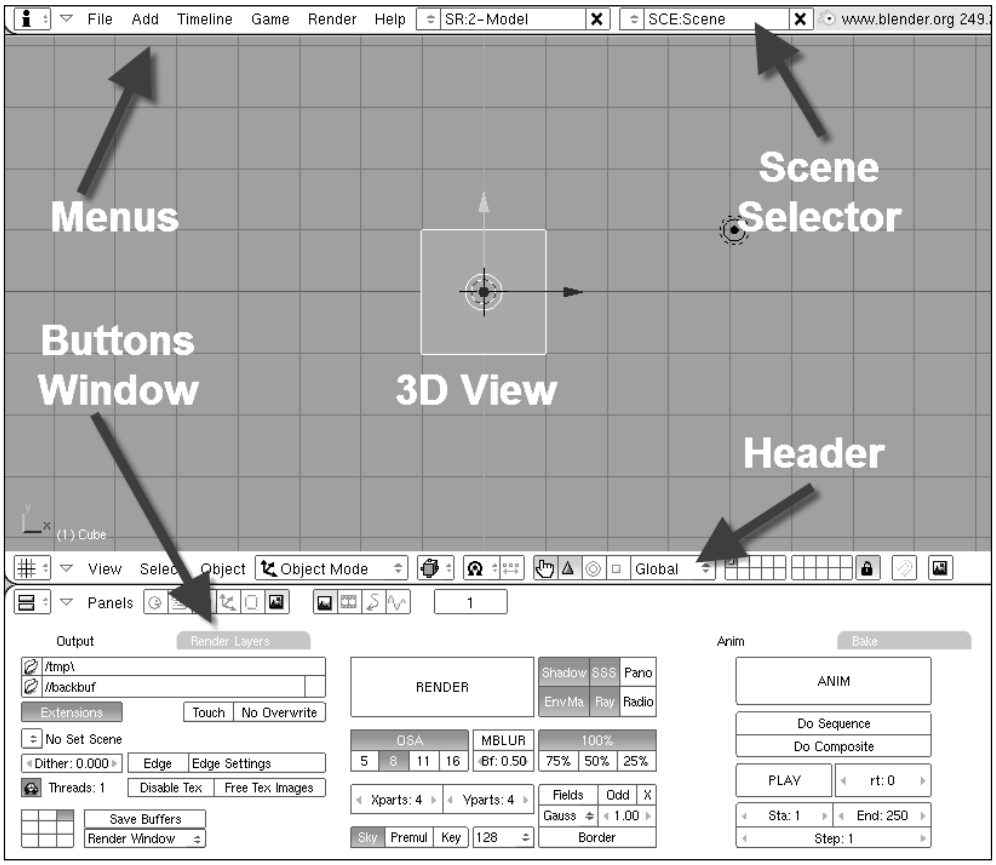
This chapter will deal with all the basic aspects related to Blender, which will be useful for artists who have never had the chance to use Blender. We must learn the basics first before we get into more specific questions about modeling and rendering architectural visualizations and scenarios. It's very important to understand how object manipulation, creation, and editing works in Blender. This way, we will be able to work a lot faster and create better models and visualizations.

If you have already worked with Blender before, you might want to skip this chapter and move on to the next one.

Interface

One of the most important parts of any software is the interface, and with Blender, it is no different. But the Blender interface is unique, because it's all based on OpenGL graphics built in real-time that can be redesigned any way we want. Because of that, we can say that Blender has a default interface that can be customized any way we want. It's even possible to zoom all the items in menus and buttons.

Let's take a look at the interface:

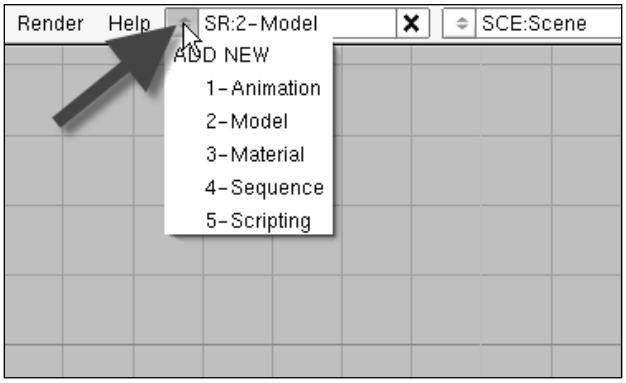


The default interface of Blender is divided into:

- **3D View**: This is the section of the interface where you visualize all your objects and manipulate them. If you are in the modeling process, this window should always be visible.
- **Buttons Window**: Here we will find almost all the tools and menus, with options to set up features like modifiers, materials, textures, and lights. We can change the options available in this window with several small icons that change the buttons with specific tasks like materials, shading, editing, and others. Those buttons will reflect the active panel in Blender, for example, when we choose materials (*F5* key). The Buttons window will then only show options related to materials.

- **Header**: All windows in Blender have a header, even if it's not visible at the time we create the window. The content of the header can change, depending on the window type. For example, in the header for the 3D View, we find options related to visualization, object manipulation, and selection.
- **Menus**: These menus work just like in any other application, with options to save files, import, and export models. Depending on the window type selected, the contents of the menu may differ.
- **Scene Selector**: We can create various scenes in Blender, and this selector allows us to choose and create these scenes. Because we will be modeling and dealing with scenery, the Scene selector will be an important tool for us.

These parts make up the default interface of Blender, but we can change all aspects of the interface. There are even some modified screens, adapted to some common tasks with Blender, for us to choose. To access these modified screen sets, we must click on the selector located to the left of **Scene Selector**:

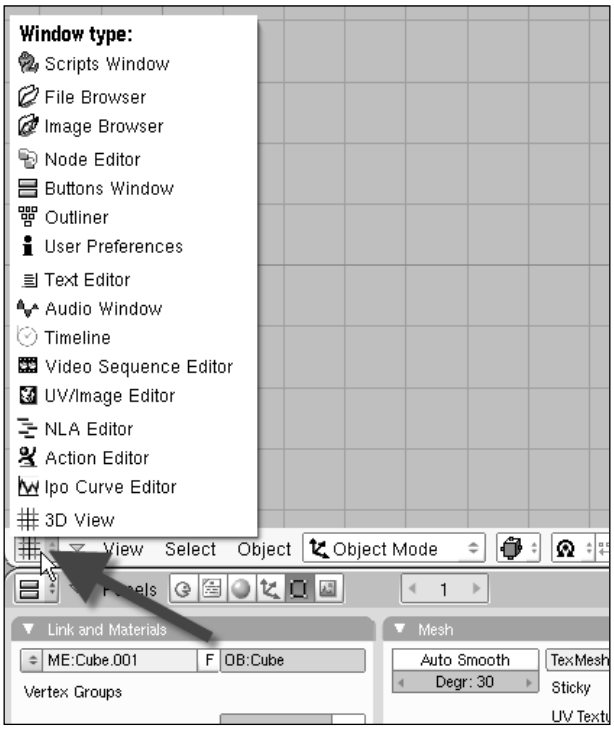


There are screen sets prepared to work with Animation, Model, Material, Sequence, and Scripting. Each of these sets has a different interface organization, optimized for its specific task. A nice way to switch between these sets is with a keyboard shortcut, which is *Ctrl* plus left arrow or right arrow. Try this shortcut, and you will switch between sets very quickly.

If you make any changes in the interface of Blender and want to overwrite the default interface, just press *Ctrl* + *U*, and your current interface will become the new default. In this way, every time Blender is started, your new interface will be shown. The same option can be reached in the **File** menu with the option named **Save Default Settings**. To restore the standard default interface, just use the option **Load Factory Settings** in the **File** menu.

Windows and menus

Blender has a lot of different windows that can do a lot of nice things. Two of the most common windows are the 3D View and the Buttons Window, but there are a lot more. With the **Window type** selector, we can choose among several types, such as **File Browser**, **Text Editor**, **Timeline**, and others. The **Window type** selector is always located in the left corner of each window, as shown in the following screenshot:



Let's see what the function of each window is:

- **Scripts Window**: This window groups some nice scripts written in Python to add some extra tools and functionalities to Blender. It works much like plugins in other 3D Packages. There are scripts to help in a lot of different tasks like modeling, animation, and importing models. Some of these scripts are very helpful to architectural modeling such as Geom Tool and Bridge Faces. For instance, we can create a city space with only a few mouse clicks using a script named Discombobulator. In most cases, the scripts will appear in the right place in the Blender menus. Use this window only if you want to browse all scripts available in your Blender Scripts folder. To run a script, just select any script from the Scripts menu.

- **File Browser**: With this window, we can browse the files of a specific folder. This window appears automatically when we save or open a file.
- **Image Browser**: Here we can view the image files in a specific folder. This window is very useful to search for image files like .jpg, .png, and others.
- **Node Editor and Compositing Nodes**: With this window, it's possible to build node sets and create complex materials and textures.
- **Buttons Window**: We already have talked about this window, but it's nice to remember that after the 3D View, this is one of the most important windows, because here we can set options for almost any tool or functionality in Blender. This is the window responsible for several tools and functions in Blender, such as lights, materials, textures, and object properties.
- **Outliner**: This window shows us a list of the objects in your scene, and lists the relations among them. Here we can see if an object is related to some other object in a hierarchical way. In this window, we can easily hide and select objects, which may be useful for complex scenes.
- **User Preferences**: As the name suggests, here we can change Blender configurations, such as file paths, themes, Auto Save, and other options.
- **Text Editor**: This window allows us to write and open text files to make comments and place notes. We can open and run Python scripts here also.
- **Audio Window**: Here we can open and compose audio files in sequences. It works much like the Video Sequence Editor, but for audio files.
- **Timeline**: That's the place where we create animation. This window gives us nice tools to add key frames and build animations.
- **Video Sequence Editor**: Here we can build and composite images and video files. It's a very nice window that can replace a video editor in some ways. We can easily create a complex animation with a lot of shots and sequence them together with this window. And, we can use the Node Editor to create complex compositions and effects.
- **UV/Image Editor**: With this window, we can edit and set up images and textures. There is even a paint application, with which we can edit and make adjustments in textures and maps. This is a very important window for us, because a lot of the texture work we will be using will involve the use of UV Textures that require a lot of adjustments in the UV/Image Editor.

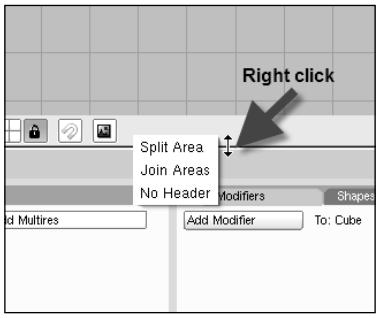
- **NLA Editor**: Here we can visualize and set up non-linear animations. This window is related more to animations and key frame visualization. A non-linear animation means that we can create small blocks of motions, which can be arranged any way we like, including copying and positioning those blocks into sequences. In Blender, these blocks are named strips. Because it's a non-linear editor, we can erase and rearrange the blocks without a break in the animation. For a linear animation system, any changes at the beginning of the animation would demand a full reconstruction of the animation from the artist.
- **Action Editor**: This window has nice options to set up actions related to character animation.
- **Ipo Curve Editor**: In this window, we can create and set up animations in a more visual way with curves. It's possible to add, edit, and delete key frames. Even for animations that don't require much work with characters and object deformations, like the ones we will be creating, it still requires a lot of work in the setup of curves to create good animation.

Now we know what each of those windows do. Some of them will be very important for your visualization tasks, such as the Buttons and Scripts Window.

Multiple windows

A great feature in Blender is the ability to split the interface and use various window types at the same moment. The way to do this is very simple. We must right-click on the borders of an existing window to access a small menu with the options to split the window. We can split a window in two ways, which are vertical division and horizontal division.

When you place the mouse cursor at the border of a window, the cursor will change into a double arrow. Just right-click and choose **Split Area** from the menu as shown in the next screenshot, and a division will be created:



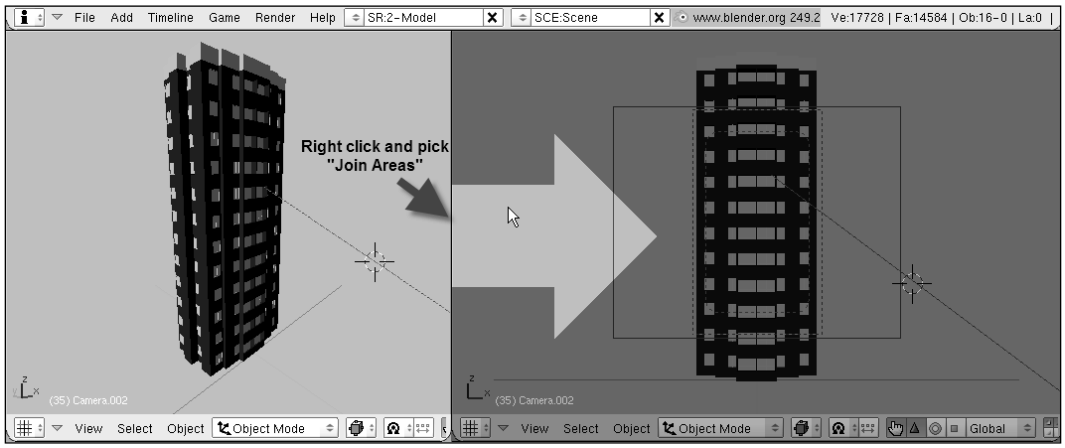
There are two kinds of divisions that we can create, which are vertical and horizontal divisions:

- **Vertical**: Click on the upper or lower border of a window to create a vertical division
- **Horizontal**: Click on the right or left border of a window to create a horizontal division

After choosing **Split Area**, just place your mouse cursor where you wish the division to be created, and left-click with your mouse.

Merge windows

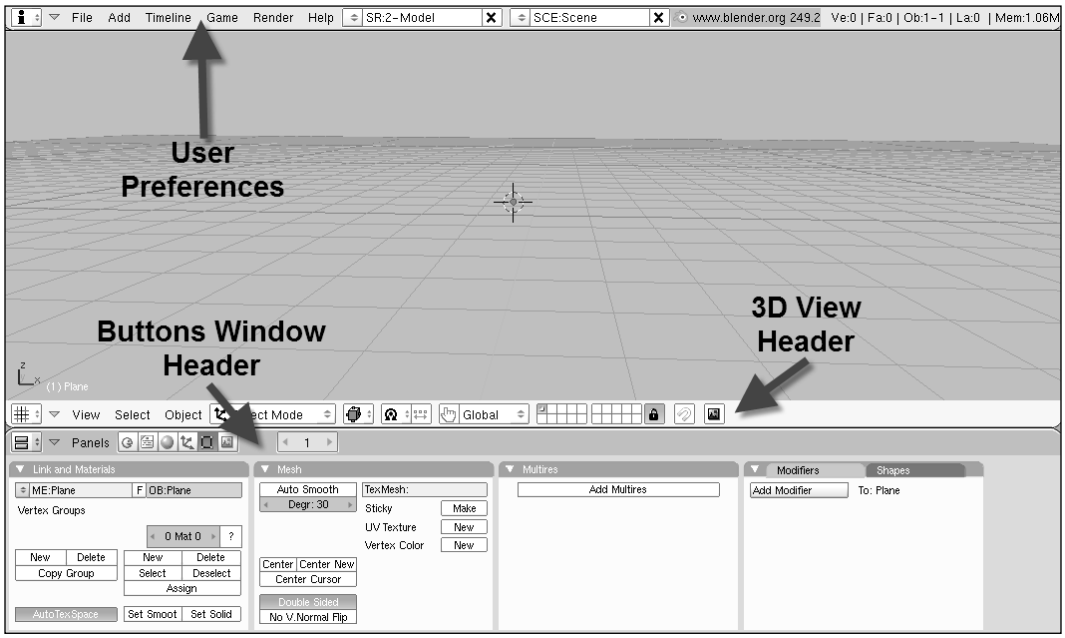
It's possible to merge two different windows too, with the same menu. There is an option named **Join Areas**, which will appear when we click with our right mouse button on the border of a window. After doing that, a big arrow will show which window will be destroyed and the arrow base shows the window that will take the place of the one destroyed:



When you have chosen which windows should be joined, just left-click with your mouse to confirm it. We must always join windows that share the entire border with each other. Windows that only share a part of their borders can't be joined together, and we must find another way to join those windows.

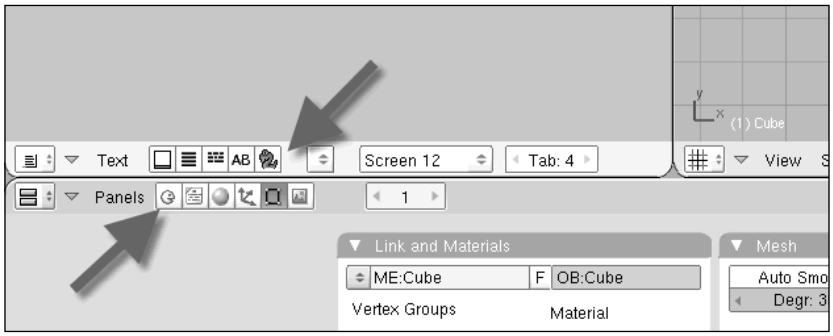
Header

Every window in Blender has a header that holds options related to the window. These headers show up as a horizontal bar that is attached to a window, and give us a few options. The headers can be placed at the upper or lower border of a window. In the default Blender interface, we can see three headers that are attached: Buttons window, 3D View, and User Preferences:

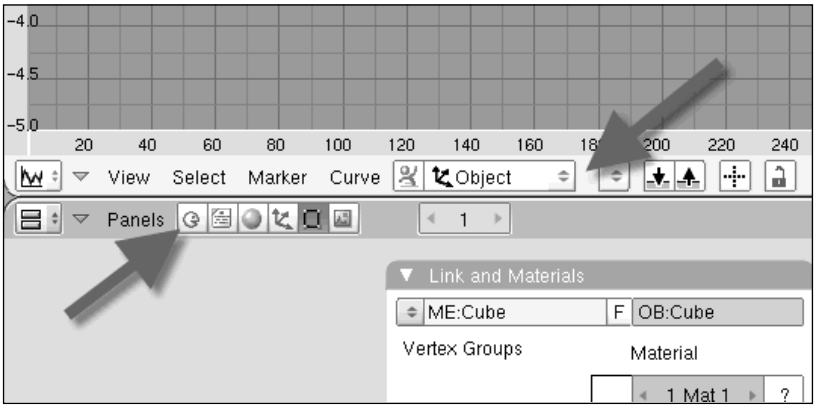


Let's see examples of what those options are for some of the Blender windows:

- **Text Editor**: The header for this window has options to edit and manipulate text. There are options to view the line numbers, change fonts, and edit tabulation, as shown in the following screenshot:



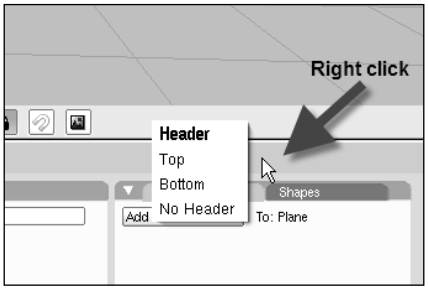
- **Ipo Curve Editor**: The header for this window shows us options like—zoom a specific area of the curve graphic, copy the curves to the clipboard, choose which type of curves to visualize, and more:



As we can see in the previous screenshot, for each window type, the header shows different and contextualized options related only to that window type.

Add or remove a header

To add or remove a header from a window, we use the same menu that creates divisions in a window. When we right-click with our mouse over the border of a window, we will have two options. If a header already exists, an option named **No Header** will show up, and if a header doesn't exist, another option named **Add Header** will appear:

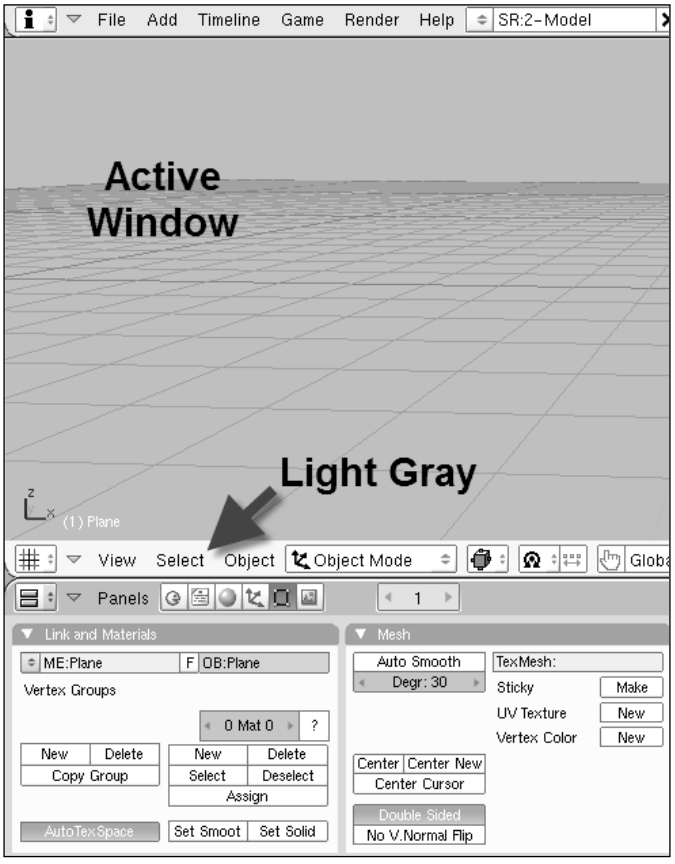


We can choose the position of the header and place it at the top or bottom of a window. To do that, we must right-click on an existing header, and a small menu will appear. With this menu, we can choose the position of the header, and there is an option to remove that specific header.

Active window

A very important concept for the Blender interface is the active window. When we are dealing with an interface divided with a lot of windows and window types, if we activate a specific tool or command—in which window it will be executed? Well, the answer is—in the active window.

Only one window can be the active one, and what makes a specific window the active one, is the mouse cursor location. When the mouse cursor is over a window, it automatically becomes the active window. We can even notice a small change in the color of the window, to a brighter gray:



To show that, let's do a small test that will show how the active window concept works. For that, we will use the *Home* key on the keyboard. When we press this key, a command adjusts the zoom to fit the visualization to all visible objects. Well, just place the mouse cursor over the **Buttons** window and press *Home*, and you will see that nothing happens. It's because all the menus and buttons there are already zoomed to fit the whole window. But, if you place your mouse cursor over the 3D View, and press *Home*, you will see that the visualization for that window will be adjusted, and we will see all the objects placed in the 3D View. Note that for this example to work, we must use the default Blender interface.

Every time your 3D View is too crowded or you change the zoom significantly, and lose your objects, just press *Home*, and the visualization will be adjusted. This is valid even for menus and the organization of the interface.

Keyboard shortcuts

We already know the Blender interface and how it works. Now it's time to start working with the keyboard shortcuts. One of the most interesting aspects of Blender is that it's built to give artists different ways to increase the efficiency of their production time. The way to do that is to focus on keyboard shortcuts, and Blender does that a lot. This may be difficult to new users, but for more experienced users, it's a real productivity gain.

There are shortcuts for almost every tool or command in Blender, and we will be using them throughout the rest of the book. With practice and continuous work, we will become more and more familiar with the shortcuts. So don't worry if you get yourself writing down a few of them at the beginning, just focus on the modeling and 3D work and soon you will naturally remember the more frequently used shortcuts.

3D visualization

When we start to work in a 3D environment, to know how to navigate and adjust the view to fit your needs is very important. At this point having a 3-button mouse is very important, because a lot of the navigation process is done either with the middle mouse button, or the mouse wheel.

Let's start with the orthographic views, which are the Top, Front, Right, Left, Bottom, and Back View. All these views can be activated with the numeric keyboard. To use these shortcuts, we must make the 3D View the active window by placing your mouse cursor over the window. Then just press these keys to activate the views:

Key	Action
8	Rotate view up
7	Top view
CTRL + 7	Bottom view
6	Rotate view right
5	Swap orthographic and perspective views
4	Rotate view left
3	Right view
Ctrl + 3	Left view
2	Rotate view down
1	Front view
Ctrl + 1	Back view
0	Camera view
Home	Fit the zoom to all objects

As we can see, there are more options besides the orthographic views, such as options to rotate the view. The orthographic views are those where the projection of the lines from the views are all orthogonal to the projection plane. In this case, the projection plane is parallel with the grid from the Blender 3D View. A few of the orthographic views include top view, right view, left view, and bottom view.

Those are the options to manipulate the view with the keyboard, but we can use the mouse to get even more control over the visualization. Here is a list with some combinations of mouse buttons and keys, to control visualization:

Key	Action
Wheel + *Ctrl*	Zoom in / Zoom out
Wheel + *Shift*	Pan view
Wheel	3D orbit view
Scroll Wheel Forward	Zoom in
Scroll Wheel Backwards	Zoom out

All these options are very important to manipulate and visualize your scenes, and only with a bit of practice will it be possible to be familiar with all the commands.

Wheel or Middle Mouse Button?

Remember that when we say wheel, it could be the middle mouse button too, if your mouse doesn't have a wheel. But with a Middle mouse button only, you won't be able to use the scrolling options. To use the Zoom and Pan options, you will have to use the *Ctrl* or *Shift* keys to manipulate the view. For laptop users with only a trackpad and two buttons available, there is an option in the User Preferences window of Blender named **Emulate 3-button Mouse**, that emulates the middle mouse button with the *Alt* + *left* mouse button.

Selecting objects

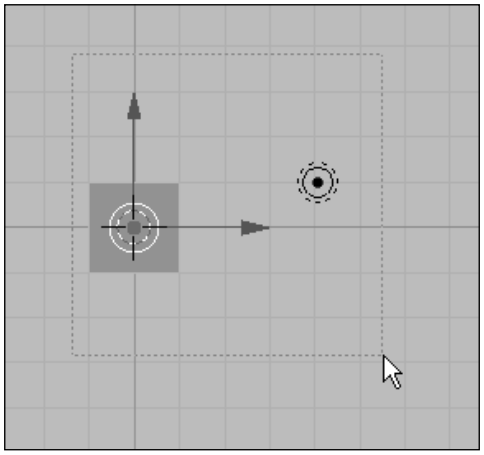
The process of selecting objects is very important to manipulate and transform objects in any 3D package. To select objects in Blender, we have options to use the mouse and to select objects by name. If we want to select a single object, just click with the right mouse button over this object, and it will be selected. To add another object to the selection, just press *Shift* and click with the right mouse button over another object.

Mouse buttons

In Blender, the mouse buttons work in a different way. In most 3D applications, the left mouse button selects one object. But with Blender, the right mouse button selects objects. If you find it confusing, it can be changed in the **User Preferences**.

The *A* Key is a very important tool when we are selecting objects, because it can unselect all objects that are selected and remove objects from selection. If there isn't any object selected, when we press the *A* key, all objects are selected. This is valid even when we are dealing with the selection of vertices, edges, and faces from objects.

To select multiple objects, we have the Box Select tool that works with the *B* Key. When we press the *B* key with the 3D View as the active window, we are able to draw a selection box around a group of objects. After pressing the *B* Key, just press your left mouse button and drag around the objects that you want to select. When all the objects are inside the selection, just release the mouse button and all objects inside the box will be selected:



If you want to remove the objects from the selection, just press the *A* Key, and all selected objects will be removed from the selection.

And if we press the *B* key twice in Edit mode, it will turn on the Brush Select. The mouse cursor will turn into a circle that we can use to "paint" the selection. Just press the left mouse button and drag the cursor to select anything that touches the circle. If you press the right mouse button and drag the cursor over any selected object, it will remove this object from the selection.

We can control the size of the circle with the + and - keys of the numeric keypad. Scrolling the mouse wheel, if you have one, will change the size of the circle as well.

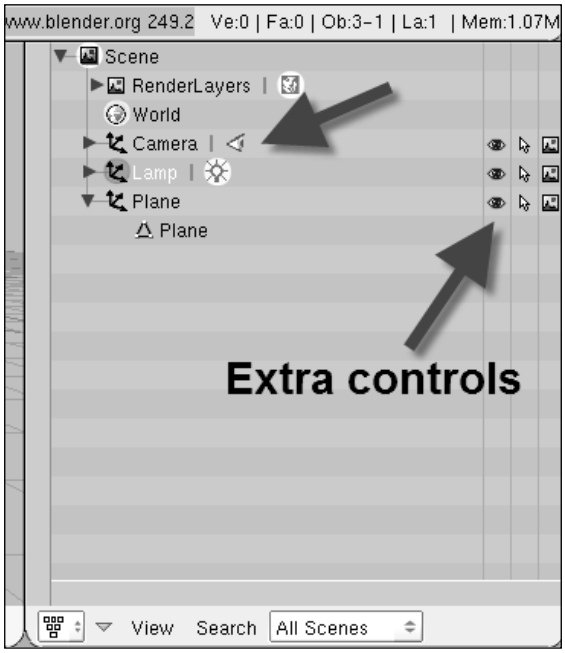
This type of selection doesn't work in the Object mode. We will talk more about work modes later in this chapter.

Selecting by name

When we start to work with more complex scenes, the number of objects in your screen will increase dramatically. Simple tasks, such as selecting one specific object, will become more complex, because we will have to find this object on the screen first. That's why sometimes it is a good practice to rename objects, so it will be easier to find them by name later.

To select objects by name in Blender, we use a window named Outliner. This window will show a list of objects, and there we can choose the objects that will be selected by name.

The Outliner window can be displayed in two ways, the Oops Schematic and the Outliner itself. To change between them, use the **View** menu in the Outliner window header:



To select an object, the process is simple; just left-click on the object name. If you want to select more than one object, just hold the *Shift* key while you click.

Besides the ability to select objects by name, we also have a few extra controls in the Outliner. On the right, we find three small icons that allow us to control a few properties of the objects:

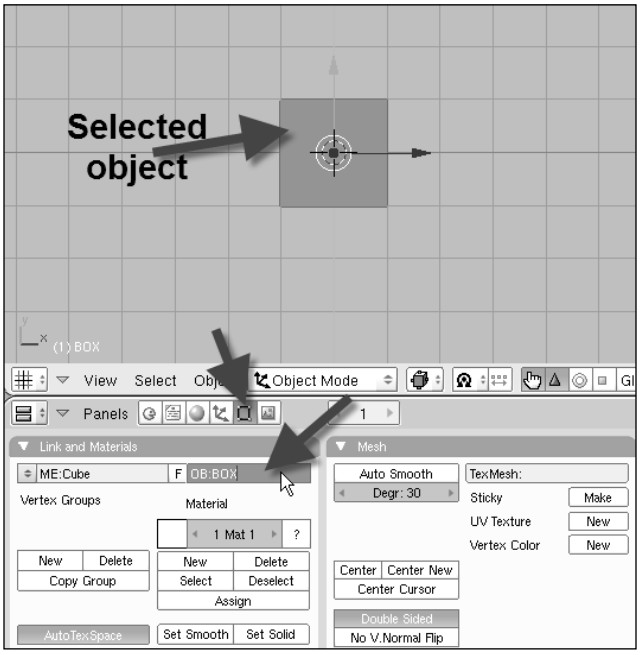
- Eye: With the Eye icon, we control the visibility of the object. If the eye is open, the object is visible, and if it's closed, the object is hidden. Right-click on it, to open and close the eye.
- Cursor: If this cursor is turned off, we won't be able to select the object in the 3D View. To turn the cursor on and off, right-click on it.
- Landscape: With this control, we can determine if the object will show up when you render the scene.

Renaming objects

We can rename any object in Blender to make the selection process easier, and your scenes more organized. Before anything else, it is important to remember that every object in Blender must have a unique name. If we try to rename two objects with the same name, Blender will automatically add a suffix to the name. This suffix will be a number, which places an order of creation.

If we have an object named "Box", and we try to give the name "Box" to another object, it will be automatically renamed to "Box.001". An important point to remember is that all names in Blender are case sensitive, which makes an object named "Box" different from another one named "box".

To rename an object we must use a panel named **Editing**. This can be accessed with the shortcut *F9* or just clicking on the small icon, pointed to in the next image:



Before renaming an object, we must first select this object. Then, click on the text box to rename it. When we select one object, we can see the object name in the lower left corner of the 3D View. Try to give names to objects that better identify the function of the object, like "left wall" or "window glass". It will be a great help when we need to select objects in complex scenes.

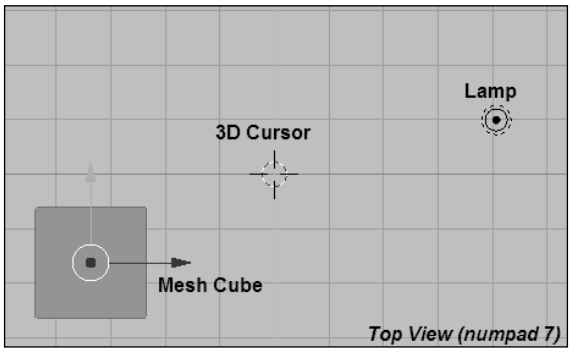
We can use the Outliner to rename objects also. Hold down the *Ctrl* key and right-click on the name of an object to rename it.

Renaming with the transform properties

There is another way of renaming objects with a small menu named **Transform Properties**. Just press the *N* key in the 3D View and select any object. In the **OB** text field, we will find the actual object name, and by clicking there, we can change the name of the object.

3D Cursor

When you are just get started with Blender, you will notice a small icon in the 3D View window that looks like a target. This is the 3D Cursor, which is an important component in Blender because it determines the place where objects are created. To place this cursor at another point of the 3D View, just click with your left mouse button anywhere in the screen:



Another function of the 3D Cursor is to become the center for object transformations. When we need to rotate an object, using a specific point as the center of the rotation, we can use the 3D Cursor as the center.

Cursor Snap

To work with the 3D Cursor and selected objects, we can use a **Snap** option to place the cursor in specific places. If we press *Shift* + *S*, the menu with the **Snap** options will appear. There are a few options in the menu:

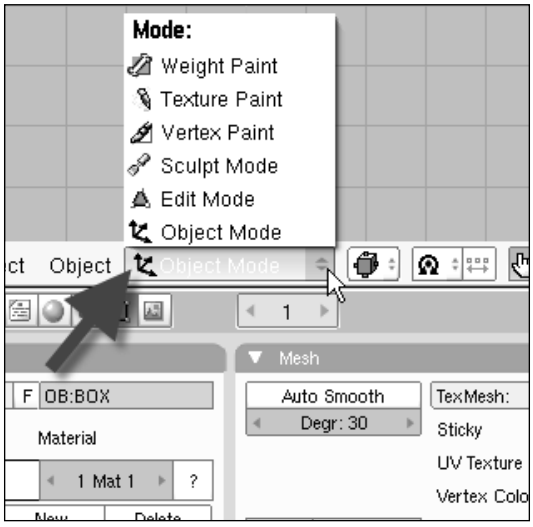
- **Selection | Grid**: This option aligns the selected objects with the grid lines.
- **Selection | Cursor**: With this option, we can align a selected object with the 3D Cursor. This way, the selected object will be moved, so it's center point is placed at the center of the cursor.

- **Selection | Center**: With this option, we can move the selected objects to the geometric center of the selection.
- **Cursor | Selection**: Here we will move the 3D Cursor to the center of the selected object.
- **Cursor | Grid**: This option aligns the 3D Cursor with the grid lines.
- **Cursor | Active**: Align the 3D Cursor with the active object.

With these options, it becomes easier for us to place the 3D cursor and the selected objects in specific places. We will use this option a lot when we deal with modeling for architecture, because it's a great tool for precision modeling. Every time we need to place the 3D cursor at a specific place, remember to use the Cursor Snap. In Chapters 4, 5, and 6, we will talk more about other methods of snapping with the snap tool.

Modes

To edit and manipulate objects in Blender, we must understand how to work with the modes. These modes determine what we want to do with an object. To manipulate and transform an object, we have the **Object Mode**, and to edit and model, we have the **Edit Mode**. To select these modes, we use a combo box located in the header of the 3D View:



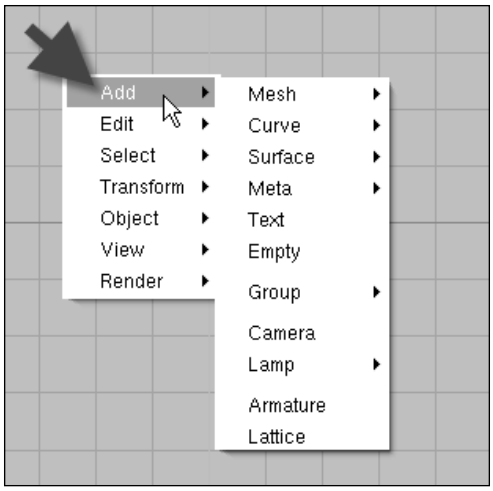
We have more modes, such as the **Sculpt Mode** and **Pose Mode**, that activate sculpt modeling tools and character animation, respectively. The most common modes we will be using are the Object and Edit modes. They are so important that we have a Keyboard shortcut to switch between them. If we press *Tab*, the active mode will be switched between Object and Edit.

See when each mode should be used:

- **Object Mode**: This mode is used to manipulate and transform one or multiple objects. If you need to scale or move an object, use this mode. When we select an object in **Object Mode**, we won't be able to change it's shape, by altering it's vertices, edges, or faces.
- **Edit Mode**: With the **Edit Mode**, we will be able to select and edit the vertices, edges, and faces of an object. This way we can easily deform and model an object. Use this mode only if you need to deform and model.

Creating Objects

Now that we know how to manipulate our interface and deal with basic aspects of Blender, let's learn how to create objects. To create anything in Blender, we use a menu named **Toolbox**. This menu appears when we press the *Space bar* on the keyboard or the *Shift* + *A* shortcut:



This menu has a lot of options, and one of them is the **Add** option. There we can find all the object types that can be created in Blender. In the modeling chapter, we will deal with many of these options, like curves and surfaces. To get started, let's see the most basic type of object, which is **Mesh**.

When we choose the **Mesh** type of object, options like **Cube**, **Circle**, **Cylinder**, and other basic shapes are available for us to choose. These are all shapes that can be deformed and divided into subforms to create more complex shapes.

To create an object, just place the 3D Cursor where you want the new object, and press the *Space bar*. Choose which kind of object you want, and it will be created. After creating an object, it is important to note the following:

- After creating an object, Blender will keep the same work mode.
- All objects are created in alignment with the base grid. This means that new objects will be added aligned with the orthographic x, y, and z axis.

Another point to remember is that any objects created in Edit mode, will be added to the edited object. For instance, suppose we start to edit a UV Sphere, and while in Edit mode, press the *Space bar*, and add a Cube. Now, we will have an object made by a Cube and a UV Sphere. To avoid that, always add new objects in Object mode.

This is very important because a lot of new users get this wrong and create all objects in Edit Mode. If you do that, your objects will be created as a single block. And it will demand some time later to split it, so it's a good practice to always switch back to object mode, unless you want it to be all together.

If you forget about it and create an object in perspective view, just use the **Clear rotation** command, and it will remove any degree of rotation applied to an object. Just press *Alt* + *R* to access this option.

Creating objects with no alignment to the view

There is a way to make all created objects aligned with the x, y, and z axis and not the view. Open the User Preferences window, and go to the **Edit Methods** tab, and turn off the **Aligned to View** button.

Erasing Objects

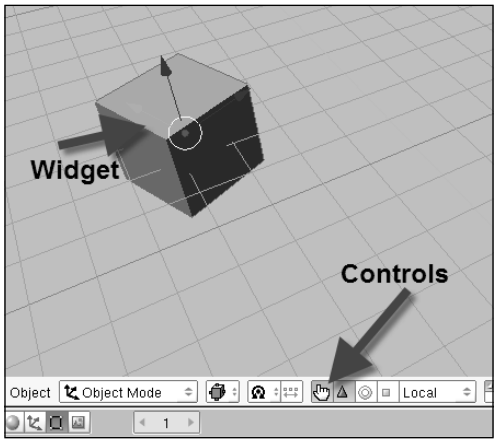
If for any reason, you have to erase an object, just select the object and press either the *X* or *Delete* keys. In Edit mode, we can select and erase vertices, edges, and faces.

Duplicating Objects

Creating copies of objects in Blender is very easy; just select the object that will be copied, and press the shortcut *Shift* + *D*. This is the simplest way of creating a copy for an object.

Transforming Objects

There are three basic transformations that we can apply to an object—Translate, Rotate, and Scale. To apply these transformations to objects, we can use both keyboard shortcuts and a transformation widget. The transformation widget is a simple icon that is displayed in the center of all objects:



There are four kinds of widgets for each transformation type. We can switch between them with the controls located in the header of the 3D View. The symbols represent each transformation type:

- **Finger**: Turn the widget off and on
- **Triangle**: Turn on the translation widget
- **Circle**: Turn on the rotation widget
- **Square**: Turn on the scale widget

Each of these widgets has individual controls for the individual axes. These controls are separated by color; red for X, green for Y, and blue for Z transformations. Then, if we want to rotate an object just in the Y axis, we will use the green arc that represents this transformation in the rotation widget. The same concept is applied to the other transformations. For a scale in the Z axis, we just grab the blue square in the scale widget and drag it.

There is a shortcut to turn on and off the widget, and select different types of transformations. We can use the *Ctrl + Space bar* to access a menu that will let us enable and disable the widget, and choose what type of transformation we want to use in the object. In the menu, we will find a widget named **Combo** that shows all the transformation options at the same time.

The other way of applying transformations to objects is with keyboard shortcuts. We can use these shortcuts to quickly transform any object. These are the shortcuts:

Key	Action
G	Grab (move)
R	Rotate
S	Scale

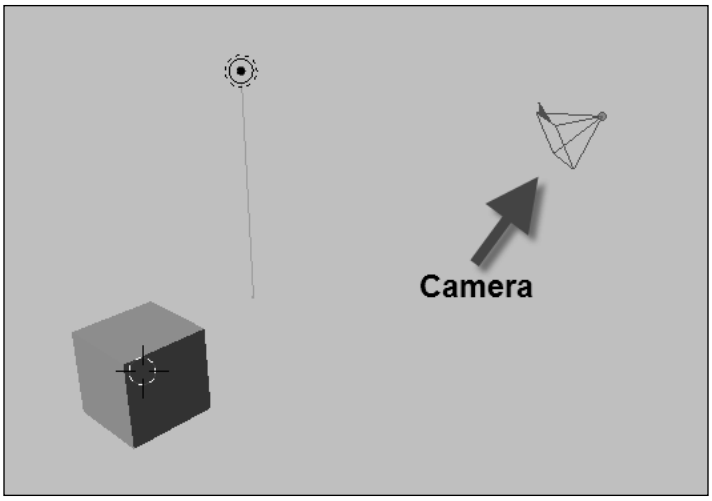
Besides these shortcuts, we can constrain the transformation to an axis with the use of more shortcuts. After pressing one of the shortcuts above, we can press *X*, *Y*, or *Z* keys to constrain the transformation to one of their respective axes. For example, if we press *R* key and then press *Z*, the selected object will rotate only in the *Z* axis.

The co-ordinates used to make that transformation are the global coordinates, but we can use the local coordinates of an object. To do that, we must press the key corresponding to the axis two times. Then, if we select an object and press the *G* key, and press *X* key two times, the transformation of the object will take place in the object's local coordinates.

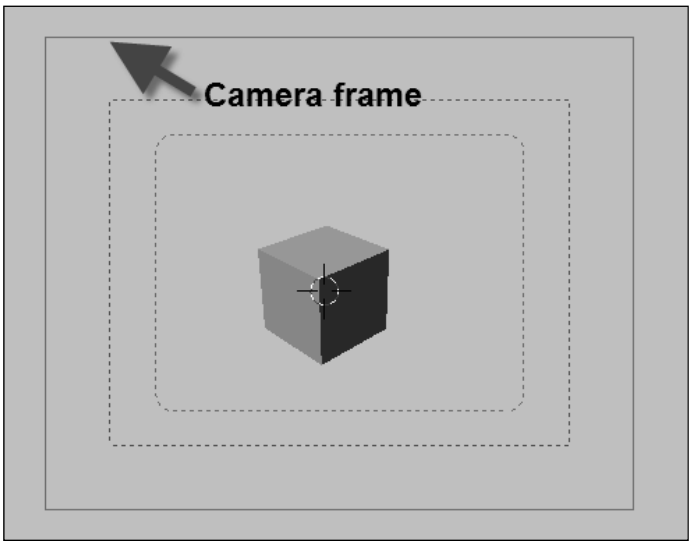
Global coordinates are related to the scene coordinates, and they don't change. And, the Local coordinates are related to the object, so they change with the object. We could compare the Global coordinates with the cardinal points, and the Local coordinates with the front, back, left, and right sides of one object. No matter where you are, the cardinal points will be always at the same position. And, the sides of an object will be relative to its orientation.

Cameras

Dealing with the camera in Blender is very important, because we can only render the camera view. Even with multiple cameras in a scene, we must always have an active camera that will be used in the rendering process. The camera is a small pyramidal object that is placed in the default scene:



To view what the camera is visualizing, just press *0* on the numeric keyboard and you will see the camera view:



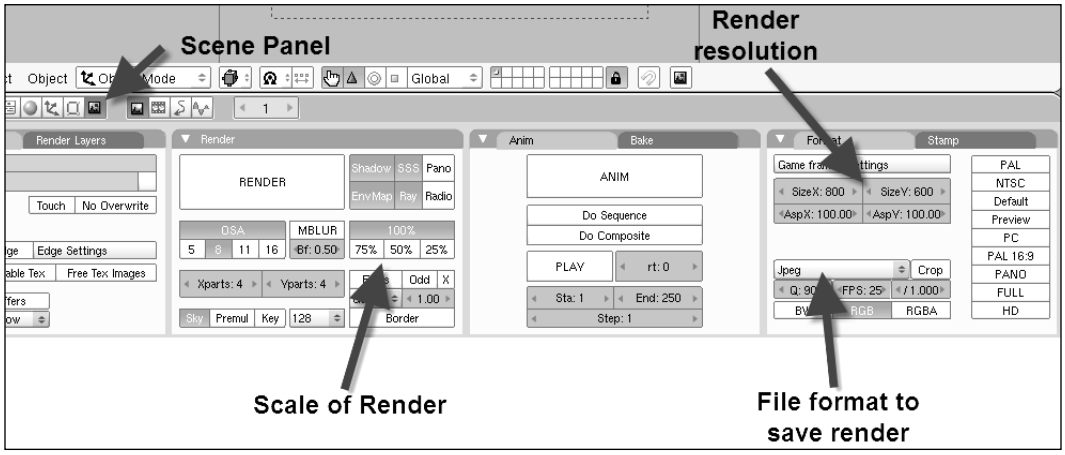
When we are in the camera view, select the camera by right-clicking on the camera frame, and press the transformation shortcuts to move or rotate the camera and change the view. In this way, we can adjust the framing and find a better fit for an object or scene.

Another way of adjusting the camera is to split the 3D View into two windows. Then, switch one of the windows' view to be a camera view. In the other view, using the widgets to move the camera or the objects, find a good frame for the objects.

If we have more than one camera, it will be necessary to set one of them as the active camera. To do that, we must select the camera and press *Ctrl + 0*. The *0* must be pressed from the numeric keyboard. In Blender, every object can be turned into a camera. Just select the object and press *Ctrl + 0*, and it will become a camera. To make an object not be a camera anymore, just select this object and press *Alt + 0*; this way, it won't be a camera anymore.

Render Basics

Now that we know the basic aspects of Blender, it's possible to learn how to render scenes and make some small adjustments. All the setup for the render can be accessed in the **Scene** Panel, which is located in the Buttons Window. To access this panel with a shortcut, press *F10*:



In this panel, there are two menus that hold the main options to set up the render. The first menu is named **Render**, and has options to set the scale of the generated image. We can choose scales between 100% and 25% of the original size. The other menu is named **Format**, and there we can find options to set the resolution of the render and a file format in which to save our renders.

We can set up a render resolution of 640x480, and choose PNG as the file format in which to save the image. If the scale is set to 100%, our image will have the full resolution, but if we want to run a test render at just 50% of the resolution, then we must change the settings in the **Render** menu.

Changing the scale for the rendered image is a great way of producing small renders, just to test our settings for lights and materials. We will be using this feature a lot for exercises.

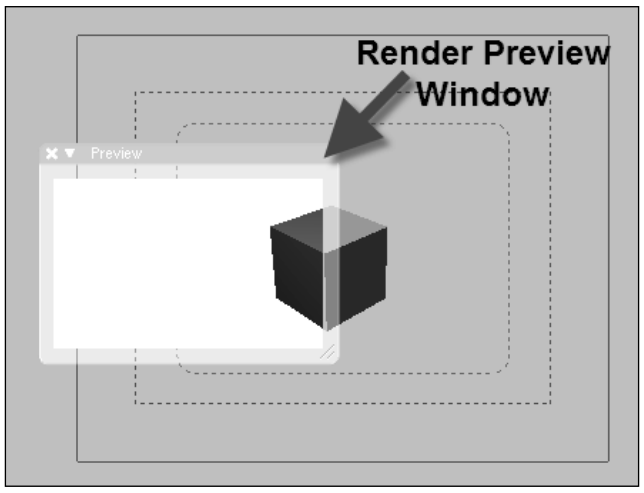
When everything is ready, just press *F12* or push the **Render** button in the **Render** menu. This way a render window will appear. After the render is finished, just press *F3* to save the file.

Make sure that you have at least one light

If we don't have any light sources in Blender, the render won't show anything but a black screen. It's different from other applications that have a default light. So, if you press *F12* and all you see is a black screen, it's probably because a light source is missing.

Render preview

Before starting a render, we can preview how it's going to look in the 3D View. There is a tool named Render Preview that can be accessed with the shortcut *Shift* + *P*. When we do that, a small window will show up in the 3D View:



We can change the size for the preview; just click on the border of the preview window and drag your mouse. That's another great way of doing a preview for your settings, to make sure everything will be fine when we need to make a bigger rendering.

Summary

In this chapter, we had a glance at Blender and how to start using it. Even for a quick start, we saw a lot of new and interesting concepts about how to work with Blender. We just learned how to:

- Use the interface
- Set up the interface
- Select Objects
- Work with modes
- Transform objects
- Create objects
- Copy objects
- Work with the camera
- Rendering basics

Even though this has been a quick start, to know and understand how Blender works is very important, because in the next chapter we will start using more complex and advanced stuff related to modeling for architecture in Blender.

3
Modeling

Now that we know the basics of Blender, the next step is to start working with the modeling tools and techniques. This will give us a greater ability to deal with specific aspects related to architectural and landscape modeling.

Types of objects

There are many object types in Blender, which can be created with the ToolBox. This is a special menu that we can use to create almost anything in Blender. To call the ToolBox, we must press *Shift* + *A*. The ToolBox will show the options to create Meshes, Curves, Surfaces, Text, and all other object types in Blender. Before we take any serious step into modeling, we must understand the differences between those types and when each one of them is most useful.

Let's take a look at each object type:

- **Mesh**: This is by far the most important object type in Blender, and we will be using it a lot. Here we have some primitive shapes like cube, sphere, cylinder, and others. We use these objects to create more geometrical forms, which is exactly what we will be using to create architectural models for buildings. The objects here are composed of vertices, edges, and faces. Most architectural modeling tools use a different type of object to model, named solids. Solids are a heavy object type, which demands more computer resources. So, with meshes, we have an advantage, and can work on more complex projects with fewer computer resources.
- **Curve**: These objects are curves that can be used to model and shape objects with perfect curved edges. This kind of object will be very useful for us to build animation trajectories.

- **Surface**: The surfaces are a type of object that is not commonly used to model in Blender (at least not for the kind of models that we will be creating), but can be very handy to create landscapes. It works with a surface that must be deformed to a specific shape, with the basic transformations. The deformation of those surfaces works with the position of control points from that surface. When we select one of those control points, the surface around that point is deformed to match the new position. The technique required to create objects with those surfaces is very different from mesh modeling. It's because the surfaces and the curves are based on NURBS curves, which are great to create organic shapes. The reason that makes NURBS curves great for organic shapes is that by default they are already created as smooth objects, which could be deformed and edited into almost any organic shape.
- **Meta**: This type of objects works like some kind of clay, in which we shape some primitives like spheres and cubes together, and create very organic shapes. The main use of Meta objects will be in landscaping, for terrain such as mountains.

These are the main object types in Blender. Of course, there are more objects, such as text and lamps, but for modeling, these are what we need. We won't be using all the types for modeling. But there is one type that will require more attention—the Mesh objects. This is because we will be working with geometrical shapes, and Blender is designed to work better with a type of modeling named subdivision.

To work with subdivision, we have to create primitive shapes and edit, transform, cut, split, and work on their vertices, edges, and faces. A lot of our modeling will be done by this method. If we think that a wall, floor, and other parts of a building can be created with cubes and planes, it becomes very clear why subdivision modeling is best suited for us.

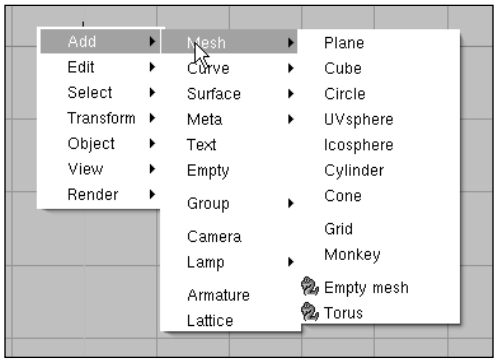
Before we head to subdivide your primitive objects, let's examine the main aspects of those objects and how to create them. This is just the starting point; it may even seem monotonous, but it is an important step. Once we understand and master all these basics, we will be able to work on more complex aspects of architectural visualization with Blender.

Subdivision and Subdivide

In polygon modeling, we often use the terms subdivision and subdivide during the process. By subdivision we mean the recursive method used to add new divisions progressively into polygons, and with that create a smooth surface. The word subdivide is related to a modeling task where we will add new edges to a polygon, and with that add a subdivision.

Mesh primitives

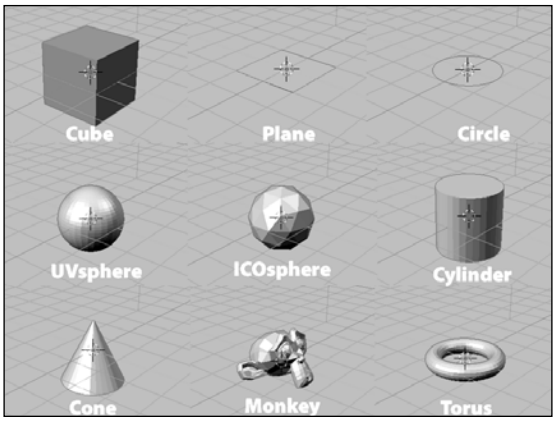
The first step to work with subdivision modeling is to learn which primitive shapes we have in Blender. This is because our first step will be always creating one of these shapes. To create these Mesh objects, we press the *Space bar* or press *Shift* + *A*. In the toolbox, we choose **Add | Mesh**:



As we can see, there are a lot of primitive types from which to choose. When we create some of these primitives, some parameters like number of sides and subdivisions must be set. Let's see how it works:

- **Plane**: This option creates a simple plane with four vertices. There aren't any more parameters for this object. We can start almost any type of modeling with a plane; objects like walls, roofs, floor, and others.
- **Cube**: With this option, a simple cube is created, with eight vertices. There aren't any options to set up a Cube Mesh. Like the plane object, we can start almost any type of modeling with a cube.
- **Circle**: The name of this option could be Polygon, because it's what we can create here. When we choose Circle, we must set a number of sides for the object. There are two parameters, which are **Vertices** and **Radius**. The first one sets the number of sides for the circle and the second sets the distance between the center and the border of the circle. If we need a square, just create a circle with four vertices, or with six vertices to create a hexagon.
- **UVsphere**: Here we can create a sphere with square faces. There are three parameters to set, which are **Segments**, **Rings**, and **Radius**. The **Segments** and **Rings** set the amount of subdivision for the sphere. Higher values result in more perfect spheres. But beware; higher values demand more from the computer and can eventually crash it. The **Radius** sets the distance between the center of the sphere and its border. The UVsphere has meridians and parallels, like the earth.

- **Icosphere**: Here also, we have a sphere. The only difference is that this sphere is made up of triangular faces, not squares as in the UVsphere. The setup here is simpler; we don't have Segments and Rings, just a **Subdivision** option.
- **Cylinder**: This option creates a Cylinder, but like the Circle option, we can use it to create shapes, such as a prism. In fact, with a Cylinder, we will get a polygon with a height, which could be useful to model a pillar or a column. We must set up three parameters here. Like the circle, there are **Vertices** and **Radius**, but there are two more options, which are **Depth** (that determines the height of the object), and a button named **Cap Ends**. If this button is turned on, the object will have two faces to cap it's upper and lower extremities.
- **Cone**: Here we can create a cone-like object, just like the Cylinder; if we set up a lower number of sides; other types of objects can be created, like pyramids. The options are almost the same as the Cylinder; the only difference is the button **Cap Ends**, which will fill only the lower extremity of the object.
- **Grid**: This object works much like a Plane, with the difference being that a Grid has more subdivisions than a Plane. To set up a Grid, we must provide the subdivision level for the X axis and Y axis, with the **X res** and **Y res** options.
- **Monkey**: Yes! We can create a monkey head. If you are asking yourself, why place a monkey head here? Well, this is Suzanne, who is the Blender mascot. We won't be using it to model anything, but because the head has some good geometry, it turns out to be a great option to test materials and textures.
- **Torus**: Here we have another object type, which will give us something like a doughnut or tire. When we create a Torus, we must set four parameters. There is a **Major Radius** and **Minor Radius**, which set up both external and internal radius of the Torus, and there are **Major Segments** and **Minor Segments**, which determine the level of subdivision for the Torus.

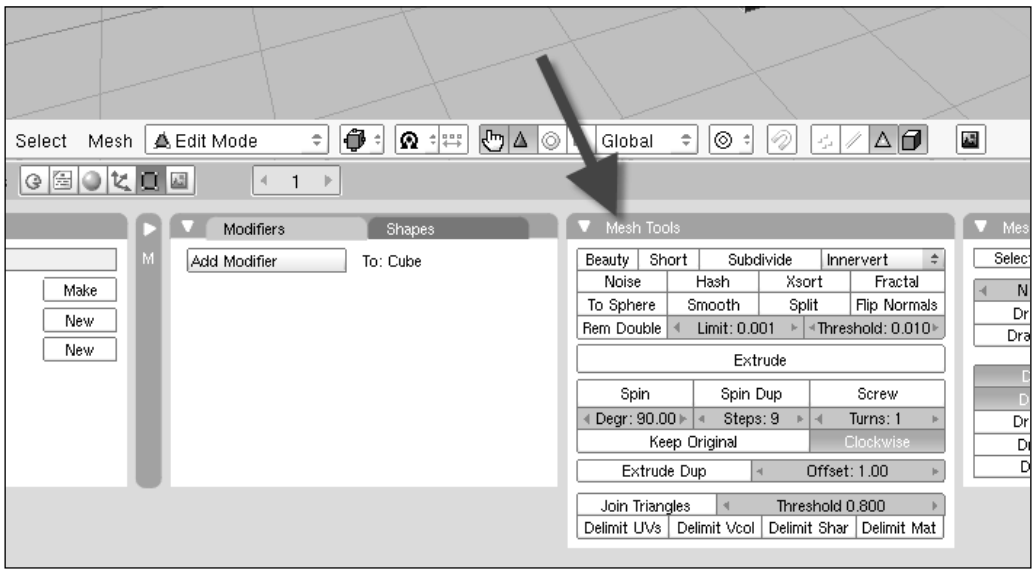


These are all Mesh type objects available to us in Blender. Which ones of them are the most important for architectural modeling? We will be extensively using the Plane, Cube, and Circle to model. Those are common shapes, and we can create other shapes easily with them. A circle can become a cone with editing, and a plane can become a grid. It's just a matter of using the right tools.

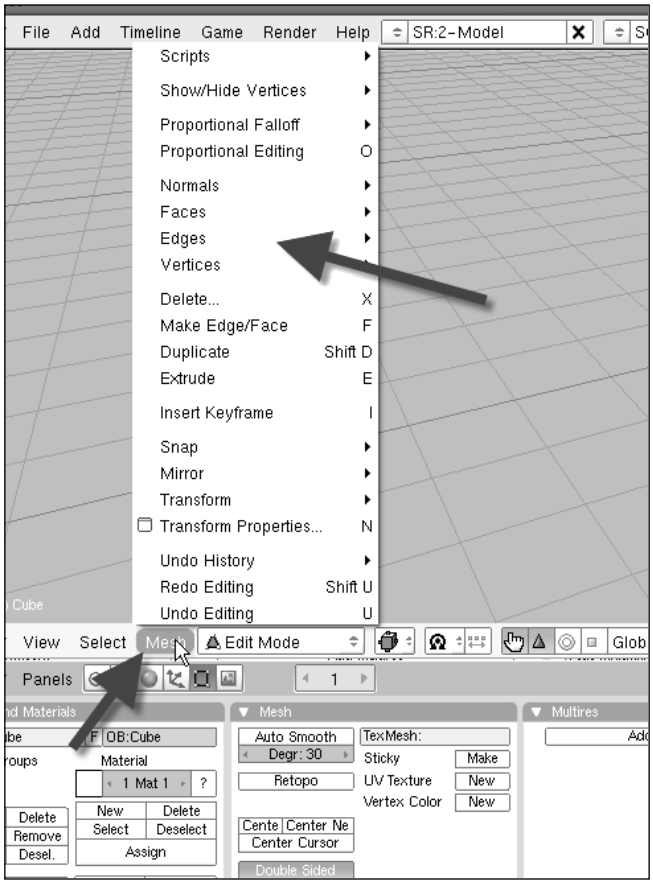
How can a primitive shape become a more complex object like a wall or a window? This is the basis of the mesh and subdivision modeling, which are used by Blender. We can compare this modeling with some kind of sculpture, where we have to take something simple and cut, push, pull, and rotate to create something complex. Let's learn how to apply these transformations.

Mesh editing

To subdivide a Mesh object and get more complex shapes, we must apply some editing to these objects. For that, Blender has a number of Mesh Editing options to cut, slice, split, and manipulate the vertices and edges in several different ways. Many of these tools are located in a menu named **Mesh Tools**, which is located in the **Editing Panel**:



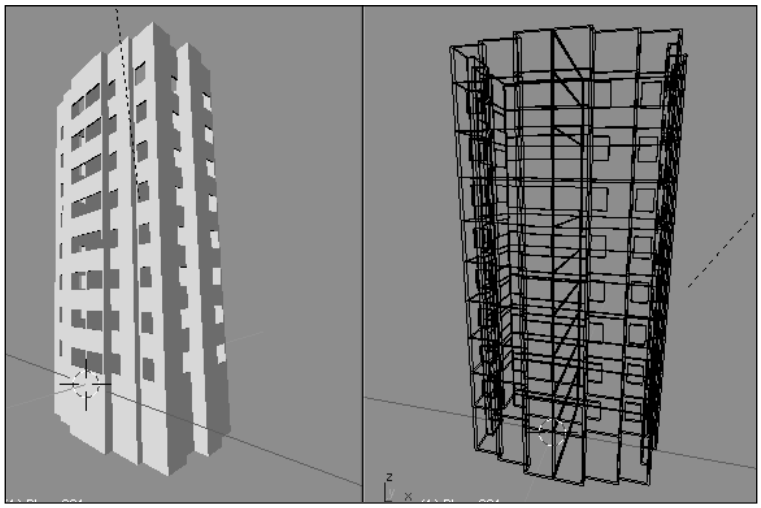
But these are not the only tools available; some of them are placed in the **Mesh** menu, located in the header of the 3D View window. This menu has options to edit vertices, edges, and faces:



Let's see how to edit the Mesh objects with some of these tools that will be important for us when we deal with more specific modeling for architecture and scenarios.

Because Mesh editing is important to architectural modeling, we will use a simple model of a building for most of our examples. This chapter is not aimed at architectural details or elements, but it doesn't mean we can't contextualize our examples to make things more clear.

Following is the model that we will be using for some examples:



It's a simple building, but we can use it to exemplify the mesh editing options. In the following chapters, we will be adding more details to this model, until we get to the final render!

Transformations

The first tool to edit mesh objects is transformations. They are very simple tools, but it is with transformations that we will create most deformations in meshes. If you don't remember the last chapter, we will repeat the shortcuts to transformations:

- *G* key: Grab or move
- *R* key: Rotate
- *S* key: Scale

These transformations can be applied to vertices, edges, or faces. We can also constrain the transformations, with the *X*, *Y*, or *Z* keys, pressing these keys right after the key corresponding to the transformation. For instance, if we start to move a cube with the *G* key, just press the *X* key while the cube is moving, and the transformation will be constrained to the x axis.

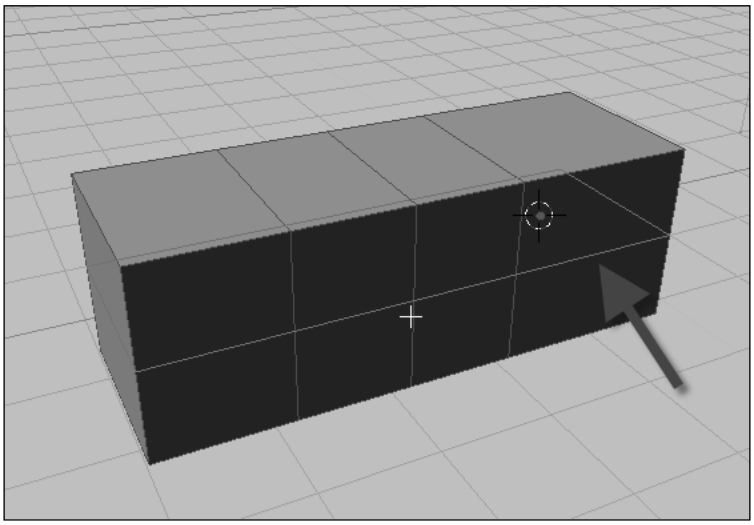
Transforming with precision

To make transformations with precision, we can hold down the *Ctrl* key to use the gridlines as a guide. For instance, if we try to move an object and hold down the *Ctrl* key, the transformation will be constrained to the gridlines. The same goes for scale and rotation transformations. And to fine-tune the transformation even more, we can hold down the *Shift* key instead of the *Ctrl*. By using the *Ctrl* key, all transformations will be constrained by 1 unit, because that is the default distance between the gridlines, and if we hold the *Shift* key, the transformations will be constrained by 0.0001 units.

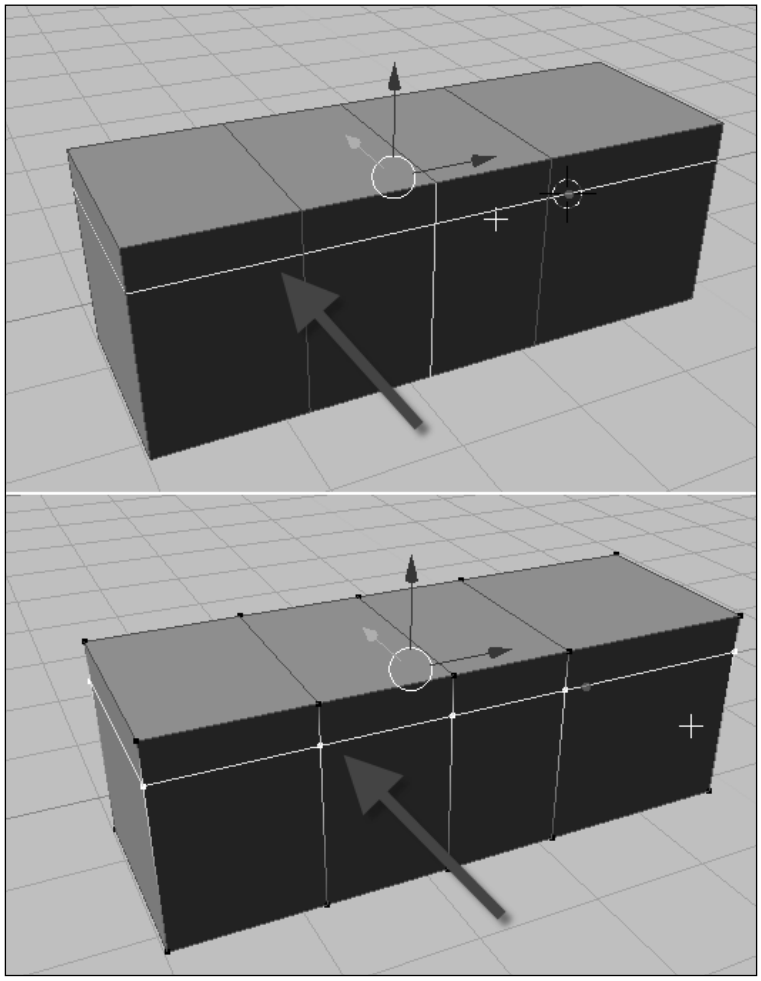
Loop Subdivide

When we create a cube or a plane, these objects appear with no subdivision and very few vertices and edges. To create more complex forms, we will certainly need more edges to extrude and transform. There are two ways for creating a loop subdivide. We can use the **Mesh** menu in the header of the 3D View and choose **Edges | Loop Subdivide**, or use a keyboard shortcut, which is a lot faster, with the keys *Ctrl + R*.

Before we use this option, we must select one Mesh object and enter into Edit mode. The loop subdivides only when working in this mode. If everything was done right, when we press *Ctrl + R*, a pink line will appear around the object. In the following example, a cube was scaled with the *S* key to make it narrower:

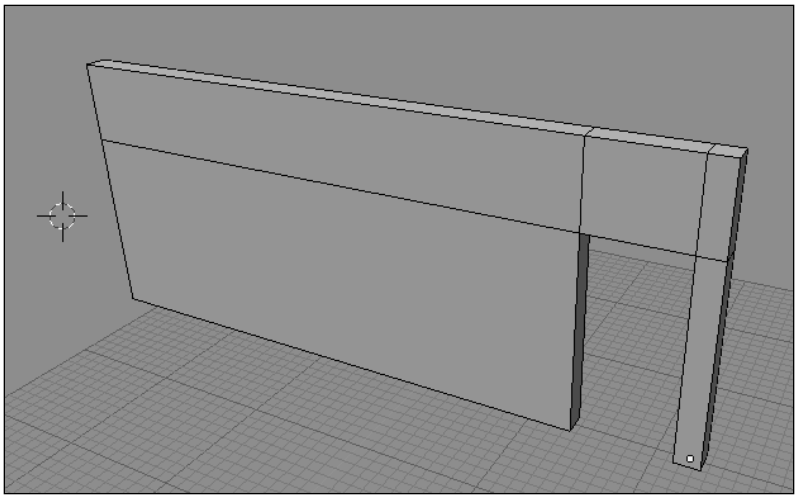


This line will place the loop for the model, and we can change the orientation of the cut by moving the cursor around the object. Then, a yellow line will appear for us to choose the right place to create the cut. Just move the mouse cursor to place the cut somewhere, click with your left mouse button to finish, and a new edge loop will be added to the Mesh object:

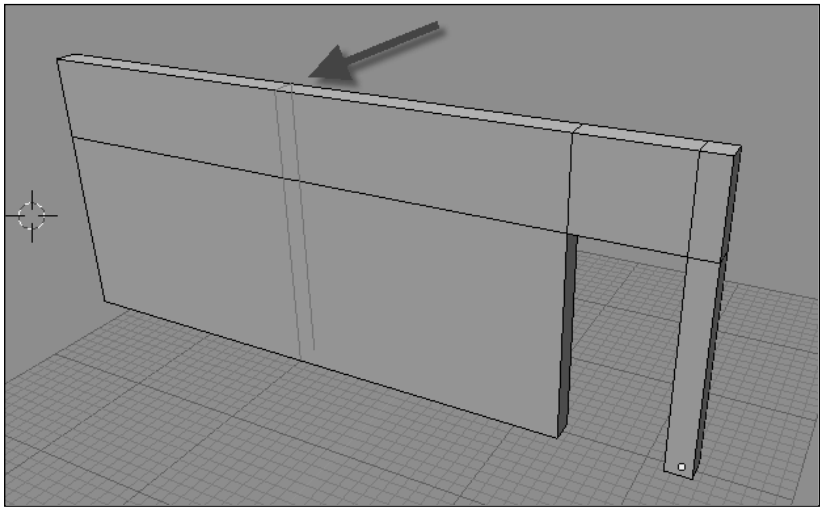


When we use this kind of tool with additional transformations, the possibilities to create new shapes increase dramatically. Let's see an example:

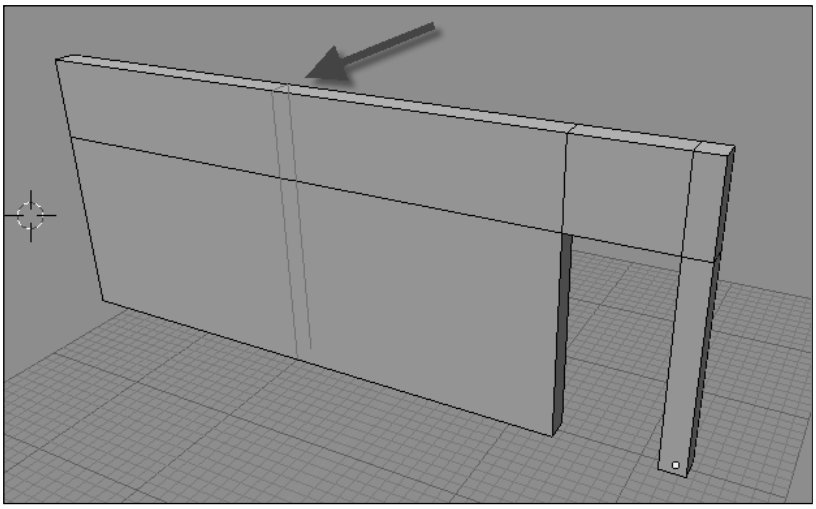
- With the Loop Subdivide, we can change the shape of a wall, and add different planes to it.
- We have a model of a wall with an opening for a door, transformed from a base cube:



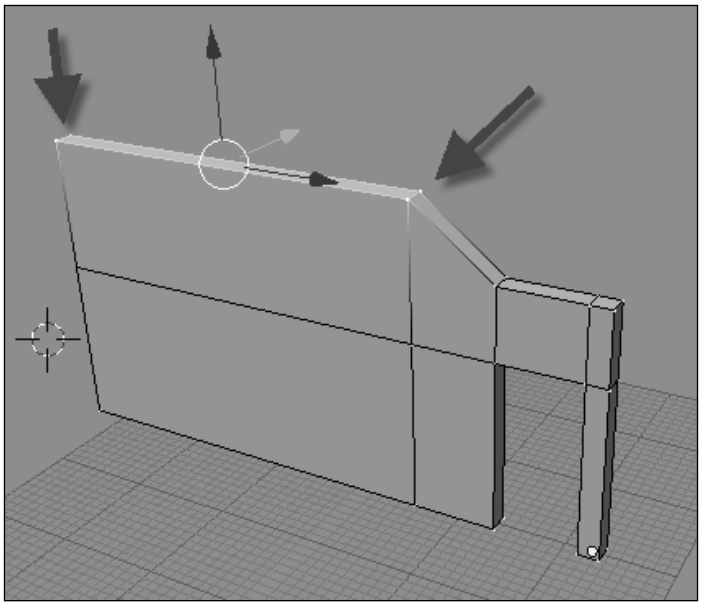
- Let's change the geometry of that wall with a loop subdivide. When we press the *Ctrl + R* shortcut, a new loop subdivide will be added to the wall:



- We can place the new loop right before the door:



- Now, we select the vertices selected in the following image, and scale them in the Z axis. It will make a slant plane:



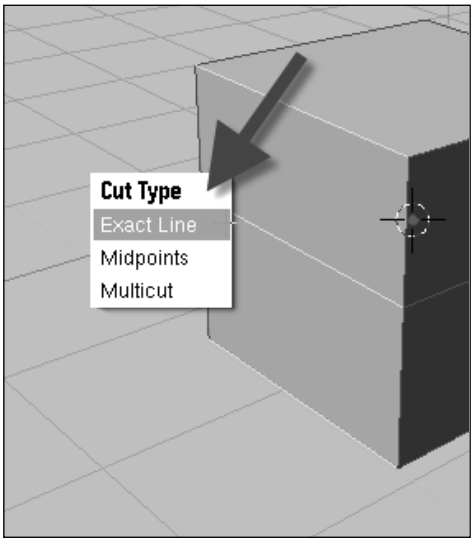
This new shape was possible with the use of a loop cut. Of course, there are other ways to create this kind of geometry, but this is a good example of what we can do with loop cuts.

Knife tool

The loop cut is a great tool for creating regular subdivisions in Mesh objects, but sometimes we will need more irregular or non-orthographic cuts. For that, we have a Knife tool, which can create cuts in a number of different ways. To use this tool, a Mesh object must be selected, and Edit mode activated. Before we start, another important thing about this tool is that the face, or part of the object that will be sliced, must be selected. So, if we are going to cut a specific face, this face must be selected before we start.

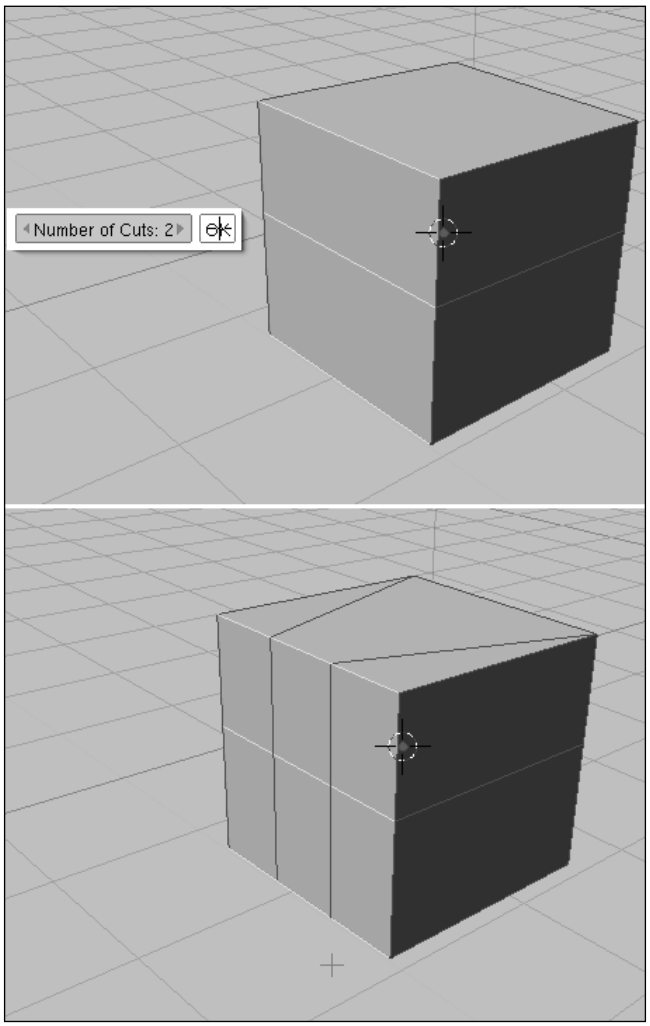
There are two ways to activate the Knife Tool:

- The first is with the **Mesh** menu in the 3D View header. If we choose **Mesh | Edges | Knife Subdivide**, the Knife tool will be activated.
- The other way is with a keyboard shortcut; by pressing *Shift* + *K*, we can activate the Knife tool as well. To create a cut, we must draw a cutting line above the selected faces that we want to cut. Before we get to draw this line, there are some different types of cuts that we need to understand. We have three different types of options to use with the Knife, and a menu will always appear right after we activate the tool:



These options are:

- **Exact Line**: With this option, the cut will be created using the line that we draw over the Mesh.
- **Midpoints**: This option creates the cut using the midpoints of the selected faces. The line that we draw will work as a guide for the tool to choose what midpoints to use.
- **Multicut**: Here we have an option to create multiple cuts for a face. After choosing this option, a menu will appear asking the number of cuts that we want for that face. If we choose two cuts, then your selected face will look like this:



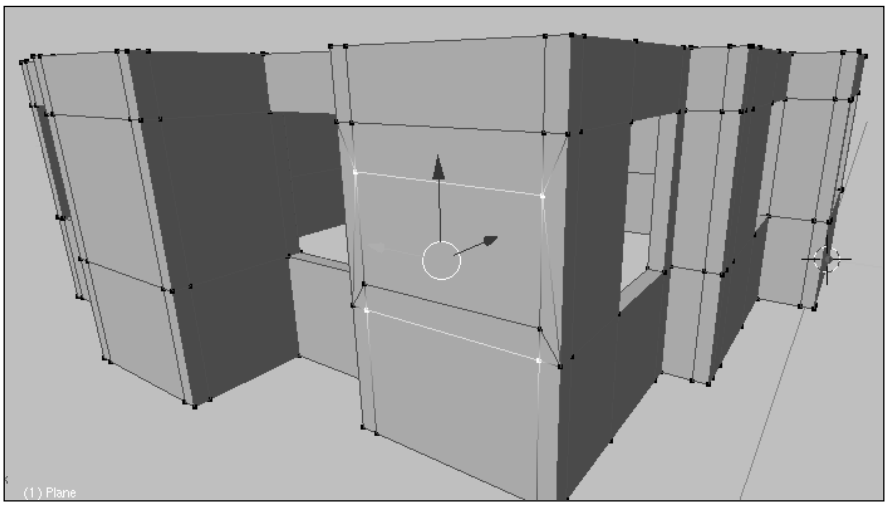
As we can see, this is more like an option to create cuts with non-orthographical lines, but it's a nice option for more geometrical modeling.

Loop/Cut menu

There is a menu which has options to choose between the Loop Cut and Knife tools. When you are in Edit mode, just press the *K* key, and a menu will appear with four options to access these tools quickly.

When we use the knife tool, we will have in some cases faces with three or four sides. Here is something very important in Blender—avoid three-sided faces, especially if you want to edit this object later.

Three-sided faces mess with the edge loops, and will make the editing process harder. Let's see an example:

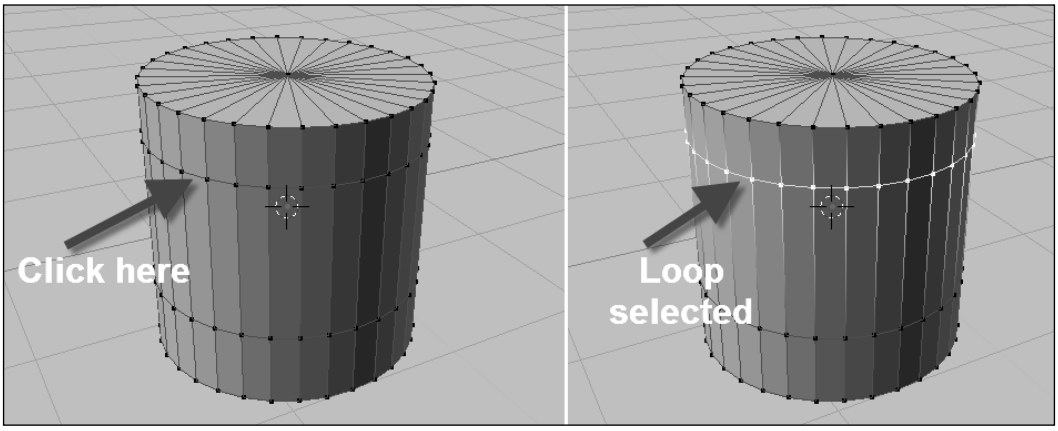


The previous image shows the result of the using the Knife tool. The wall now has triangular faces that will be hard to edit later.

Selecting loops

We saw how to create some cuts for your Mesh objects, and added a lot of new vertices and edges to it. There will come a time when we will need to select these loops for editing, like moving or scaling a vertex loop. To help with these kinds of selections, we have a very useful short cut in Blender.

Every time you need to select a loop, just press *Alt* on the keyboard, and right-click on one edge between two vertices of that loop. This way all other vertices will be selected:



New edges and faces

Sometimes, when we edit a Mesh object, we will have a hole or some vertices that we will want to connect with a new face. For this kind of creation, Blender has a tool named **Make Edge/Face**. To use this tool, we must select two or more vertices that define a new face or edge. If we select only two vertices, then a new edge will be created. But, if we select three or four vertices, a new face is created.

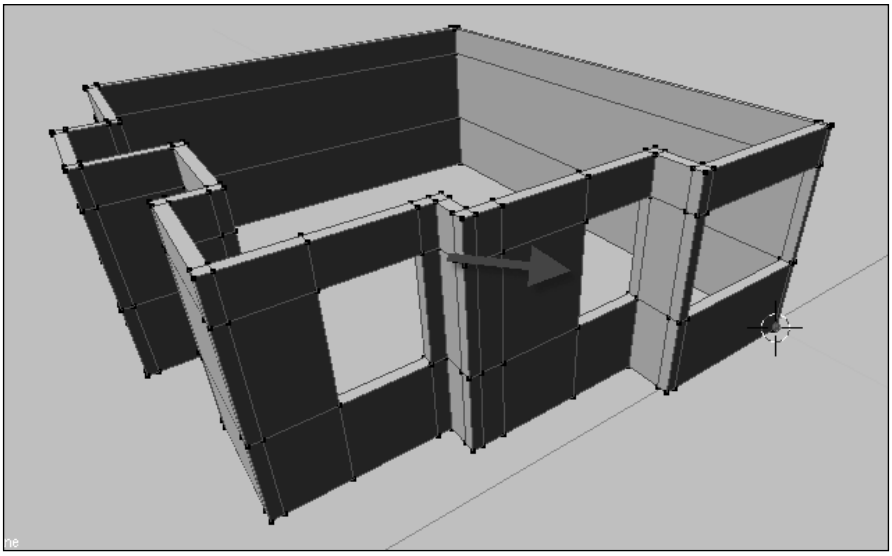
Not only when we have a hole in the mesh, but also when we have geometry with a lot of triangular faces, things can get messy. So the best option is to erase these faces and create new ones, using only four-sided faces.

Why only three or four vertices? Well, that's a limitation of Blender that only supports faces with three or four sides. If we want to make more complex planes and faces, we will have to subdivide the shape into more than one face. The fact that Blender doesn't support faces with more than four sides is actually a good thing. It will result in clean and organized models, without complex polygons side-by-side with squares and triangles. A lot of tools support faces with more than four sides, named n-gons, but without them, we will always be pushed to create organized and clean 3D models.

Blender only shows us the internal subdivision of the faces, requiring a bit more of discipline in modeling, but giving us the same results. And more importantly, there won't be any problem at the rendering; so if a lot of faces are coplanar, we won't see any differences. Two faces are coplanar when they share the same plane in space. For simple visualization they can be ignored, but for complex and more elaborate images they can start to cause problems with the object. The main problem with coplanar faces occurs when we apply materials and textures. For instance, we could select just one of two coplanar faces and apply a texture to it. During the render, we would see part of the faces with a texture, and part without them.

After selecting the vertices or edges, always in Edit mode, we will activate the tool in the **Mesh** menu in the 3D View header, by **Mesh | Edges | Make Edge/Face**. There is way to activate it with a shortcut, which is the *F* key.

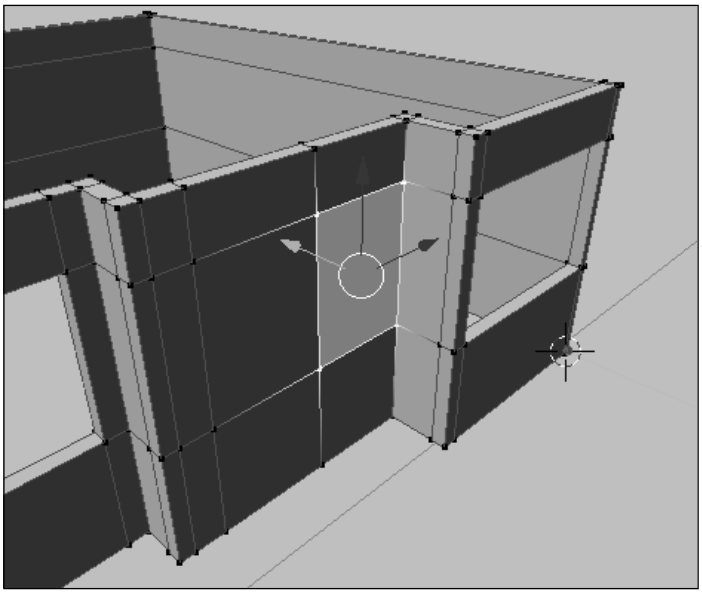
Let's see an example of how it works. The following image shows us a wall with a window opening, and we want to create a new face and close one of those openings:



To select the four vertices that define this new face, we can press the *B* key in Edit Mode and draw a Box Selection to make things easier. If you make any mistakes in the selection, press *A* to remove everything from the selection and try again.

When the faces are selected, just press *F*, and a new face will be created. It works the same way with two vertices, but only a new edge will be created. If you want to try it, select two vertices and press *F*; a new edge will connect those selected vertices.

This tool will be very useful to make adjustments in models that have little holes or need heavy editing, like windows and doors openings:



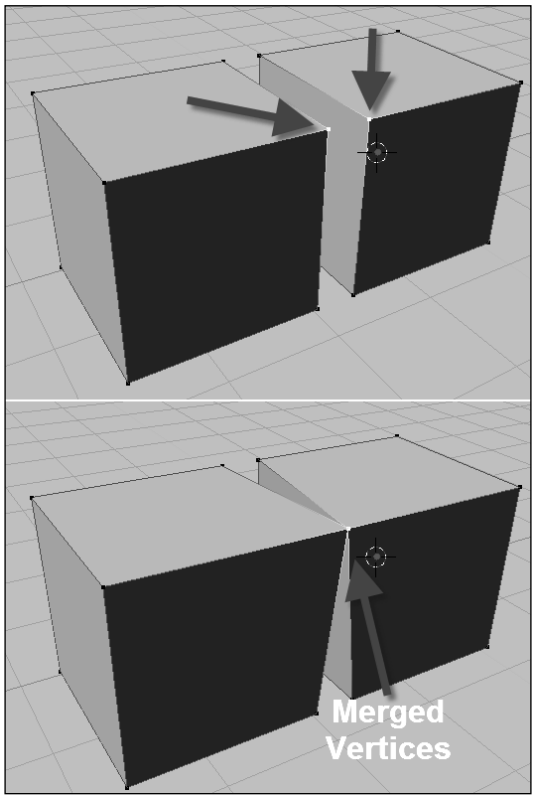
Merge

Another way of editing a Mesh object is merging existing vertices into a single vertex. This can be achieved with the **Merge** tool, which is available in Edit Mode. To use this tool, we must select any number of vertices from a Mesh, and then activate the Merge option. To do it, we have three different options, which are: **Mesh** menu in the 3D View header, using the *Alt* + *M* keyboard shortcut, or pressing the *W* key and choosing **Merge** from the **Specials** menu.

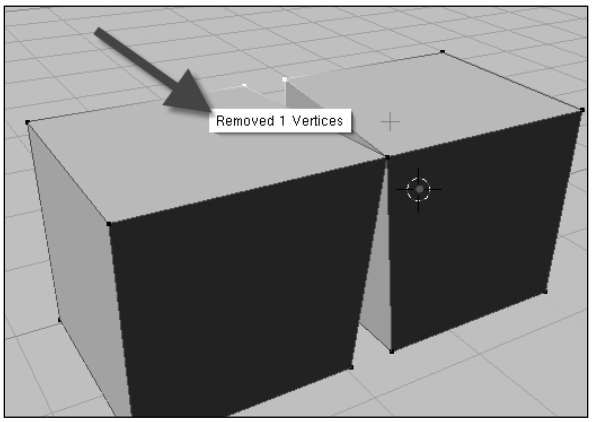
After choosing one of those methods, we will have to choose a way to do the Merge. Yes, there are five different ways of doing a Merge. These are the methods:

- **At First**: If we select vertices one by one, choosing this option will make the Merge place the resulting vertex at the same place as the first-selected vertex.
- **At Last**: Here we have the opposite of the previous option. The resulting vertex will be placed in the same place as the last selected vertex.
- **At Center**: If we choose this option, the Merge will result in one vertex placed at the middle point between all selected vertices.
- **At Cursor**: With this option, the resulting point of the Merge will be placed at the 3D cursor.

- **Collapse**: This option just collapses all selected vertices into one vertex.



Every time a Merge operation is done, we will see a message in the 3D View, which will display **Removed x Vertices**, where the **x** will be the number of vertices that are placed in the same position in space:

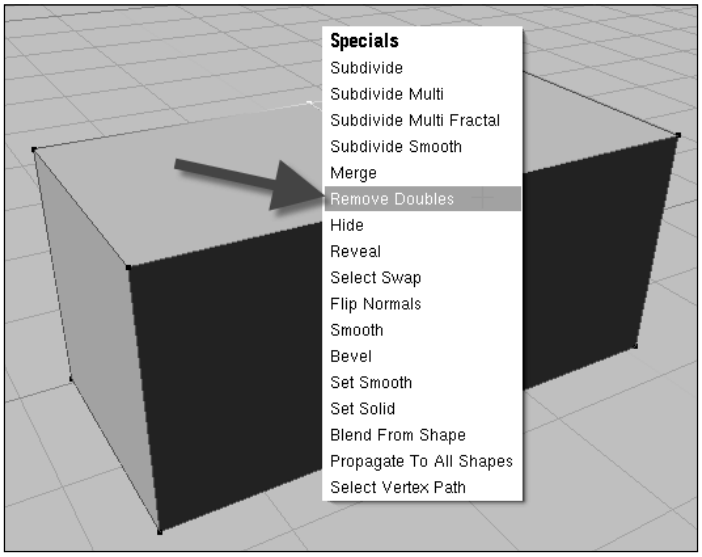


Removing double vertices

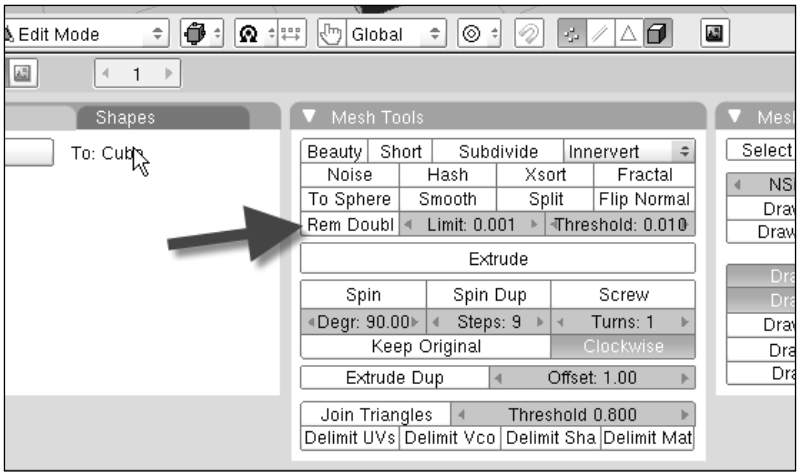
Having double vertices in a model is not a good thing, especially if we want to smooth the model. This happens when two vertices share the exact same point in space, and are not connected. This can confuse the editing process, because we will try to select and transform an edge or face, and it won't be transformed the way we want.

Some tools in Blender automatically remove duplicate vertices when they find them, such as the Merge tool that always removes double vertices. But sometimes we will need to remove double vertices manually, because they can be the result of a bad modeling edition or just caused by a wrong interpretation of an external file, such as when we import a 3D model from other formats into Blender.

To remove these double vertices, just select any number of vertices in Edit Mode and press the *W* key, and then choose **Remove Doubles** from the **Specials** menu. If any of the selected vertices are doubled, this tool will merge them into only one vertex:

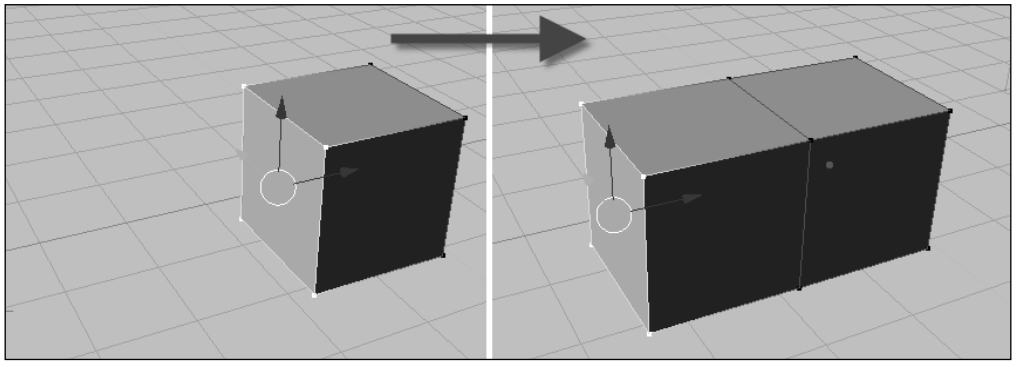


In addition to the **Specials** menu, we can activate and change a parameter of the **Remove Doubles** in the **Mesh Tools** menu. This menu is located in the **Editing** panel in Edit Mode. If it doesn't appear, just press *F9* in Edit Mode to open it. There is a button, **Rem Doubl**, which works exactly the same way as the **Remove Doubles** in the **Specials** menu. And, we have an option named **Limit**, which will determine the minimum distance between two vertices, for the Remove Doubles to merge these vertices into one:



Extrude

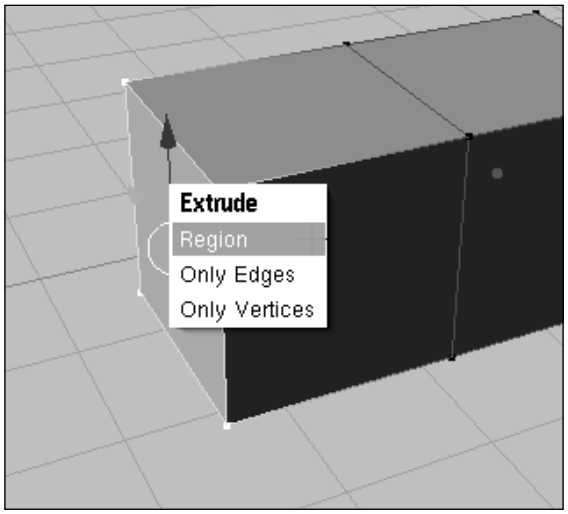
One of the main tools we will be using for modeling is extrude, which can make new geometry from selected vertices, edges, or faces. To create an extrude, the process is very simple—first, we must create a Mesh object, and in Edit Mode, select a vertex, edge, or face of this object. When at least one object is selected, press *E* on the keyboard to make an extrusion:



The extrude modeling doesn't work the same way for all kinds of objects. If we have a vertex, edge, or face selected, the result will be different. If you don't remember, to switch between selection modes, just press *Ctrl + Tab* in Edit mode.

Extrude with vertex

When we have one or more vertex of an object selected, a small menu will appear every time the key *E* is pressed. This menu can have three options, depending on the number of vertices that are selected:



- **Region**: Here we will be able to create an entire face. This option is available only if we select a full face.
- **Only Edges**: This option extrudes only the edges of a Mesh, creating new planes. We can access this option only when we select two or more vertices.
- **Only Vertices**: With this option, only the vertex will be extruded. The result will be new vertices, and a line connecting these new vertices to the old ones. This option appears for any number of selected vertices.

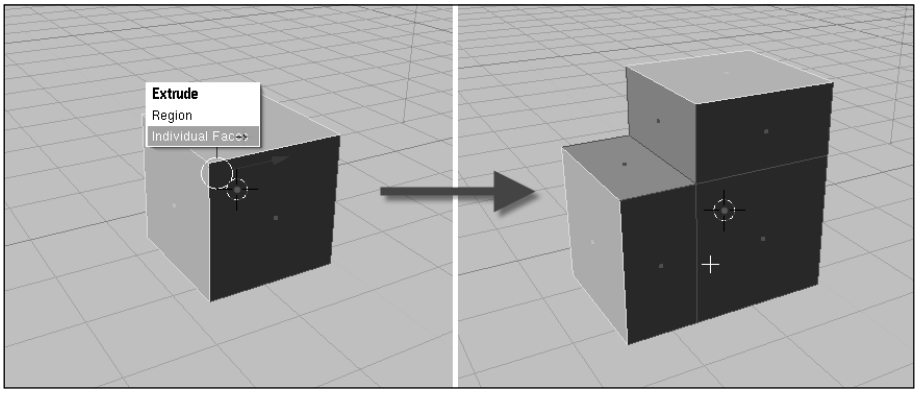
Extrude with edges

The edge mode doesn't give us as many options as the vertex. The only situation, in which a menu with some options will show up, is if we have edges that define a full face selected. Like in vertex mode, an option named **Region** will appear in the menu. Otherwise, extrude will only create new planes from the selected edges.

Extrude with faces

If we choose to work with faces, selecting a single face will enable us to create new geometry quickly, because we will select a full face and extrude it. The only situation where a menu will appear when we are working with faces is for an extrusion of more than one face. If we select two faces, there will be two options to make an extrusion:

- **Region**: This is the default extrude, where all faces are extruded in the same direction.
- **Individual Faces**: With this option, the selected faces are extruded along their individual normals, just as if they were extruded individually:

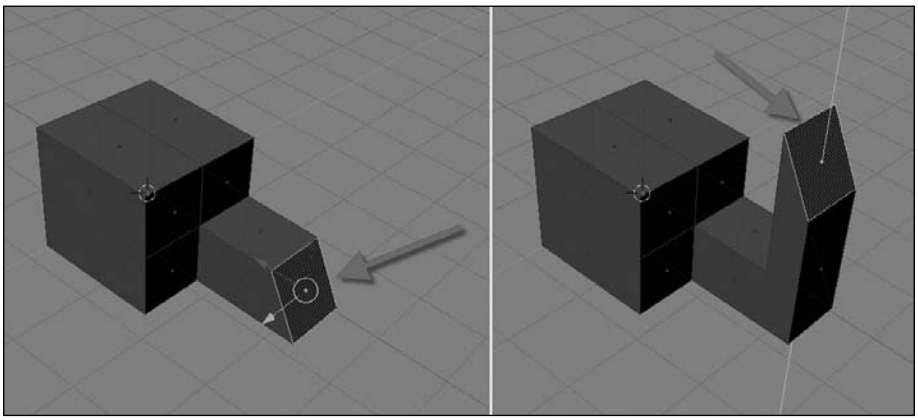


Extrude is a very powerful tool to create new geometry, but it won't do the job alone. Along with extrude, we must use other tools to edit mesh objects and create cuts, slices, and split the object. For that, there are some nice Mesh Editing tools in Blender.

Constraining the extrude

If we want to constrain the extrusion along an axis, we can do that with a shortcut key. Use this shortcut key right after you call the extrude:

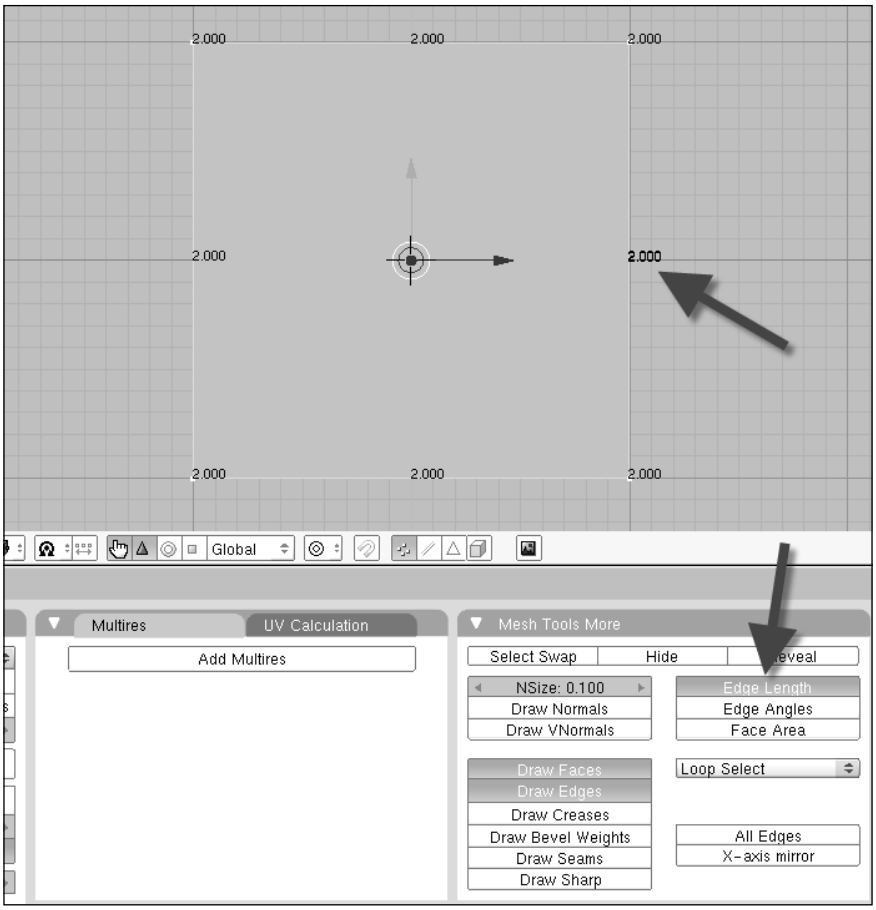
- *X*, *Y* or *Z* Key: If we press any of those keys, the extrude will be constrained to the global X, Y, or Z axis. Pressing any of those twice will constrain the extrusion to the local axis.
- Here is an example of a plane, constrained to the Z axis using the *Z* key:



Modeling example

Now that we know how to work with meshes, let's take a look at how we can transform something simple, like a plane, into the walls of a building. This example will show how to start the modeling of the building that was presented at the beginning of the chapter.

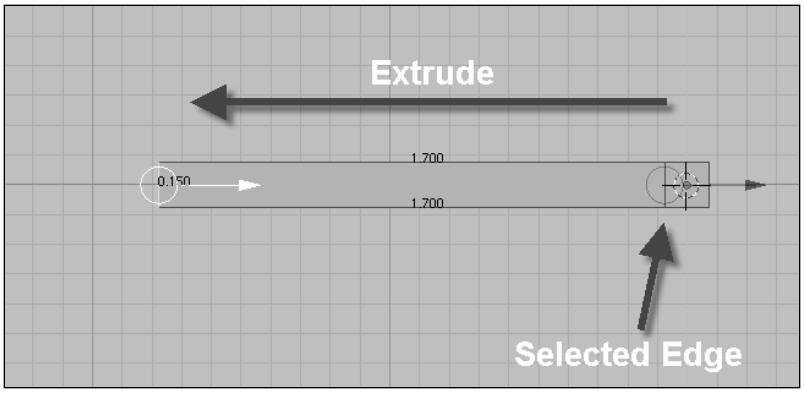
The first step is to start with a primitive shape like a plane. To work with measurements, there is an interesting option in the Editing panel—the menu named **Mesh Tools More**. In this menu, turn on the **Edge Length** option—to display the length of all edges in the 3D View:



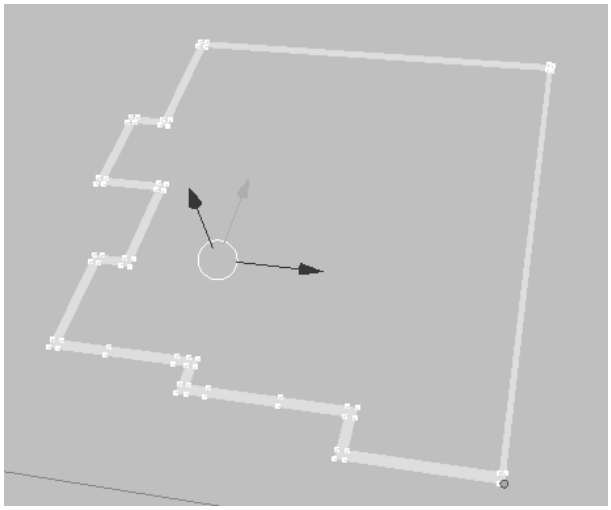
With the length of each edge visible, we can scale the plane until we get it to the right measurements. For instance, if the wall should have a width of 0.15 meters, the plane can be scaled down to 0.15. To make the transformation more precise, remember to hold down the *Ctrl*, *Shift* or the combination of *Ctrl* + *Shift* to fine-tune it.

When the plane has a length of 0.15 for each edge, we can start the modeling process. The process is basically a series of extrudes. To make the selection process easier, we can change the selection mode to Edge and select one of the edges.

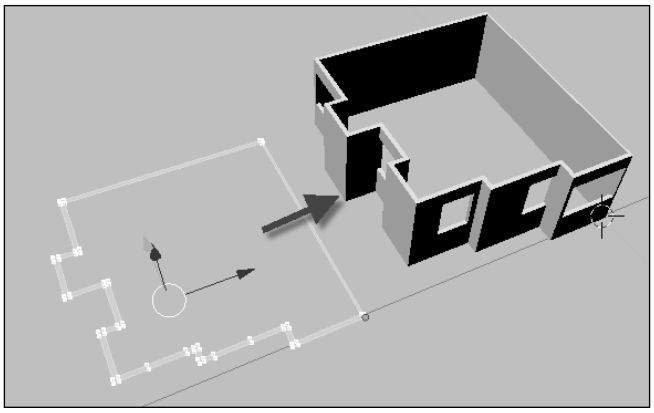
Then start to extrude this edge. Always use the *Ctrl* key to extrude these edges, and you will have the new faces snapped to the gridlines:



After a few extrusions, we will have the basis to create the walls. This was all done only with extrude:

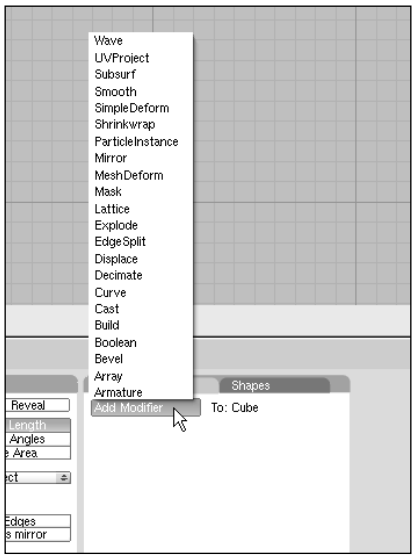


With all the faces selected, we can make another extrusion constrained to the Z axis. Along with extrude, we will have to make additional edits, to get the desired shape of the wall. The whole process will be covered in the next chapter. But most of the task is about the use of extrude:

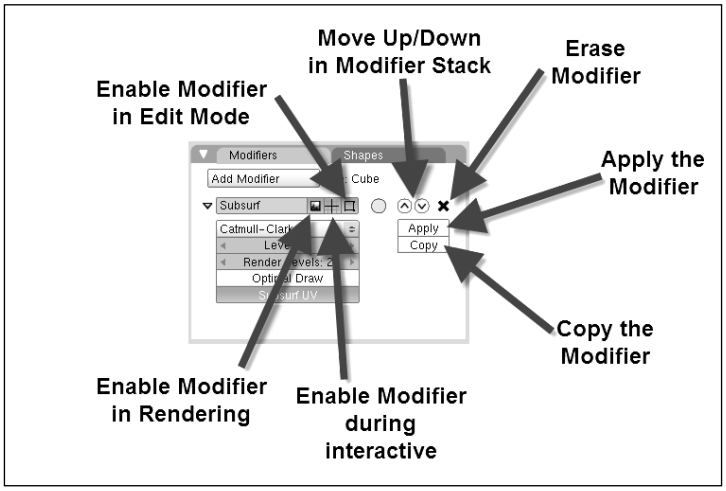


Modifiers

Besides the editing tools available in the **Mesh** menu, we have tools named Modifiers, which are grouped in the **Modifiers** menu in the Editing panel. There are many different types of modifiers, and each one can help us with specific problems. But, some of them that are very useful for architectural modeling. These are the **Array**, **Subsurf**, and **Boolean** modifiers:



All these modifiers give us a lot of flexibility in the modeling process, because we can make stacks of modifiers to combine them. Because of that, the sequence of modifiers in the stack can change the way a model is affected by them. We can add a modifier to an object either in Edit or Object mode, but some of the modifiers will only work properly when we are in a specific mode. To control and manage the modifiers, we can use these controls:

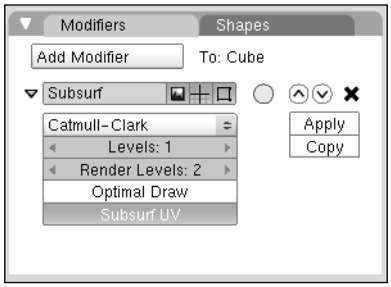


Let's take a look at four of these modifiers that will be very useful for us in the modeling process for scenarios and architectural modeling.

Subsurf modifier

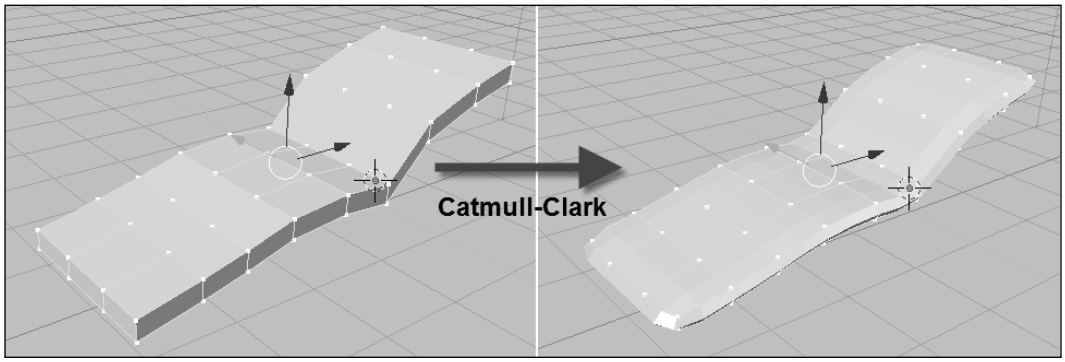
The **Subsurf** Modifier subdivides the models to make them smoother. With more subdivisions, a model that looks crispy with hard edges between faces gains some soft edges between its faces and looks more organic. This is one of the oldest tools in Blender, and largely used in character modeling. For us, this tool will be more useful in furniture modeling and landscapes.

We can add this modifier either in Edit or Object modes in Blender. When the modifier is applied to one Mesh object, a few options to control how the subdivision works will appear in the Modifiers menu:



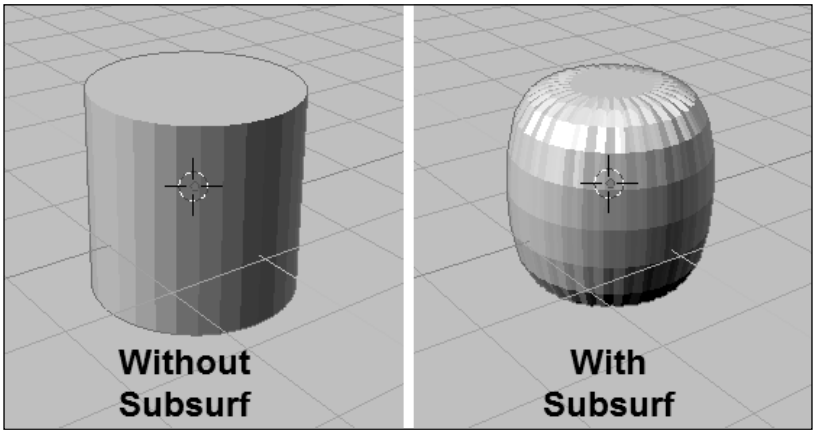
As we can see, there are two main controls that affect the level of subdivision a model has. The first one named **Levels** controls the subdivision of the model for the 3D View, and the other one named **Render Levels** controls the subdivision of the model for the rendering. This way, we can work in the 3D View with a low level of subdivision just for editing, and when the model is ready for rendering, a higher level of subdivision will create a smoother model.

There are two types of subdivision from which to choose. They are **Simple** Subdivision and **Catmull-Clark**. When we choose the first type, the model face will be subdivided but not smoothed. The Catmull-Clark subdivision will subdivide and smoothen the edges between the faces:

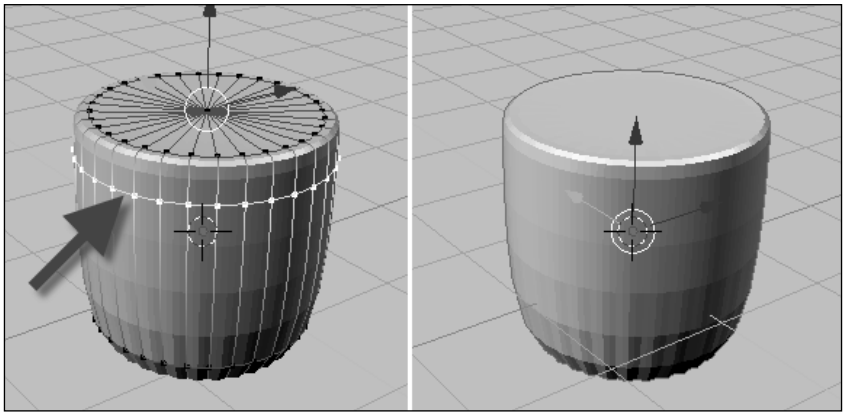


If we want to control the level of subdivision and the way all borders from different models work, we can use the Loop Cut tool with the subdivide. A new loop cut near the edge of a model will make the radius of the smoothing smaller, giving us more control over details. Let's see how it works:

- If we apply the Subsurf Modifier to a Cylinder, it will turn it into something like a capsule:

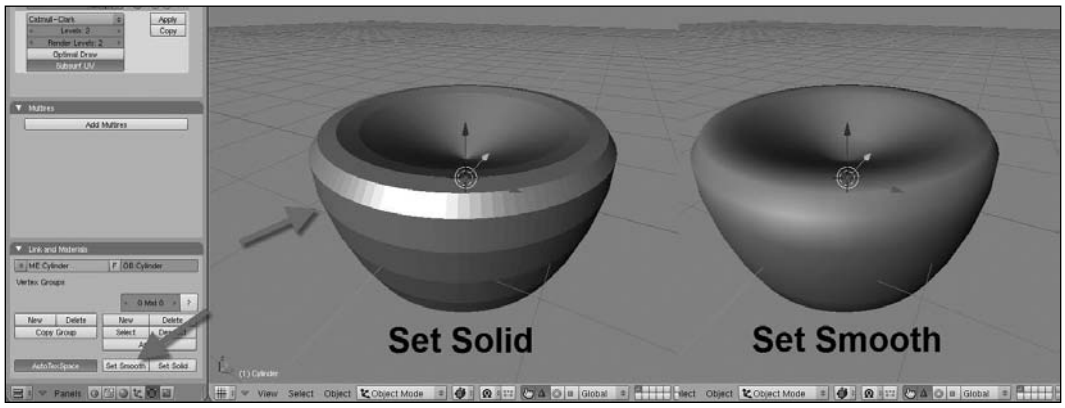


- When we select this cylinder and make a new loop cut, we can see that the border near the cut will become less smooth and the radius of the border will be smaller:



Smoothing faces

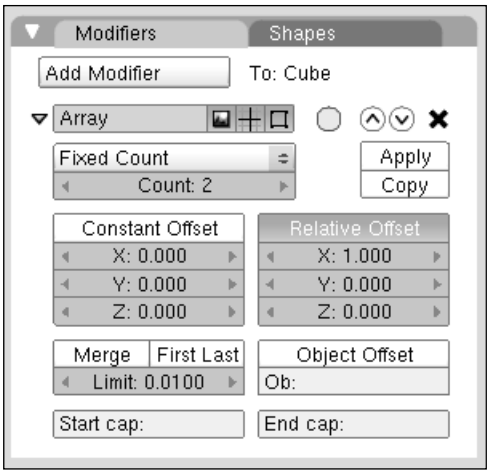
Even for subdivided models, we will still see some small faces in objects. To make models look softer in the 3D View and render, we need to turn on a tool named **Set Smooth**. To do this, select the model, and in Edit Mode, select all vertices, edges, or faces for this model. Then press the **Set Smooth** button, located in the **Link and Materials** menu in the Editing panel:



The models will look softer with this option and won't show any faces, only a smooth surface. To make the surfaces come back to a faceted look, just press the **Set Solid** button.

Array modifier

The **Array** Modifier is a great tool to create copies of objects, organizing them into lines and columns. With this modifier, we can easily spread chairs in a room, or place any object over a surface in an organized way. To use the **Array** modifier, we must select one object first, and then choose the modifier either in Object or Edit mode:



The first step to use this modifier is choosing between the three available methods to copy objects. Each of these methods distributes the objects in different ways:

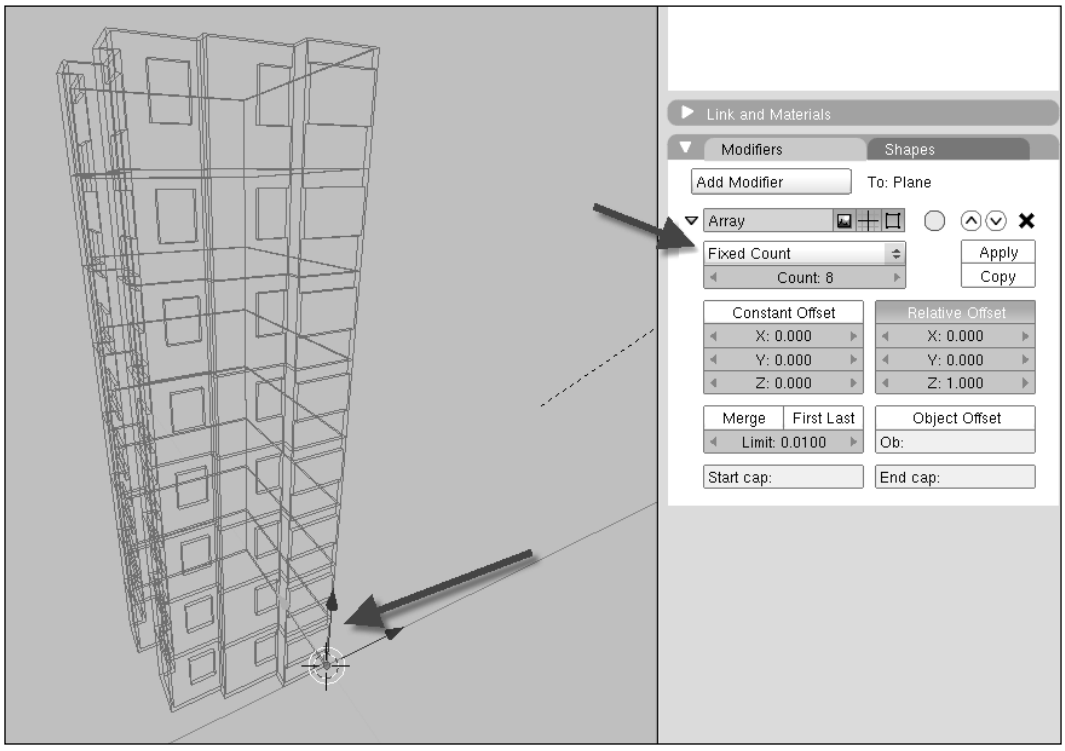
- **Fit to Curve Length**: This option distributes the copied objects based on the length of a curve. With this option, we can distribute trees or signs over a sidewalk. All we have to do is draw a curve over the surface and apply the modifier to the objects that we want to copy.
- **Fixed Length**: With this option we will use a fixed length between the copied objects. Let's take the trees example. We can make copies of tree models in a straight line and determine that between each copy we will need a distance of 50 units. Then we will set up a total range for the array, such as 200 units. This way, in 200 units, we will have tree models placed with a distance of 50 units between them.
- **Fixed Count**: Here we will make copies of objects based on the count of objects. What we have to set up here is the number of objects that we need, such as 10 objects with a fixed length of 40 units between them. It doesn't matter if we will end up with a copy range of 400 units; the number of copies that we specified will be created.

To select each type of Array method, we use the combo box placed right at the top of the menu. When we choose each Array type, the text box below changes to reflect the parameters needed for each type of Array.

Array example

Let's see how we can use the **Array** modifier to make copies of our walls. We can select the wall model, created with the extrude option, and apply the modifier.

The array can be set up to make the copies with a **Relative Offset** and with **1** unit in the Z axis. If we choose to make a **Fixed Count** of copies and choose **8** copies, it will result in eight copies of the wall, one above the other:



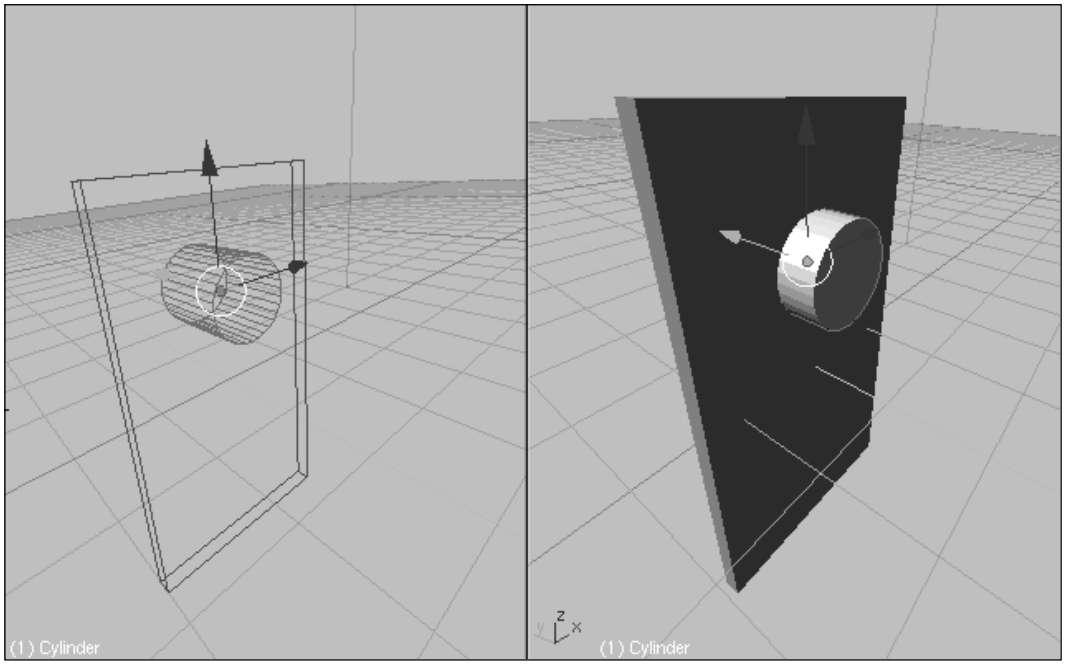
Boolean modifier

This modifier gives us a few extra options to edit and create complex meshes. If you have some experience with CAD software, you may be familiar with Booleans operations like union, subtraction, and intersection. With these operations, we can model complex objects with only a few mouse clicks, but at a price. They result in triangular faces in most cases, making any post-editing very difficult.

Let's see what we can do with each type of operation and how they are named in Blender:

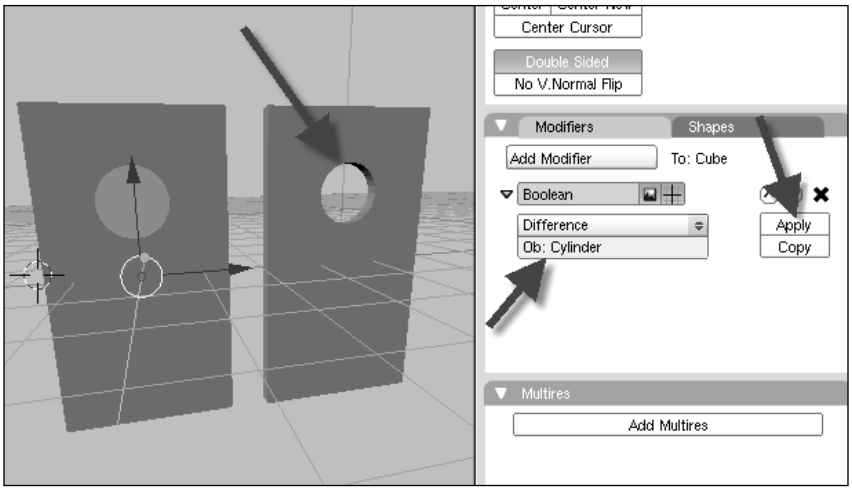
- **Difference**: With this option, we can subtract the shape of one object from another one. For instance, we can open a hole in a wall for a window. For that, we must create the wall model and another object with the corresponding shape of the hole. Then we apply the modifier to create a new object based on the subtraction of the two shapes.
- **Union**: Here we can create a new object based on the union of two different meshes.
- **Intersection**: This last type of operation creates a new object based on the area where two different shapes intersect.

From all those three operations, the Difference is by far the most used for architectural modeling. Let's see an example of how it works. We can use the Boolean modifier to open a rounded hole to an object:



To create the hole, we have to make two simple objects—the shape of the first object, which can be a cube with a few scale transformations, and the cylinder for the hole.

Place both objects in such a way that they share a common area. Select the door object and apply the modifier. Choose **Difference** and type the name of the cylinder object. To finish this operation, press the **Apply** button to make a new object based on the difference:

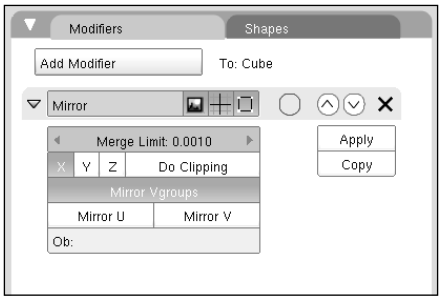


All other operations works the same way—we must select one object, choose the operation, and type the name of the second object. Just remember, only use this modifier if you are sure that this object won't need any type of post-editing.

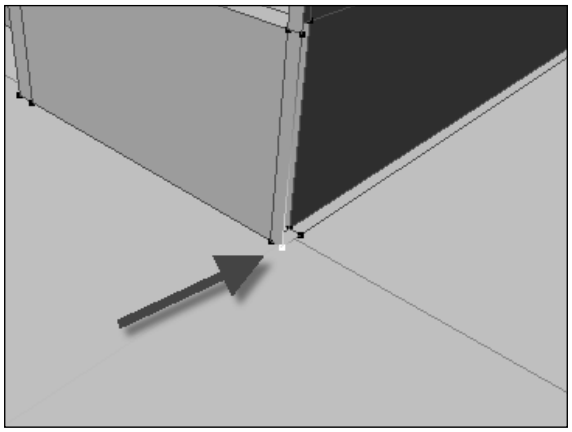
Mirror modifier

Here we have a time-saver modifier, which allow us to mirror one side of a model and make an inverted copy of the model. If your model has any kind of symmetry, it can use this modifier.

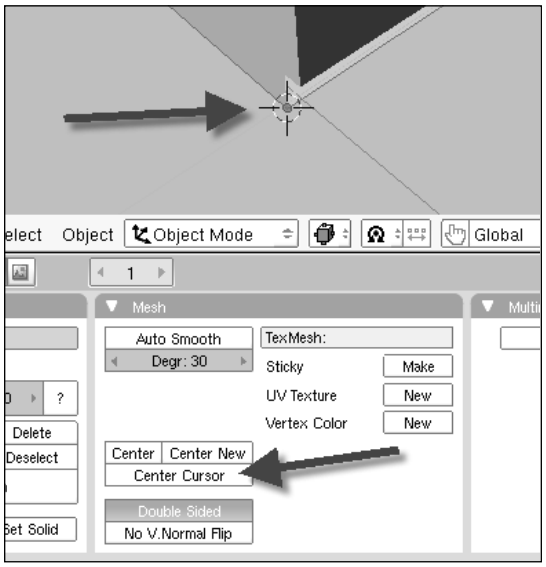
The modifier is pretty simple and easy to use. The only trick about it is the center of the mirror, which must be set at the right position, otherwise the mirror will generate the copy at the wrong position:



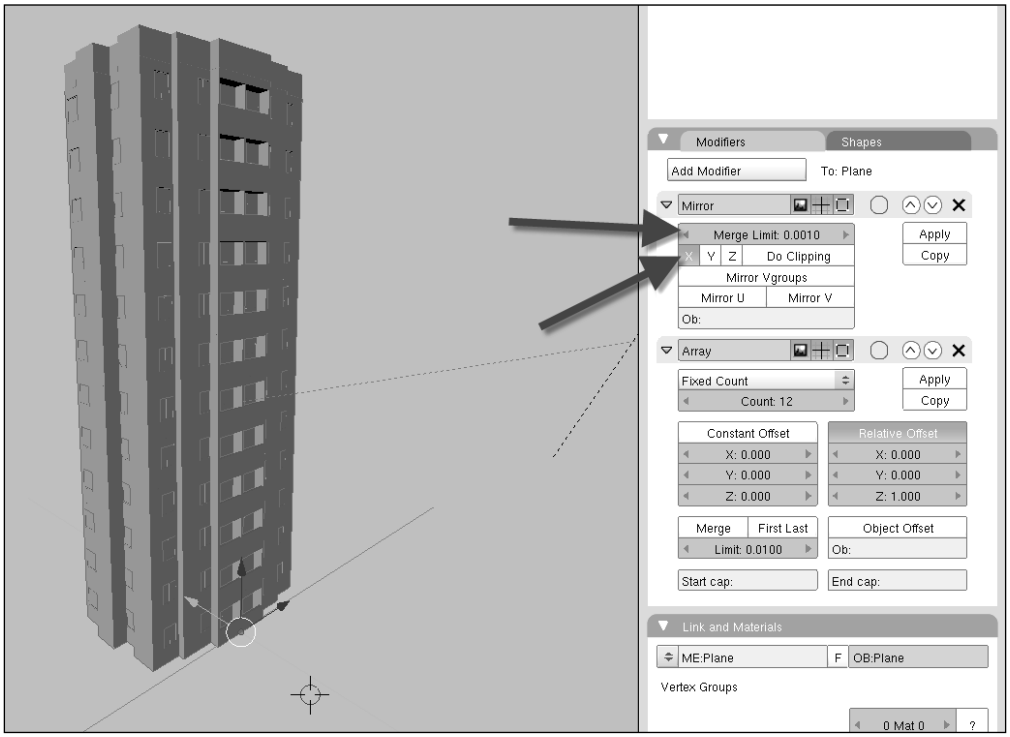
Let's see how it works with the same model to which we have applied the **Array** modifier. The first thing to do is to place the object center at the correct position. For this, we have to change the work mode to Edit, and then we select one or two vertices at the exact position where we will want the object center:



The center of the object will work as an axis for the mirror. After selecting the vertices, edges, or face, press the *Shift* + *S* shortcut key, and choose **Cursor | Selection**. It will cause the 3D Cursor to be placed at the position of the selected object. If we select more than one object, the cursor will be placed at the median point of the selection:



Change the work mode to Object and in the **Editing** panel, press the **Center Cursor** Button. It will make the center of the object to be placed exactly where the 3D cursor is:



Now, all we have to do is to apply the **Mirror** modifier and choose the axis of the Mirror. Along with the axis, we can choose the distance where each vertex is merged. It will be used if we apply the Mirror modifier.

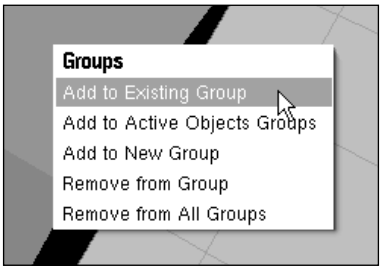
Groups

Groups aren't a modeling tool, but they help to organize complex models. The name says it all. With groups, we can gather a lot of objects and make them work as a single object. For instance, if we have a scene with a lot of chairs, instead of selecting each chair for editing, we can make a group of them and select it with a single mouse click.

We can use this example with some chair models to see how the grouping option works. There are two ways of dealing with groups; we have a keyboard shortcut, which is *Ctrl + G,* and a menu in the Object panel. With the shortcut, we can do almost everything, but with the menu we can manage all groups available in a scene.

How to create a group?

To create a group, we must select the objects that will be part of the group and press *Ctrl* + *G*. Then several options will be shown in the 3D View:

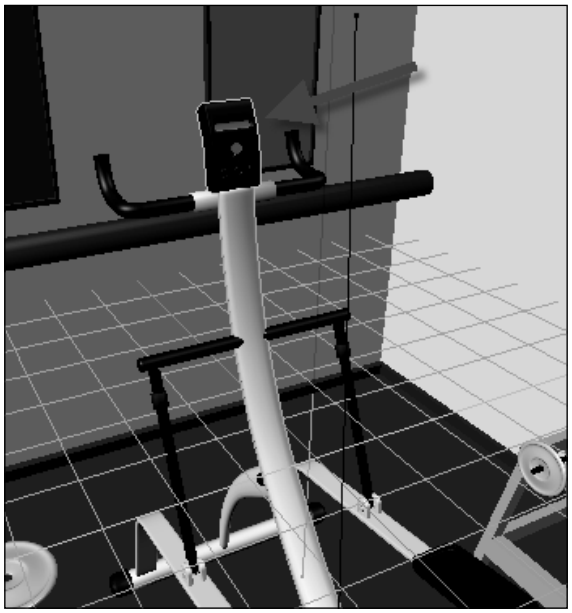


Here is what each option means:

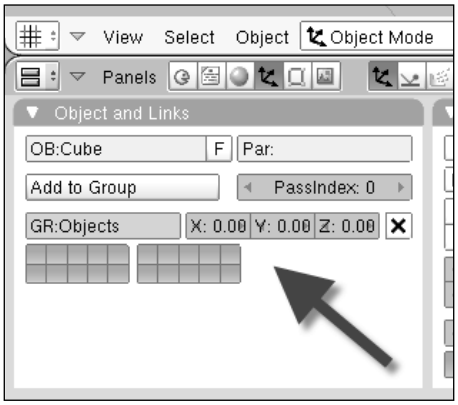
- **Add to Existing Group**: Here we can add the selected objects to a group that already exists. After we choose this option, another menu will allow us to choose to which group we want to add the objects.
- **Add to Active Objects Groups**: If we select more than one object, the last selected object will be always the active one. If this last object is already in a group, we will be able to add all other selected objects to the same group.
- **Add to New Group**: This option creates a new group with the selected objects.
- **Remove from Group**: If the object is already assigned to a existing group, this option will remove it from the group.
- **Remove from All Groups**: If the object is already in a group, this option removes the object from all groups.

The first three options will add the objects to a group. Because our scene example doesn't have any groups yet, we can choose the first one.

When the group is created, we can identify that an object is part of a group by the green line that will be marking this object when we select it:



To view all groups available in one scene, go to the **Object** panel and choose the **Object and Links** menu. It will show all groups available in the scene:

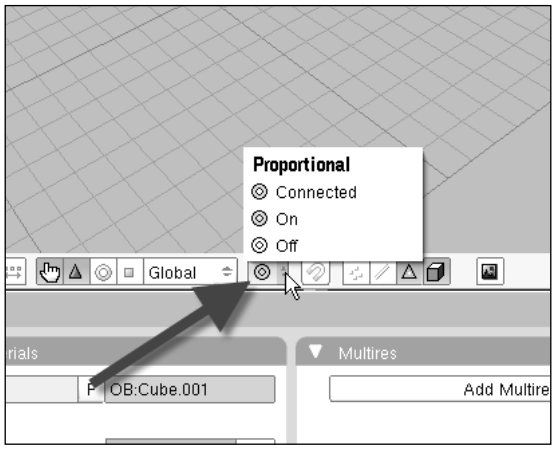


If you want to erase an existing Group, just press the **X** button right next to the group name in the **Object** panel. Erasing the group won't erase the objects, just the group itself.

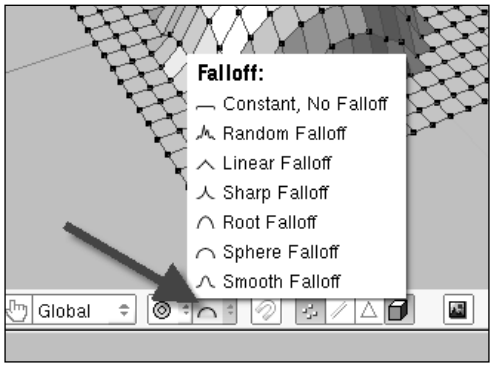
Proportional editing

To finish off this chapter, we have a tool that is very important in landscape modeling. Proportional editing allows us to transform a vertex, and transfer part of the editing to the surrounding vertices.

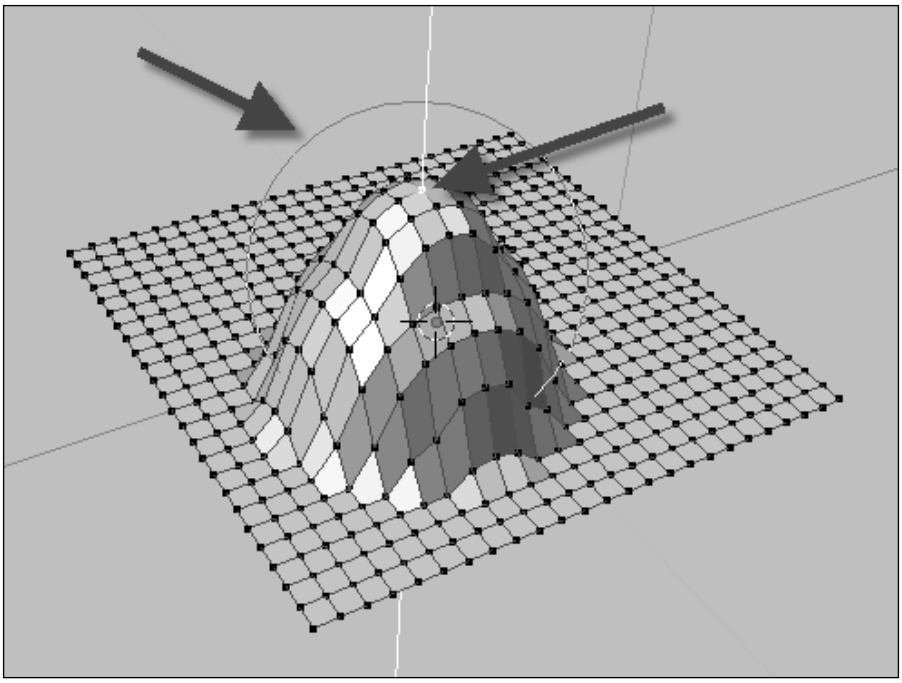
The proportional editing can be turned on with the *O* shortcut key or an option in the 3D View header. In the header, we can find the selector for different proportional editing types:



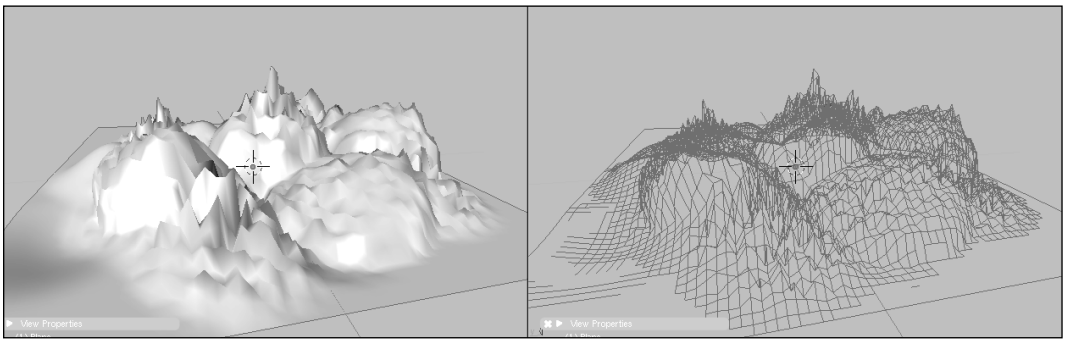
The tool works together with the transformation options. We must select one or more vertices and press *G*, *R*, or *S* to begin. When the transformation is chosen, press the *O* key to turn on proportional. Along with the option to turn on and off, we can choose from several types of proportional editing in the header:



If we use the *O* key to turn on the proportional editing, and then press *G*, a small circle will appear, surrounding the selected vertex. This is the area of effect of the proportional, which will proportionately transform all vertices inside this circle. Here is what happens if we translate this vertex in the Z axis:



With the + and - keys of the numeric keyboard or the mouse wheel, we can change the size of the circle. Here is what we can do with a few proportional editing's to create a landscape:



Summary

This latest chapter was about how Blender creates models. It was the start of your experience with modeling, but we already have learned:

- How to create objects with Blender
- What are meshes and how to edit them
- The advantages of using meshes instead of solids
- How to transform the objects
- How to extrude vertices, edges, and faces
- How to work with modifiers
- How to work with groups
- How to model with proportional editing

In the next chapter we will start to work with more options related to architectural modeling.

4
Modeling for Architecture

In the previous chapters, we learned the Blender basics, how to create objects, how to model, and more. Now, we will put all that knowledge together to create models for architectural visualization. Let's see techniques to create walls, floors, roofs, and other specific architectural elements. Some of these elements are pretty simple to create, but some require special tricks or adjustments in the modeling to be created.

Architectural modeling

Before we go any further in architectural modeling, let's see what are the differences and peculiarities of this kind of modeling from character and other types of modeling processes. If we think about the models and their characteristics, there are some points that we can list:

- The scales of the models are usually big
- All the models are based on geometrical forms; very few of them require organic shapes
- The models are usually modular, with model parts that get repeated
- The models must be created with the right proportions to show the project in the best way

These are the peculiarities of architectural modeling. As we can see, it's not much, but all those aspects must be very clear to us. The scale of the models is big because of the very nature of architecture where buildings have to be big. The only models that will require a smaller scale are those of furniture.

The shapes of models are important too, because the construction process which we use to build houses and buildings, use mostly geometrical forms such as cubes. Our models will also be strongly based on these shapes. For some people, it will make the process easier, but don't forget that geometrical forms require a lot of editing and tuning to represent architecture in the best way.

What about modular modeling? It's a common characteristic of architecture too. Some projects are built with shapes, which get repeated often in the project. This makes the construction process easier and cheaper, and we can take advantage of this by using the tools to copy models and accelerate the modeling process.

Another important thing is the precision of modeling, which is very important, because all models must be the exact equivalent of a real object. But we don't have to stick to real measures. If a wall is four meters long, and we don't create the wall with this dimension, it's going to be difficult for anybody to measure the wall in a rendered image or video.

Does this mean that we don't have to stick to real measures? Well, real measures are important, but we can make changes, as long as the proportions of the models are kept the same. If we create the model with the right proportions, not dimensions, it won't be a problem because it's the proportions that are important. These proportions will be more visible when we add furniture, cars, or people to our models.

Modeling by proportions

To model an object by it's proportions, we can use a simple example to make things clear. Let's say we had to model a table with the following dimensions:

- Height: 80 cm
- Width: 300 cm
- Depth: 100 cm

What if we wanted to change the dimensions of this table? As long as we kept the proportions same, we should be fine. We could model the same table with half it's original size, and keep the proportions the same:

- Height: 40 cm
- Width: 150 cm
- Depth: 50 cm

Or we could double the size as well. It doesn't matter how we represent the measures of this model; no one will be able to measure it with a ruler. Just be careful to keep the proportions the same, and always check if the model or object fits correctly into your scene.

This type of modeling is very useful for furniture modeling, when we have a picture of the furniture, and we use this picture as reference to model the furniture. We can follow the picture reference to keep the proportions same, and almost forget the dimensions. When the model is finished, we can scale it to make the object fit in a specific scene.

Planning is the key to success

I guess planning is important for all aspects of our lives, but for the computer graphics business, and especially for architectural visualization, it can save us a lot of time. And with more time saved, we will be able to create more images and models. With more productivity, we will get more clients and work with more projects. And with more projects, we will make more money, so here time is money too.

To make our plan successful, we have to split the visualization projects into two types. The first is the project in which we are the authors, so every decision is up to us. I don't have to say that this is the easiest project with which to work, because we have the power to make all decisions and plan everything in the modeling process. But there is another type of project, which demands even more planning. It's the project that has somebody else as author. This kind of project demands more planning because we won't be able to make changes to the project if we find any problems in the modeling.

In any case, there are some rules that we must follow to make the modeling process easier. With these tips, we can optimize our work, and create more:

- Model only what's going to be visible at the rendering
- Work with reusable objects
- Copy as much you can
- Make backups of all models
- Use file versions with dates
- Know the lighting conditions of the environment
- Know all materials and textures of surfaces

If you are the author of the project, it's going to be easier to make most of these decisions, but if you are not, talk extensively with the author, and try to make things as clear as you can.

The first rule says that we have to model only what's going to be visible, so before starting to model anything, think about how many renderings will be required for that project. When this is clear, the modeling process can start, and we will be able to select only the visible parts of a project. For animations, we will have to make a small storyboard or draw the camera path in the project as reference.

Unless you have a reason to model all objects of a project, you shouldn't spend your time modeling things that won't be shown at renderings. It's a very common mistake to model everything first, and then choose the camera view. With this kind of workflow, you will find yourself with a very detailed object, which took hours or days to be finished, and it won't be showed at the rendering. This is extremely bad for your productivity and can compromise the deadline and even the quality of the visualization, because you will have less time to work on lights and textures.

Reuse models from other projects on which you may have worked. A good library of models is very important to speed up modeling. There is even a great market for those objects, with companies that only sell models of chairs, sofas, tables, and more. For every new project, keep all small models that were used to compose the environments and organize them into libraries.

The third tip is about copying as much as possible! Hey, don't take it in the wrong way. When we say copy as much as possible, it's to make copies of our own models! Almost every model has a repetitive pattern of shapes, which allows us to create only a single piece of that shape and then create copies to build the rest of the model.

The fourth and fifth tips are related, and they are about making backups of our models. It's a very important task that can't be left aside, and most artists just don't take the required time. A good habit is to create versions for our files, with some kind of identification for dates or subject. This will help to identify the last update those files have received, and create a new starting point if the project has to be changed. A very good file naming convention is to use "project_title_year_month_day" for all files, where project is the overall name for the project and title may be a subtitle, such as living room or kitchen.

Save your files

If you have any experience with architectural projects, you know how often a project can change while it's in development. These changes can make a mess of your models if you are still working on them and the project changes. If you don't have a backup of your files with previous versions of the project, then sometimes we will have to simply start from scratch because the changes in the project won't allow us to adapt the existing geometry.

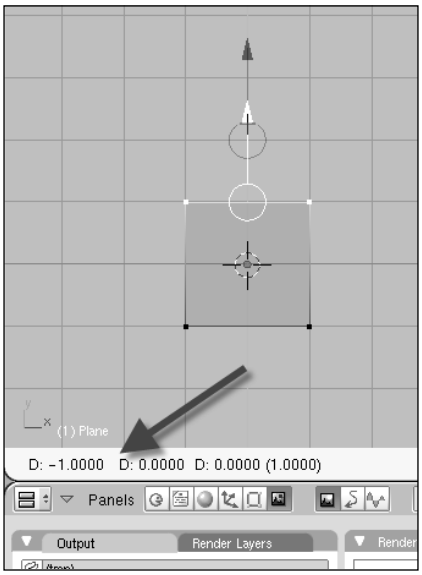
The sixth tip is about lighting conditions for the environment where the project is placed. It is very important to have a reference to start working on the simulation of that lighting. If we are the authors of the project, it's going to be easier; otherwise, ask the author about that. Where is the north? What's the geographic location of the project? How does sunlight get into the rooms through openings?

And the last tip is about materials and textures, which are important also. A lot of projects get into visualization without a clear definition of what materials are going to be used, and it can affect the modeling process. Because with a clear definition of what we will need, we can leave some details for texture maps and make the modeling faster.

I guess now it becomes very clear why planning the modeling is important and has an effect on other parts of the process, like lighting and texturing. Before a project begins, make a plan; take notes for every detail of that project. Especially if you are not the author of the project, ask the author as much as you can about the project, and ask for feedback for your questions and suggestions on the visualization.

Precision modeling

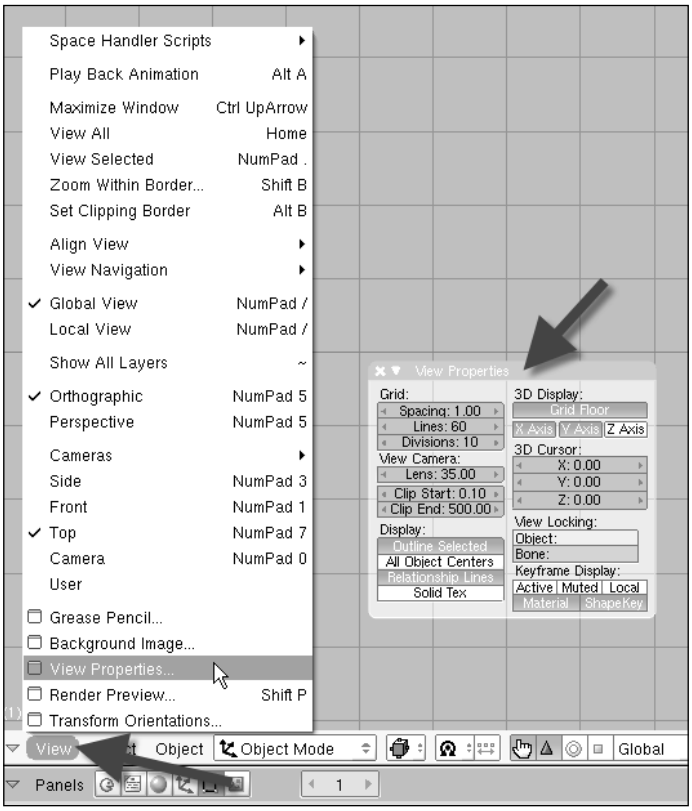
The first step to get deep into architectural modeling in Blender is to learn how we can model with precision to get the dimensions of our models into the right proportions. The easiest way to model with precision is to work with the background grid. If you have noticed, there is a grid in the 3D View, which we can use to make our models more orthogonal. To use this grid, we have to press the *Ctrl* key every time a transformation is applied to an object. There is an easy way to test it, just select one object, and press the *G* key to move the object. Before moving the mouse, just press the *Ctrl* key and hold, then move the mouse. The object will be moved, but now it won't slide across the 3D View, it will make small jumps because it's using the gridlines as a guide. Use the status bar to track the distances:



We can use the gridlines to move, rotate, and scale objects. And it works not only with the entire object, but with vertices, edges, and faces too. If we select a face and press *R*, the face rotation will use the grid lines as guide, and the rotation will be more regular. By pressing the *Ctrl* key, the rotation will be made using an increment of 5 degrees.

The default gridlines are fine for almost any kind of model, but we can customize their appearance to suit our needs. The distance between these lines are 1 unit, which could represent 1 meter, 1 centimeter, or 1 inch. It will be up to the artist to choose a scale unit with which to work, but if we need a smaller or bigger distance, there is a way to change it.

Use the **View** menu, located in the 3D View header, and choose **View Properties**. This will show you a menu with options to change the appearance of the grid:

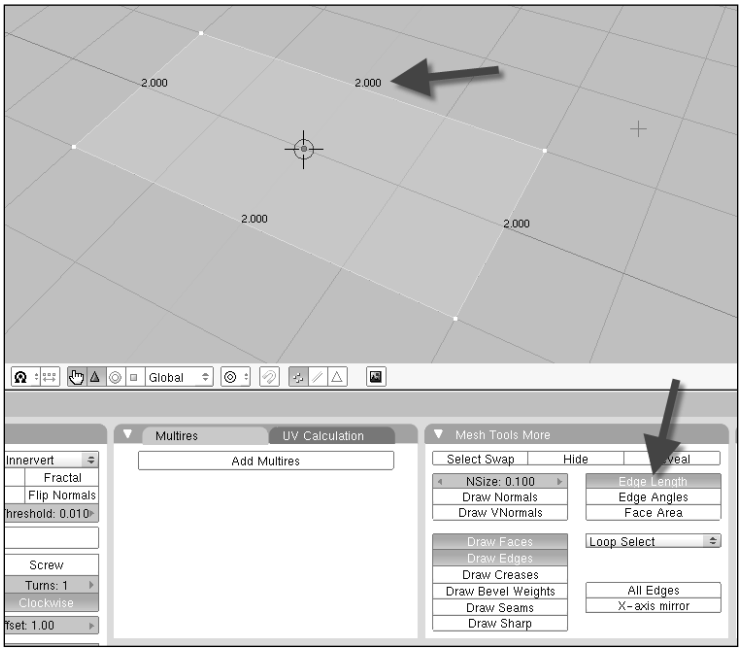


Let's see what we can do with these options:

- **Spacing**: With this option, we can set the distance between the major grid lines. The default is **1.00**, but we can change that to any value that best suits our modeling.
- **Lines**: This option controls the number of lines that are displayed in the 3D View. Only a limited number of lines can be displayed at the same time. This was created to save computer resources.
- **Divisions**: Between the major grid lines, we have some division lines to make a more precise use of the grid. We can set up the number of lines with this option.
- **Grid Floor**: This button turns on and off the visualization for the Grid. Even though it is a very helpful tool, we can turn off the Grid with this button, and make the 3D View cleaner.

Edge length

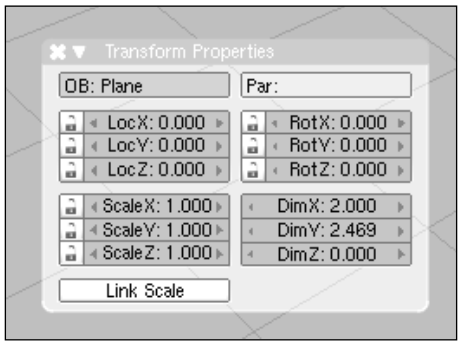
Another way to keep control over our models' dimensions is visualizing them while we create and transform the models. To do this, we have to select an object and enter into Edit mode. There is a menu named **Mesh Tools More**, which has an option named **Edge Length**. If we turn this option on, every time we select an edge, its length will be displayed:



This is a great way to check out the distance between two vertices very quickly. If you want to keep track of all distances while modeling, just keep this option always turned on.

Transforming with precision

Another way of working with precision on Blender is applying transformations with numeric values. We often use this in CAD drawing, when we need to move an object just four units on the X axis. To perform this kind of operation in Blender, we must use the **Transform Properties** menu, which allows us to give numeric values to all transformations. This menu can be opened with the *N* key or in the **Object** Menu, in the 3D View header:



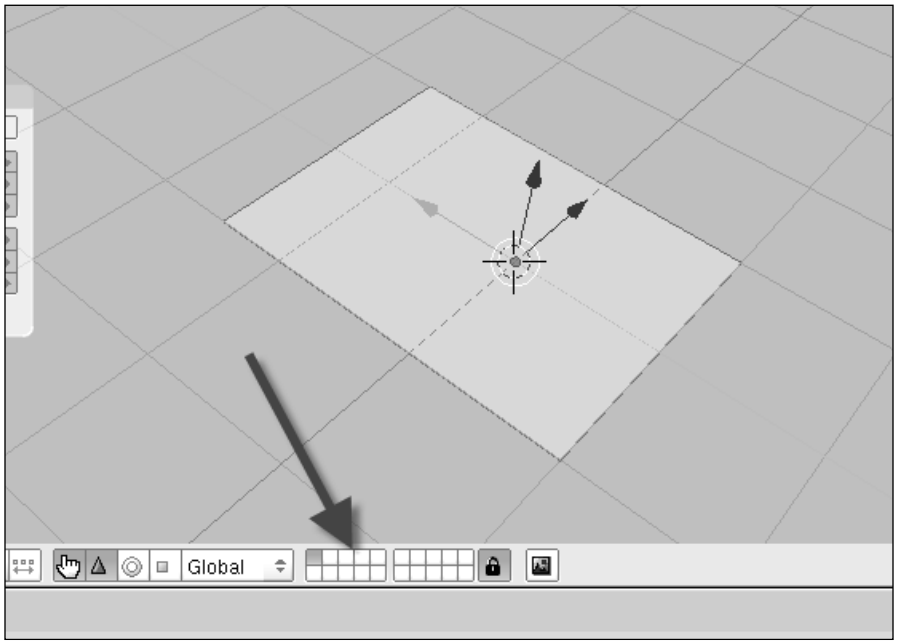
A good thing about this menu is that it can be opened from almost any Blender window. If the window allows us to draw or manipulate anything, it has a **Transform Properties** menu. Let's see how it works. Just select one object and press the *N* key. Right after the menu appears, just type the new value for the transformations in the text boxes:

- **Loc X/Y/Z**: This option controls the position of an object. We can set up a translation with a numeric value here. If we type **3** in the **Loc Y** field, the object will move three units in the Y axis.
- **Rot X/Y/Z**: Here we have the rotation controls. Just like the Loc option, if we type **-30** in the **Rot Z** field, the object will rotate -30 degrees over the Z axis.
- **Scale X/Y/Z**: This works much like the previous options, but here we can control the scale of objects.
- **Dim X/Y/Z**: This looks a lot like the Scale, but the difference here is that what gets changed is the model bounding box.

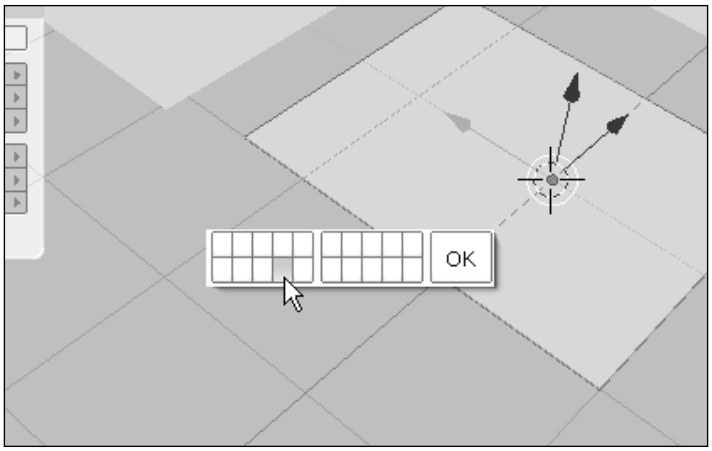
If we know that an object won't have any changes in a particular transformation, there is a way to lock all transformations for that object. This menu has small locks, right next to all transformation options. If we turn the lock on, we won't be able to apply any kind of transformation for the selected axis. If you know that your object is done, and placed at the right position, it's a good idea to lock it.

Layers

Almost all modeling packages have some kind of layer system to let artists organize their environments. In Blender, it's no different. We have a layer system which lets us control when an object should be or not be visible. The layer control buttons are located in the 3D View header:



There are twenty layers to use and place objects, and the use of these buttons is very simple. To put an object on a different layer, just select the object and press the *M* key, and a small menu will appear, where we have to choose a layer to which to send the object:



If we want to turn on multiple layers, we must hold the *Shift* key while clicking on the layers buttons. This way, more than one layer can be turned on and off.

Layers for backup

We can use Layers to make backup of models inside a scene. If we have to make a very complex edit into a model, it's a good practice to make a copy of this object and put it into another layer. If the editing doesn't work or the project changes, we will have a backup to which to go back. To copy an object to another layer, select the objects, and press the *Shift* + *D* keys. Place the duplicates somewhere, or press *Esc* to use the same location as the copied objects. When the copies are placed and still selected, press the *M* key, and choose another layer to which to move them. Turn off the layer and we will have a backup copy.

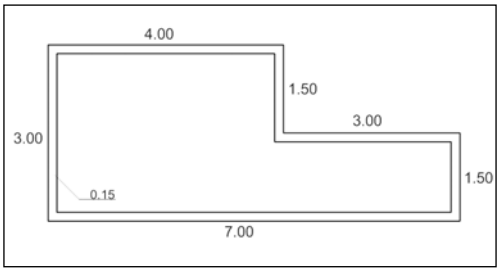
Modeling in practice

Now let's get our hands dirty! It's time to make some architectural models. To make things easier, we will split the modeling process in different parts. In the first part, let's learn how to deal with walls.

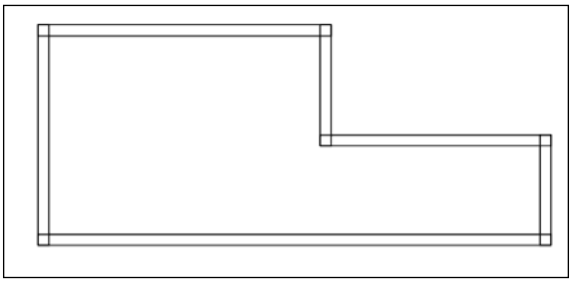
Walls

What's the best way to create walls? Well, there isn't one best way. What we have are different techniques that can be used to create this kind of object. Because each project has different needs, we have to analyze the project and choose the best way to model.

In most cases, starting with a simple shape like a plane can create very good results. If we start with a plane, there will be a limited number of vertices for us to edit and manipulate. This way, our modeling will be easier and will be done with little trouble. With this plane, we will build a base for our walls. When the base is created, it's just a matter of extruding the faces to create the rest of the walls. To see how it can be done, let's model these walls:

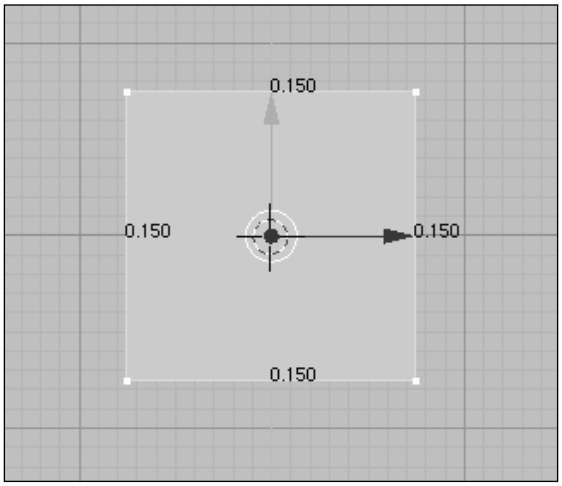


Because we will be using planes to model, it's a good practice to split all parts of a plane into rectangular parts. This way it will become clear what faces we have to create. If you don't have much practice, don't worry, we can do that even on a piece of paper if you don't want to use the computer. It's going to be a sketch of the model, with the divisions:

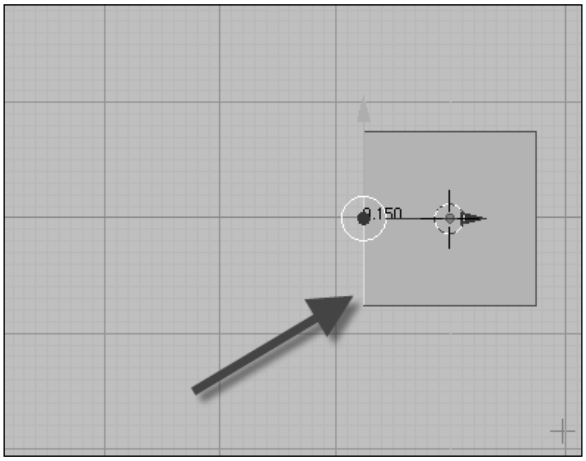


See that this top view doesn't have the openings marked, so we won't be concerned about them yet. The first step is to create a plane, which will have the dimensions of a corner for the room. According to the plan, the wall should have a width of 0.15. Then we have to create a plane, with 0.15 on both sides.

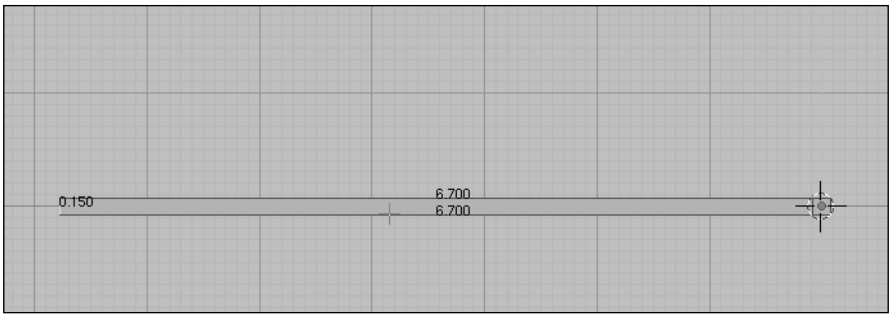
To make things easier, change the work mode to Edit, turn on the **Edge Length** option on, and with the scale transformation, resize the plane until it gets a width of **0.15**. If you need, hold the *Ctrl* key to use the grid lines:



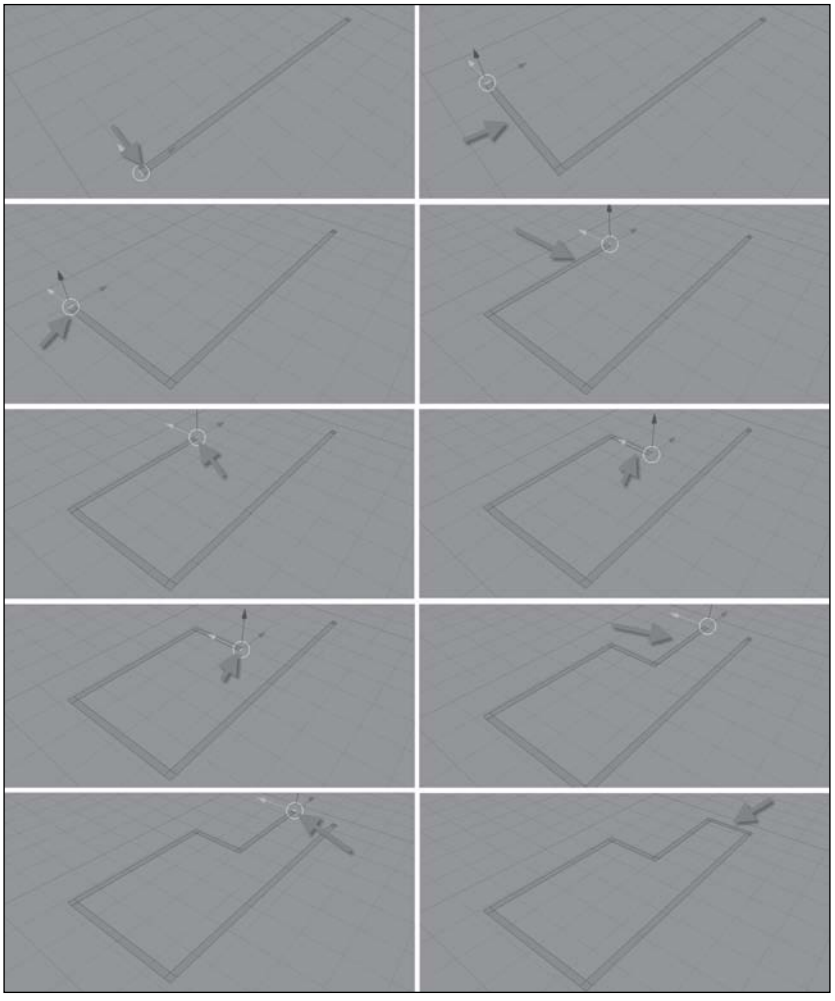
With the right dimensions for the plane, we can start to extrude an edge to create the other parts of the walls. While in Edit mode, change the selection mode to Edge with the shortcut *Ctrl* + *Tab*, and select one edge of the plane:



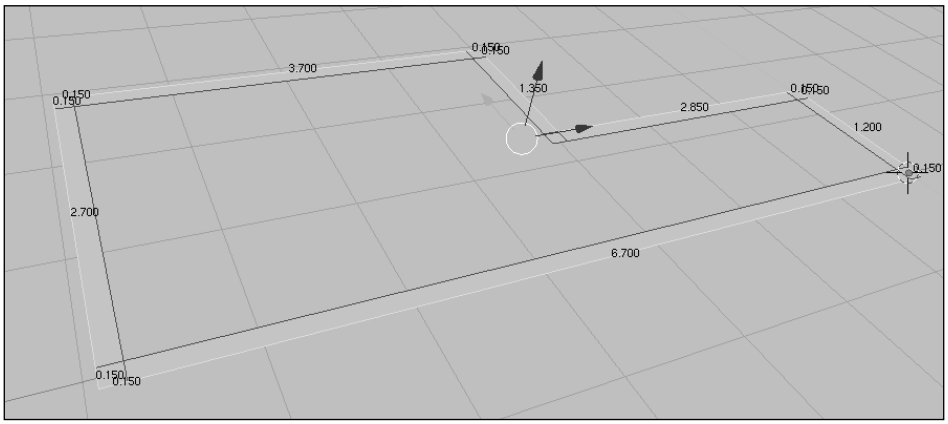
Now press *E* and extrude the edge to create new planes. To help this process, just hold the *Ctrl* key to use the grid lines, and a new face will be created with the right alignment:



When we reach the right dimension for the plane, create another small plane and follow the extrudes shown in the following image. It is a series of ten extrudes:

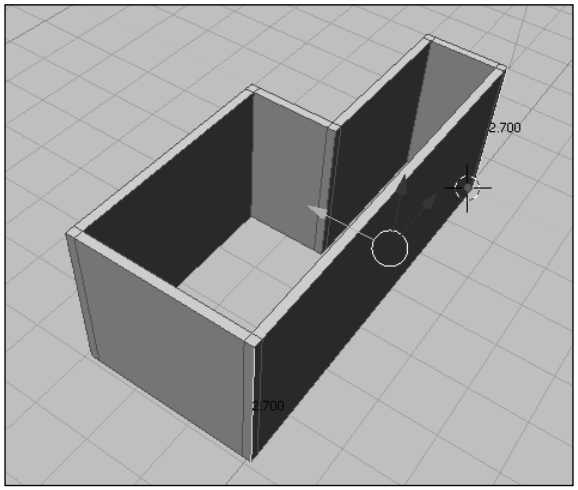


By the end, we should have something like this:



When we get all the required planes, it's time to remove any duplicated vertices. Press the *A* key to select all objects. Then the press the *W* key and choose **Remove Doubles**, to remove all duplicated vertices.

Now, it's time to extrude all faces to actually create the walls. With the faces still selected, press the *E* key to extrude. Hold the *Ctrl* key to use the grid lines, and place the upper faces in the right position. Let's say that these walls must be **2.70** units tall:



That's it, now we have our walls, and the project is in 3D! This may be easy, but things are going to start to get more difficult. Walls without openings are very easy to create, but when we start to work with openings, we have to plan even more to get the work done.

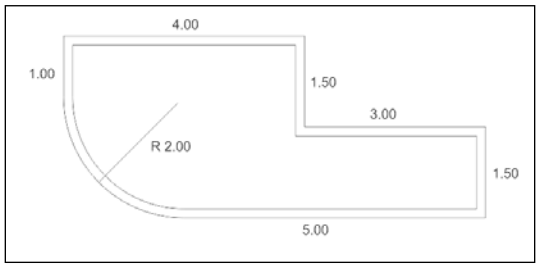
Top View

Make sure to create your models in Top view. Just press the *7* key on the numeric keyboard.

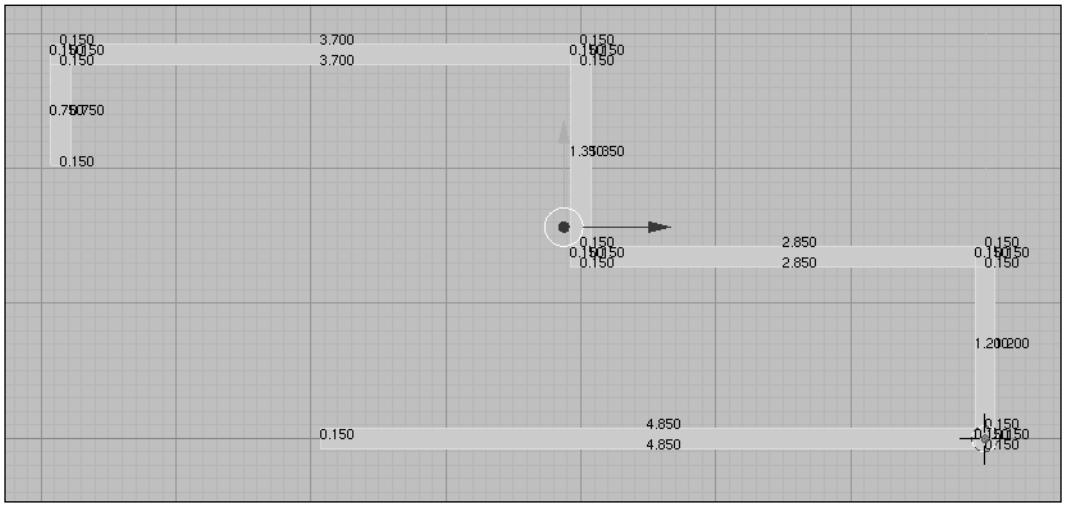
Rounded corners

And when we have a rounded corner? How we can create a wall like that? Well, to create a wall like that, we must use a tool in Blender named Spin, which can create extrusions of an edge based on a rotation. The first thing we have to know, before doing anything else is the radius of the rounded corner, otherwise it would be very difficult to create the arc required for this wall.

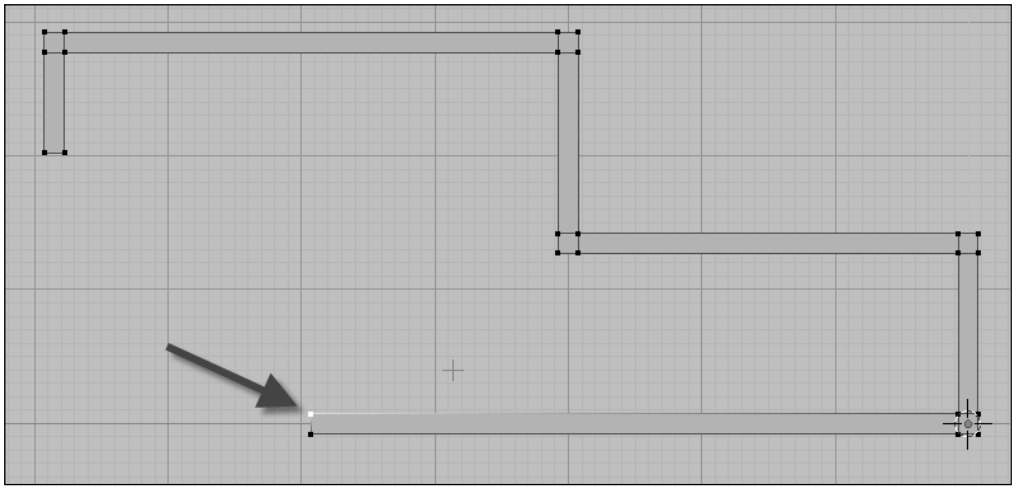
Let's say that we have a wall with a rounded corner, and a radius of **2**:



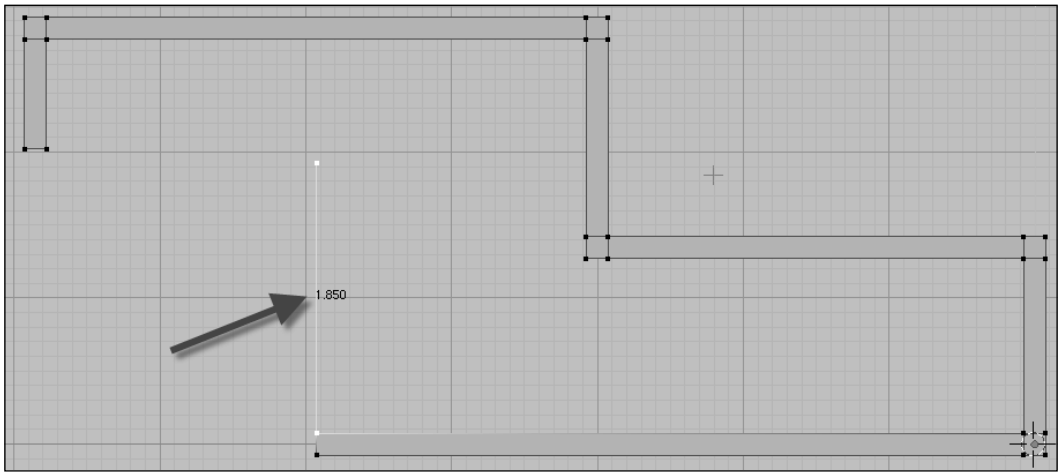
To create a wall like that, there is a little trick with the Snap and the Spin tools. First, create the planes, until we get to the beginning of the rounded corner:



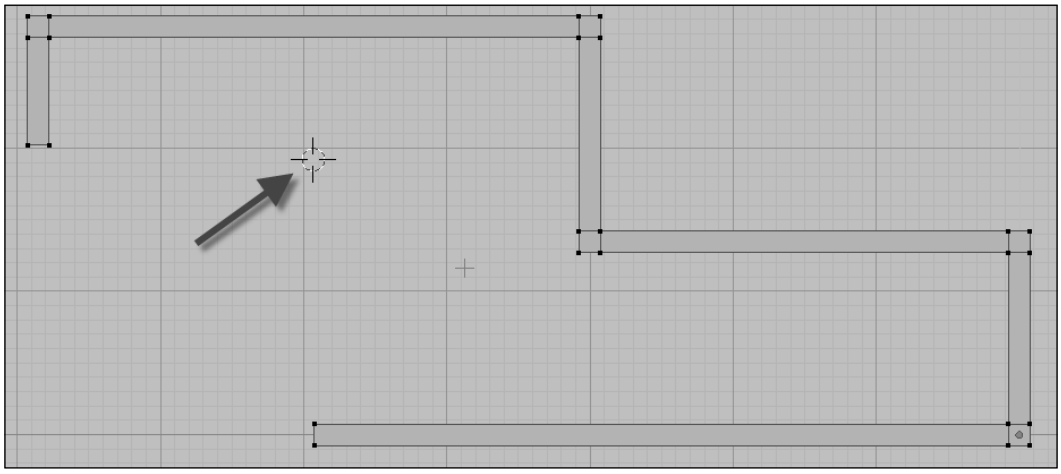
The trick—to draw the arc, we will be using the Spin tool, which takes an object and rotates it around a point. This point is determined by the 3D Cursor. So, we have to find a way to place the 3D Cursor at the center of the arc. This can be done with the Snap tool. But before we use the Snap tool, there is one thing that we must do. If the selection mode is not at vertices, then change it to select just the vertex pointed to in the next image:



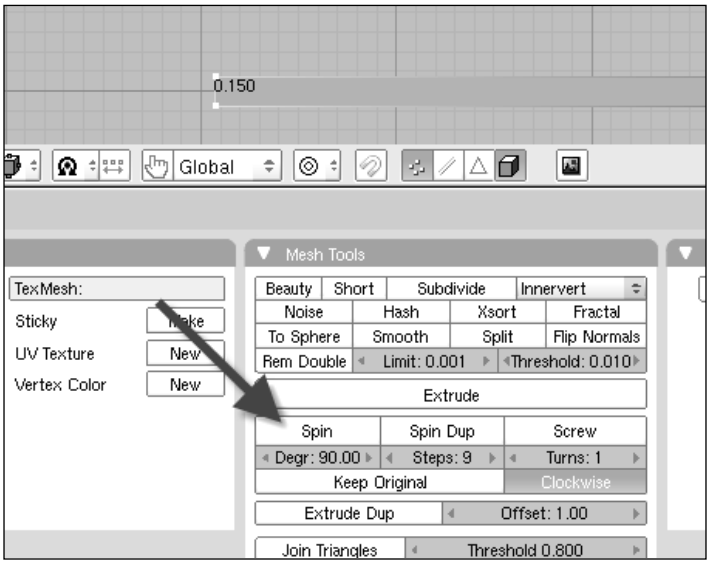
When this vertex is selected, press the *E* key to extrude only this vertex. Hold the *Ctrl* key and place the new vertex at the center point for the arc. To do that, use the radius distance to move the vertex. For instance, if the rounded corner has a radius of **1.85** meters, then we have to move the vertex exactly **1.85** units. This way, we will have the vertex placed at the center of the arc that represents the rounded corner:



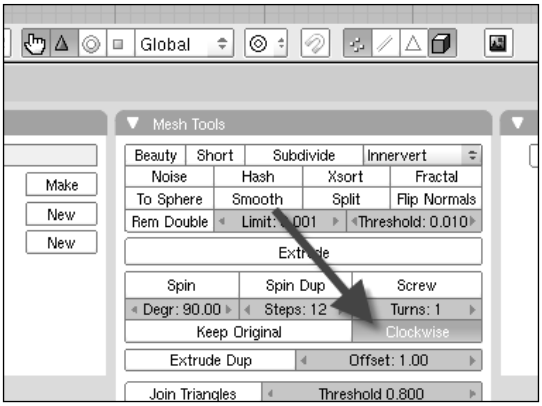
With the new vertex still selected, press *Shift* + *S* to activate the Snap, and choose **Cursor | Selection**. It will make the 3D Cursor jump to the position of this vertex. Now that it's placed where we want it, let's make the Spin. The new vertex and edge that we created to place the 3D Cursor won't be necessary now. Change the selection mode to Edge, select this edge, and press the *X* key, and choose **Edge** to erase it:



The Spin tool works with a selected object, which may be a vertex, edge, or face. When this object is selected, we can set up the **Spin**, which is located in the **Mesh Tools** menu in the Editing Panel:

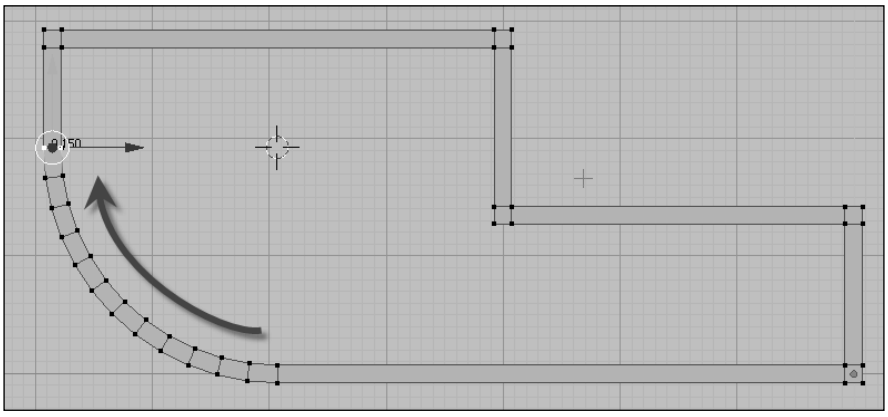


There are two parameters that we must set up, which are **Degrees** and **Steps**. The first one determines the rotation angle for the Spin. For us it will be a **90** degrees rotation. Besides that, we can tell how many steps our rounded corner will have in the **Steps** parameter. If we put a high number, a more perfect corner will be created. Try to use values above **12**, and if it's not enough, raise it until you get satisfied, but remember that it will make the model heavier, with a lot more vertices. At the bottom of this menu, we have a button labeled **Clockwise**, which determines the direction for a Spin rotation. If this button is not pressed, then we will have a counter-clockwise rotation:



Make sure you are in the Top View, because with the Spin, all faces are created at a perpendicular plane to the view, so it's very important to be in Top View. Otherwise, the new planes won't be created in the right position.

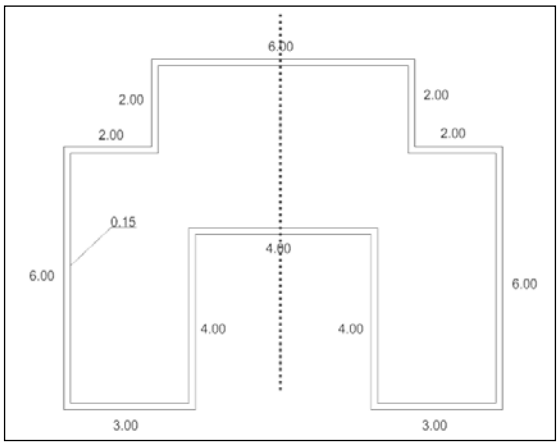
Now, press the **Spin** button and the rounded corner will be created. Remember, if the arc doesn't get as rounded as the project demands, increase the **Steps** value and it will be more rounded, but with more geometry:



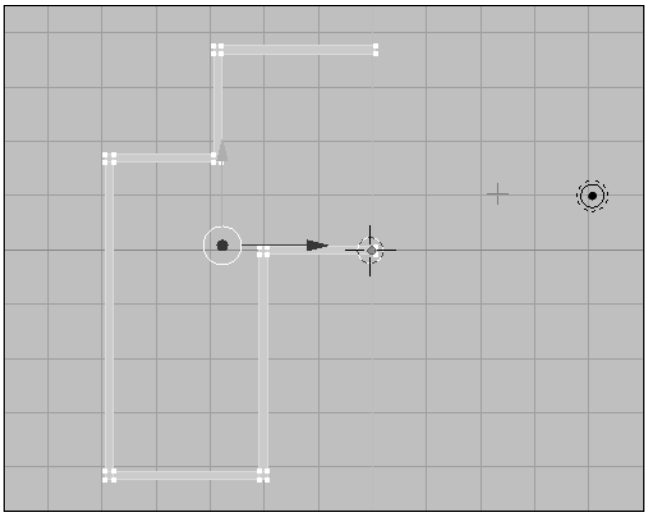
Symmetry

Very often, architectural projects have symmetrical parts where a section of a project is mirrored to create another part. It's a very good practice that makes the visualization process faster, because we will need to model only half the project, and apply a mirror modifier to create the rest.

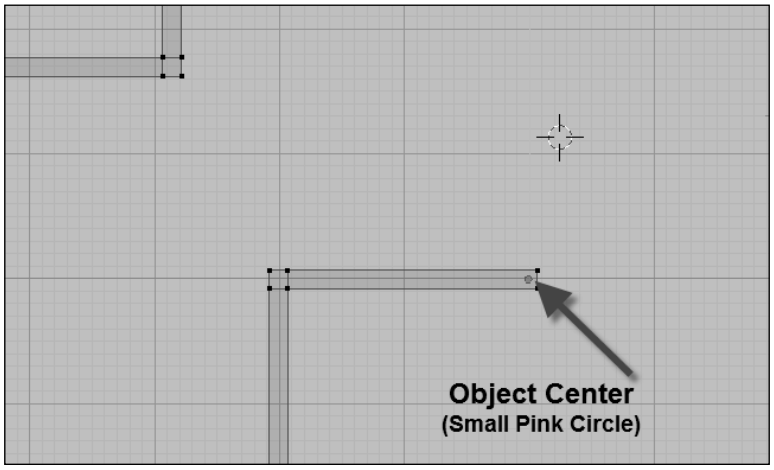
Suppose we have a project just like the one shown in the following image:



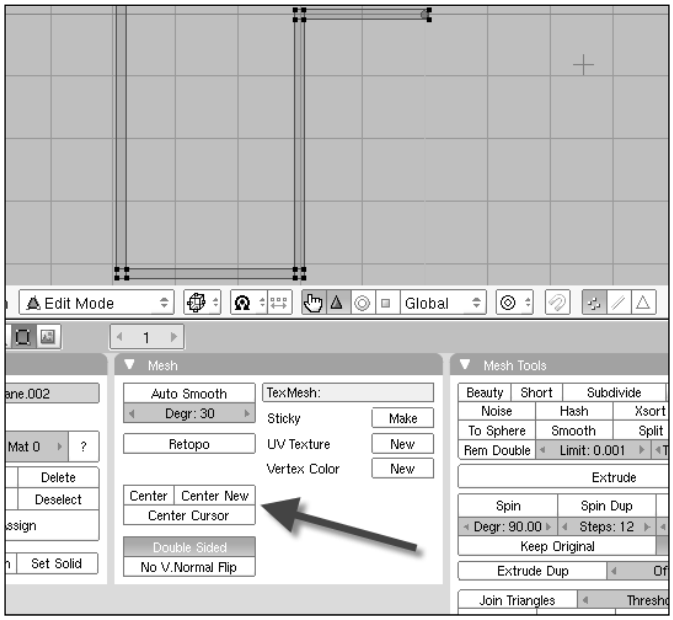
We can see that the right side of this project is symmetrical to the left side. If we model only one of them, the other side can be mirrored. Let's see how it works. We have to first model one side of the project. Here we have the left side of the project with all walls created:



The trick to using the Mirror modifier is to place the center of the object in the right position. Because the modifier uses the center to place the rotation, which happens in the mirroring process; it is created around the center point. Make sure the center is at the left or right corner of half the walls. If you haven't noticed, the center of the objects are represented as a small pink circle:



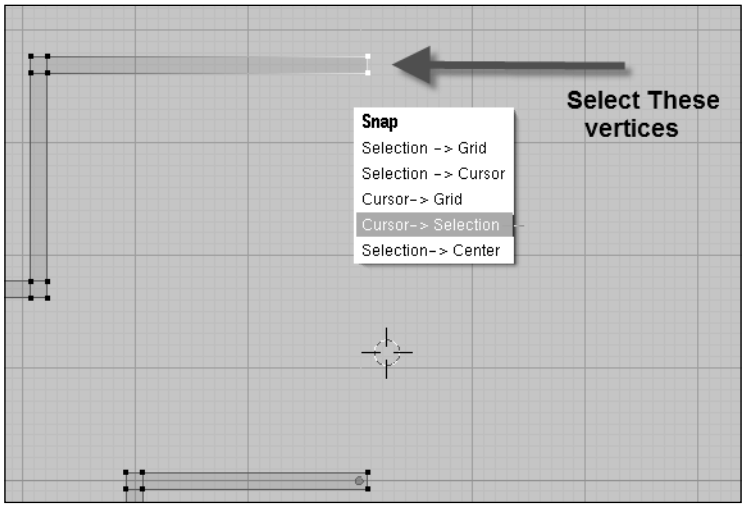
To change the position of the center point, we have an option in the **Mesh** menu that manipulates the center of objects:



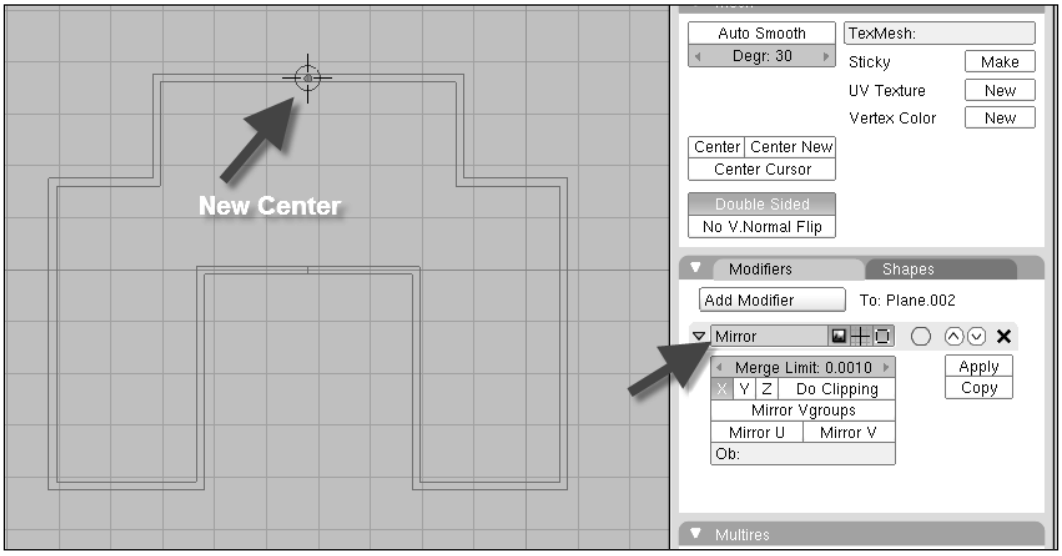
In this menu, we can move the center point in three ways, which are as follows:

- **Center**: Here we can place the object's geometrical center at the object's center position. In Blender, the geometrical center of an object can be in a different position than the mesh center. For instance, we can have a cube where the geometrical center is exactly at the mid-distance, between all vertices, but the mesh center is placed far from the cube. This button will align both centers at the same location.
- **Center New**: Here we can place the object center at the object geometrical center position.
- **Center Cursor**: The object center will be placed at the position of the 3D Cursor.

The option that will be most useful for us is the last one, which can place the center at the location of the 3D Cursor. Here is how it works: select an edge, or two vertices that define an edge. Press the *Shift* + *S* shortcut to call the **Snap** menu, and choose **Cursor | Selection**:



This will make the 3D Cursor jump to the center of our selected edge. When the cursor is placed, press the **Center Cursor** option. Make sure the work mode is set to Object, because this option only works in Object mode:

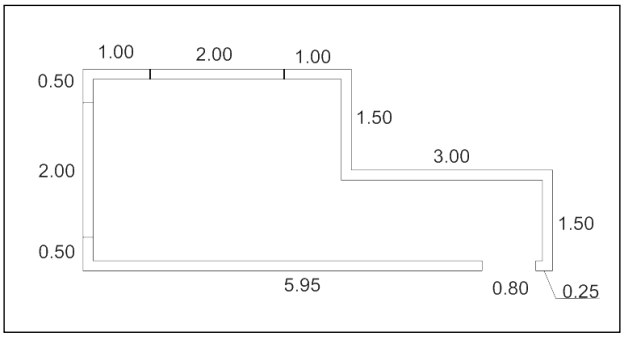


Now all we have to do is apply the **Mirror** modifier and choose an axis for the mirrored object. If the project needs another mirror, just apply more modifiers and change the mirror axis, so that more symmetrical parts can be created.

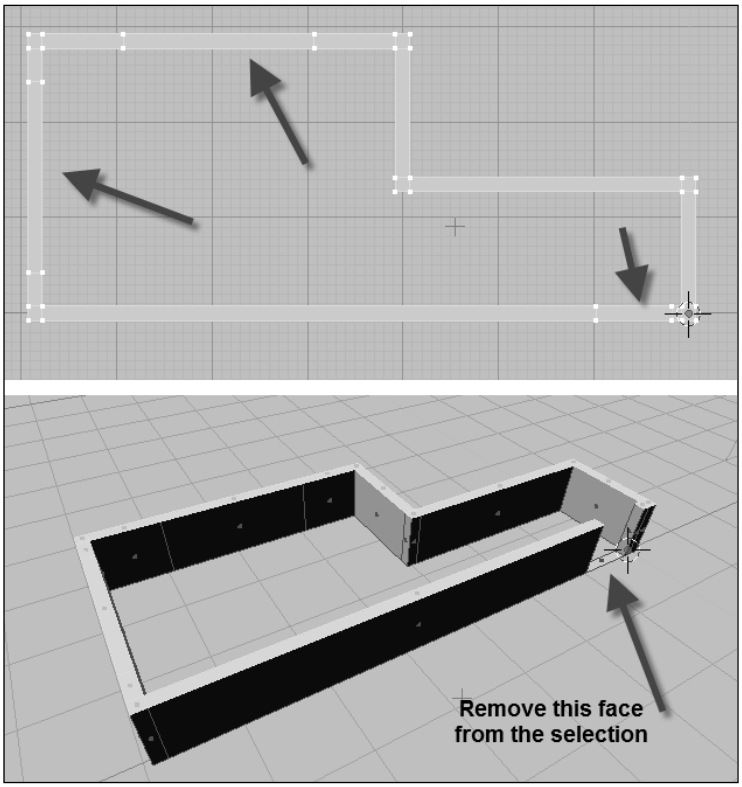
Openings

It's hard to imagine any project without openings, so get used to creating a lot of openings in walls. The best way to deal with openings is to know exactly where they are placed before we start to create the windows. If we do that, we won't have to create new loops in the walls and use a lot of Boolean operations to create holes. The trick is to create the walls already with the space for windows and doors.

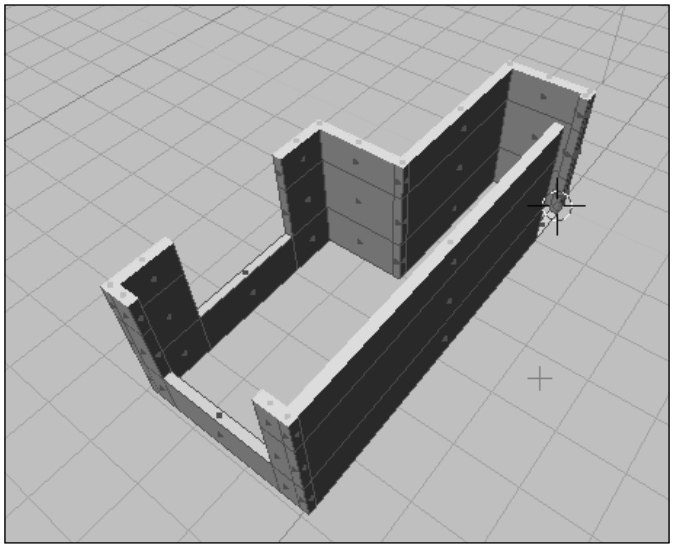
If we take the following plan, we can see that we have openings already. All those openings should be marked before we start to model:



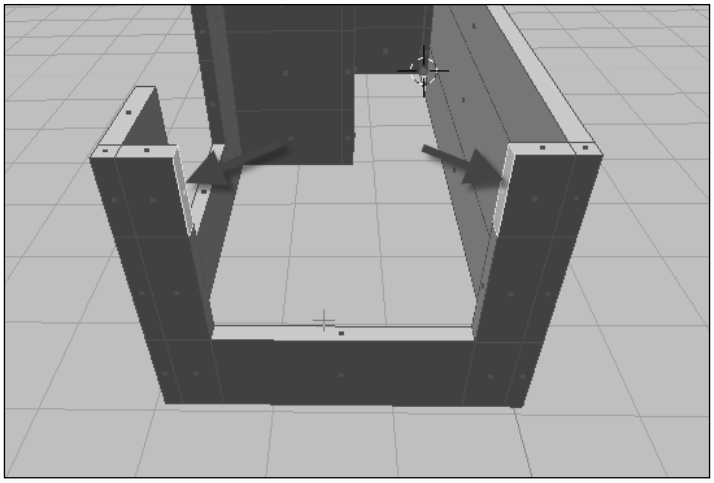
The technique is to start modeling the walls, and for every opening, just leave a plane. This will mark the place for the posterior extrude. When all the planes are created, select the planes and remove any plane that represents a door. Make the first extrude, until the wall reaches the first windows:



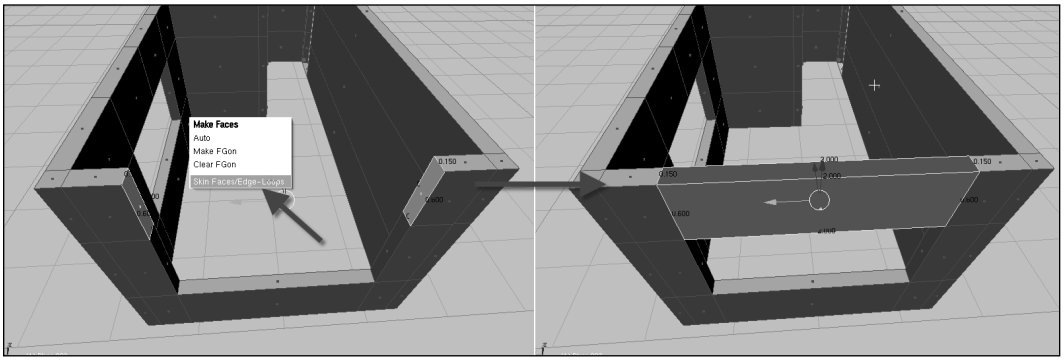
Then, remove the faces that mark the windows. After removing the faces, make another extrude until we reach the upper limit of windows and doors. And to finalize it, make the last extrude until the top of the wall is reached:



Well, now we have our walls with some big holes. To close these holes, we have two options. The first one involves keyboard shortcuts, and the other one is very simple, and makes use of a tool named **Skin Edges/Face Loops**. Let's start with the easiest. To use the script, just change the selection mode to Faces and select two faces that must be connected:

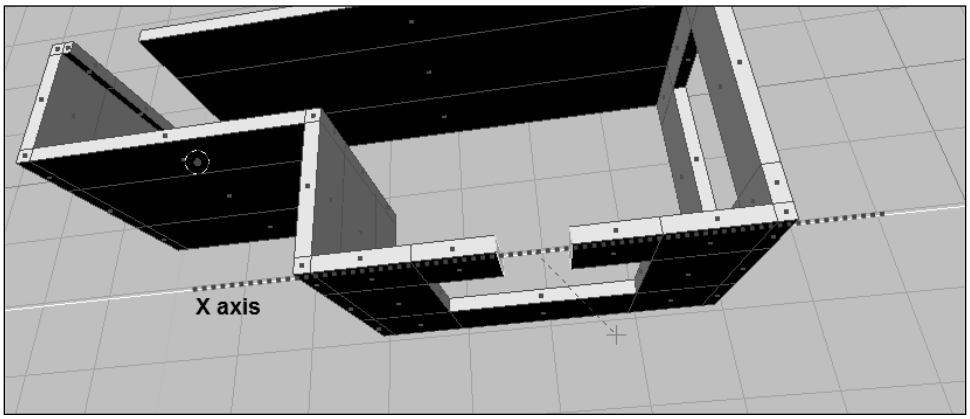


When the faces are selected, press the *F* key and choose **Skin Edges/Face Loops**. This will create new faces to connect the previously selected faces. For this tool, the faces must be exactly parallel:

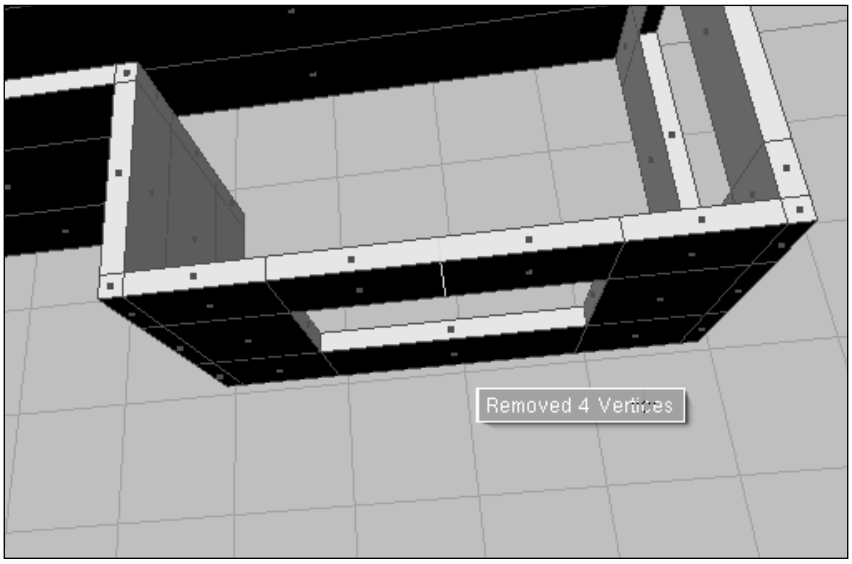


The other method works like this—we have to extrude both faces and then scale them, until they touch each other. It works like this:

- Select the faces that must be connected.
- When the faces are connected, press the *E* key to extrude them, and right after pressing the *E* key, just press *Esc* to cancel the extrusion. It's really important here not to move or click with the mouse.
- Then press the *S* key to scale them down. Right after pressing the *S* key, press the key corresponding to the axis which is perpendicular to the planes. In my case, it's the X axis, so I will press the *X* key:



- To finish, press the *0* key in the alphanumeric keyboard to make the scale turn to zero. It will connect the faces:



- Press the *W* key, and choose **Remove Doubles** to erase any duplicated vertices.

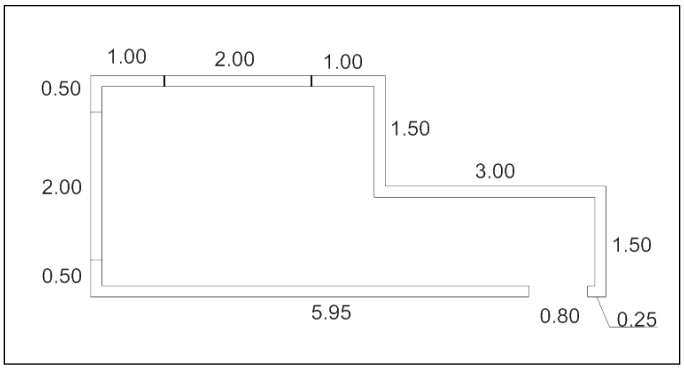
As we can see, the first method is a lot easier.

Floors and Lining

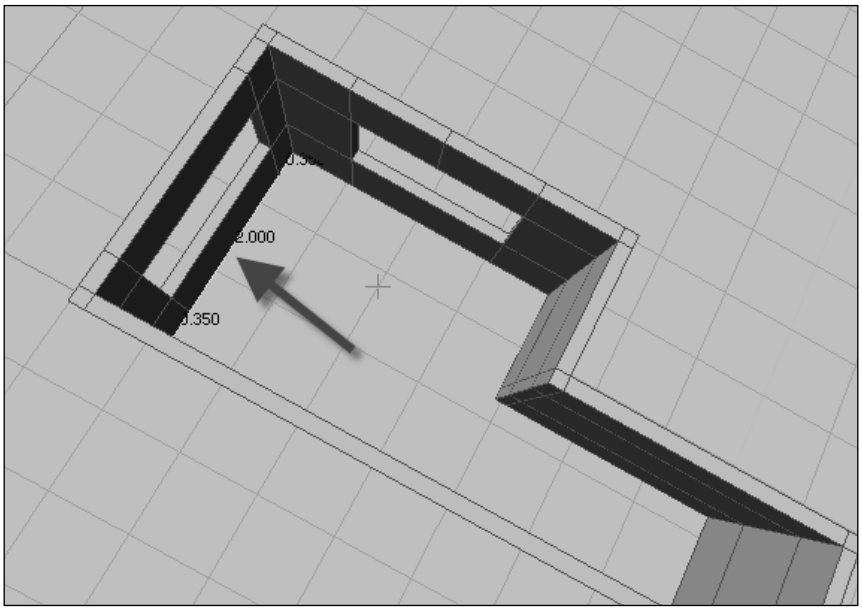
To create floors and ceilings, we can work with new geometry or use the plane to start the model as a single piece. It works like this: we create a base plane, and with this plane, the edges will be extruded to make the basis for walls. What's the best way? Again, there isn't a best way, you have to try both methods and choose the best one for each project.

Modeling using the walls

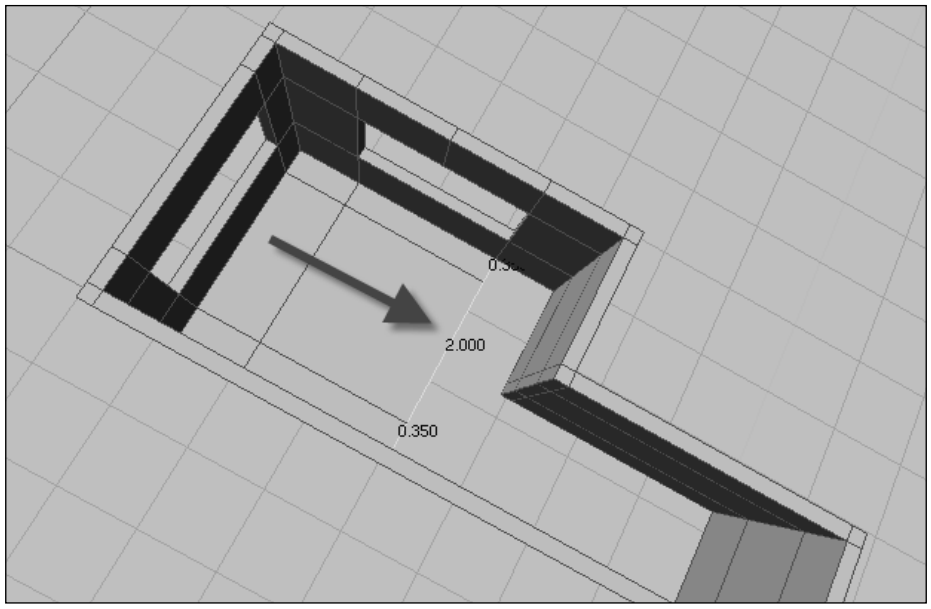
A good way to start is trying to make the floor and lining from a single piece. Let's take one of the projects on which we have worked, such as this next one:



Select one entire edge, from the side pointed to in the following image, and press the *E* key to extrude it. Hold the *Ctrl* key to use the grid lines and create a more regular plane:



Keep doing extrusions until the entire floor plane is filled. Make sure that that we fill the floor using square faces. For that, a small sketch of how these faces will be distributed may help. If you are not comfortable picturing this, just grab a piece of paper and make a sketch:



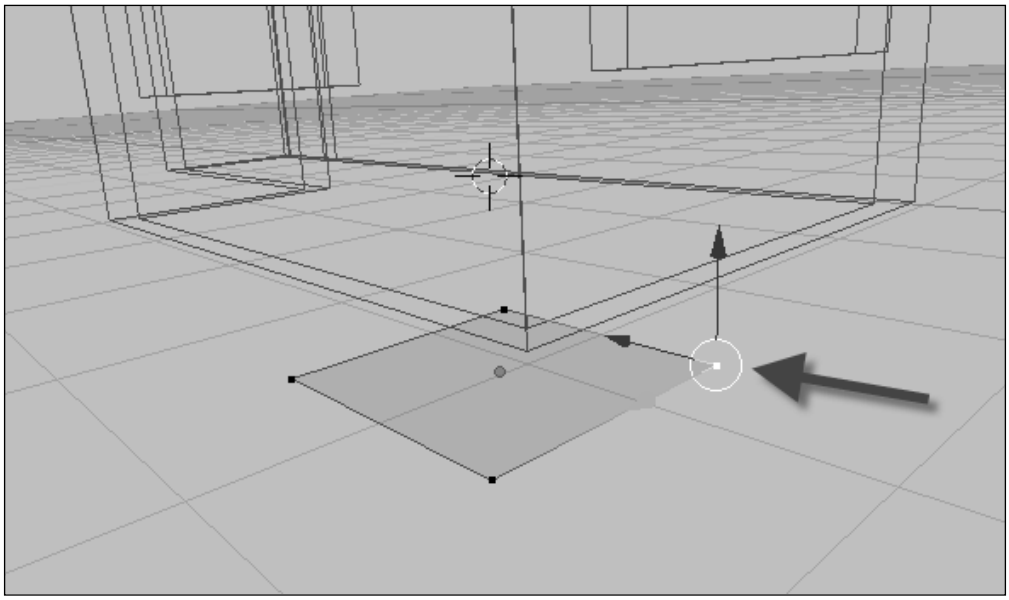
We can do the same thing for the lining. Just select one edge and extrude it until the entire plane is filled. At the end, remember to select all the objects and remove the duplicated vertices.

Use this method for simple projects or for interactive animation. It will be easier to manage and manipulate the planes. If you are going to use complex textures, it may be a little hard to set up UV Mapping.

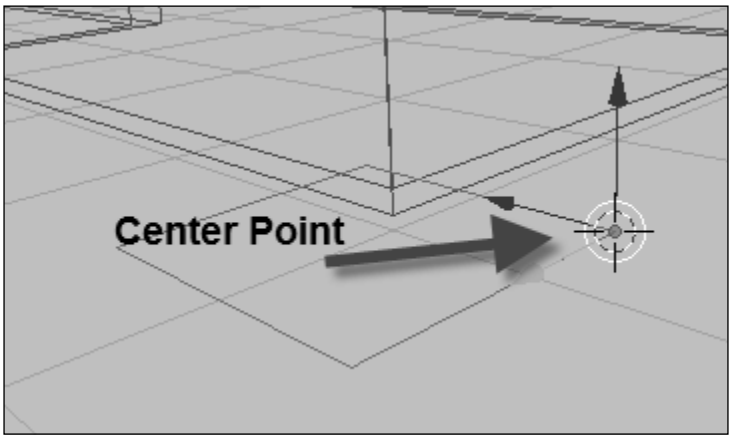
Modeling with separated objects

If you want to make a separated object, just create the plane to start the modeling. With this method, we will have to use a small trick to align the plane with the wall, using the Snap tool. The trick is to place the object center at a corner of the plane, and put the 3D Cursor at the other corner, and to use the **Snap Selection | Cursor**.

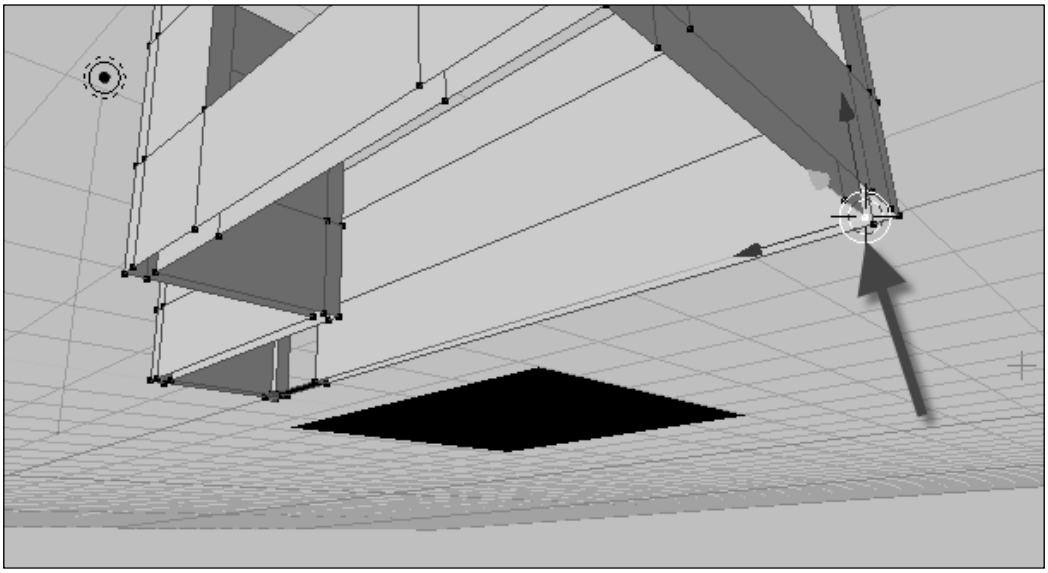
Let's see how it works. First create the plane or cube. Then, select a vertex in a corner that will be placed near the wall:



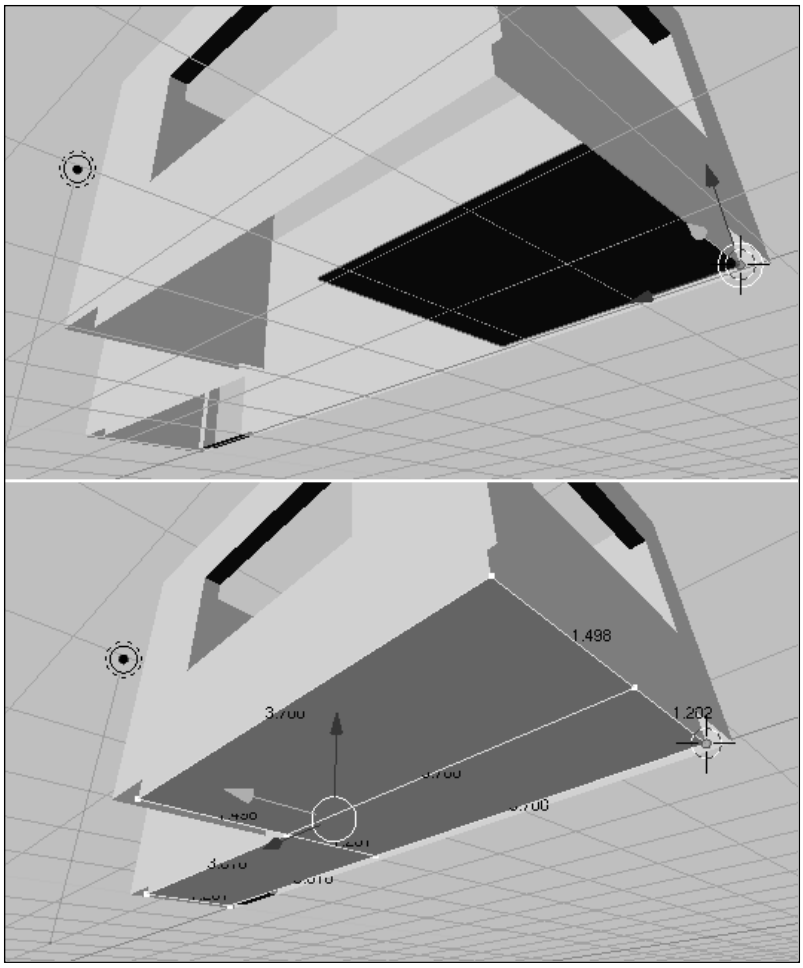
Press *Shift* + *S* to call the **Snap** menu, and choose **Cursor | Selection** to make the 3D Cursor go to the same position as the selected vertex. When the 3D Cursor is placed, go to Object mode, and in the Editing panel, press the **Center Cursor** button. It will place the object's center at the 3D Cursor's position:



Now, we have to select a wall vertex, which will work as reference for us. Right after selecting this vertex, press *Shift* + *S*, and choose **Cursor | Selection** from the **Snap** menu. It will make the 3D Cursor jump to the vertex position:



To finish this operation, select the plane that will be our floor or lining in Object mode. Press *Shift* + *S*, and choose **Selection | Cursor**, and you will see that our plane will be placed in the right position, perfectly aligned with the wall. This may be a difficult technique, with a lot of keyboard shortcuts, but it's the best way to align objects in Blender with the default tools.



This method works best for complex models, where we may want to hide the walls and work only on the floor or lining. It makes the process of texturing easier too, allowing us to work only with the floor or lining planes.

Starting from a CAD drawing

We have to learn how to deal with CAD files, because a lot of projects that come from architects are in formats like DXF and DWG. Let me tell you that Blender cannot read DWG files, so if you get any files like that, ask your clients to provide the same file in DXF, or you will have to convert it yourself. Almost any CAD software can save files in the DXF file format. That makes Blender compatible with most of these software packages.

Preparing the DXF files

Before we import a DXF file, there are a few tasks that we can do to make the DXF file cleaner and smaller. When we make a technical drawing with a CAD software, it's full of symbols and other objects that won't be very useful for 3D modeling, such as text, architectural symbols, and hatches.

If we remove some of these objects, the file size and complexity of the DXF will decrease and make everything easier. Following is a list of things we can do to prepare the DXF file in your CAD software:

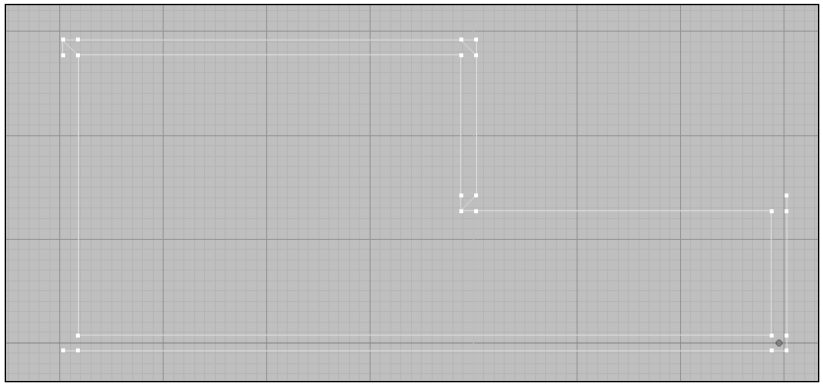
- Erase everything that you won't need. Just leave the lines of the drawings.
- Erase layers and other CAD-related stuff. Remove any line types, hatches, and other CAD-related stuff.
- Try to save your DXF in the format of AutoCAD 12. It's more compatible with Blender.
- If your project is too complex, try to split the drawings into more than one file.

These are simple tasks, but can really improve the compatibility of your technical drawings and Blender.

Importing DXF files

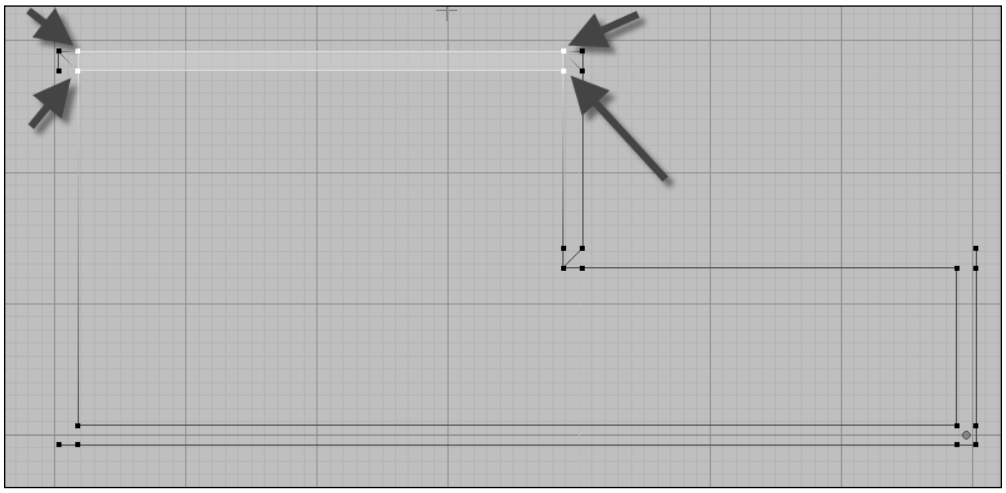
Working with CAD files is sometimes better if the project has complex shapes, because we won't have to measure edges to get the right dimensions. The CAD file will already have the right dimensions, and we can focus on the geometry only. To get the file into Blender, just access the **File** menu and then **Import | DXF**.

After we have imported the CAD file into Blender, we will have to do some editing work, because this kind of file comes with the edges separated, as we can see in the following image:



If your CAD file doesn't come with only with the lines that define walls and openings, it could be necessary to do some heavy cleaning and erase the extra data, such as text and symbols.

What we have to do here is to select the vertices that should be together, and make a merge. Just select them, and press the *W* key, choose **Merge** and a method. Usually the best method is **At Center**. When all vertices are merged, we will have to build the faces. For this, we will have to select four vertices of a face, and press the *F* key. This way we will create a new face, which will be required for a posterior extrude:



A bad thing about DXF files is that they don't support curves. If our drawing has some splines, we will have to turn them into polygons. To make things easier, we can use the Spin tool.

As we can see, using CAD files requires a lot of editing work. Is there any advantage to using these files? Yes, the only advantage is that we won't have to worry about dimensions and proportions, because the file will already have the right measurements.

Summary

We have learned a lot now! Some of the most important elements for architectural visualization were explained, and we can model them in Blender. Here are the main topics that we have talked about:

- Planning the modeling
- How to model with precision
- Organize our scenes in layers
- Create Walls
- Create Openings
- Create Floors and Linings
- How to start a model from a CAD file

In the next chapter, we will see more about modeling, but get deep into details, to make our models more complex.

5
Modeling Details

The next step for us is to learn how to add more details into our models, such as windows, doors, and stairs. In the previous chapter, we saw how to model walls and import DXF files into Blender. Let's use the objects that we have modeled previously and give them a more realistic look with details.

A great level of realism is achieved in architectural visualization by adding details to models, such as window frames, and more. If you want to give the images a good level of realism, a big part of it is related to details. Let's see how we can model these parts and make our scenes more real.

Level of detail

Before we start, it's important to talk about something named level of detail. This is a common term in game development, but we can use it in architectural modeling as well. The concept of "level of detail" deals with how much detail a model must have to be visible by the camera, because it will be pointless to produce a detailed model placed far from the camera.

That's why before you start to model anything, make sure where every camera will be placed in your scene. Otherwise, there is great chance that a big part of the time spent in modeling will be wasted.

Windows

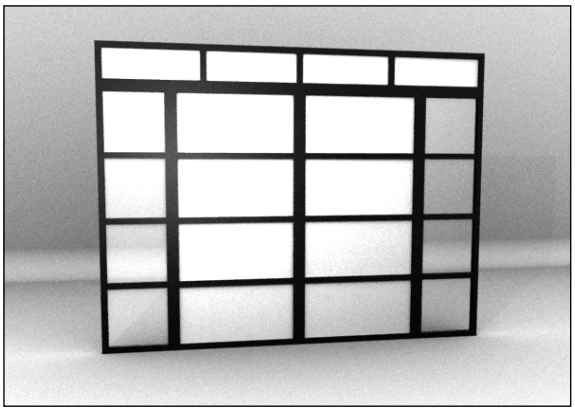
In the previous chapter, we have modeled the walls and prepared the geometry with the hole for the window. But besides the opening, we must prepare a window frame to receive the glass and other common elements for windows. There are a lot of window types, such as:

- Double-hung sash window
- Single-hung sash window
- Horizontal sliding window
- Skylight
- Roof window
- Fixed window

One of the most common window types, at least in the countries where English is the primary language, is the Double-hung sash window. Besides the type, we have to know the material of the window and the dimensions to start a model.

The dimensions of the window will be important to measure the amount of sunlight that will enter the scene. This will be important when we will deal with lighting. To make a window model, we will use the same technique used to make the walls. It all starts with a primitive mesh such as a cube, and then we extrude this cube to build the geometry needed for the window frame.

Let's use this Double-hung sash window as an example:

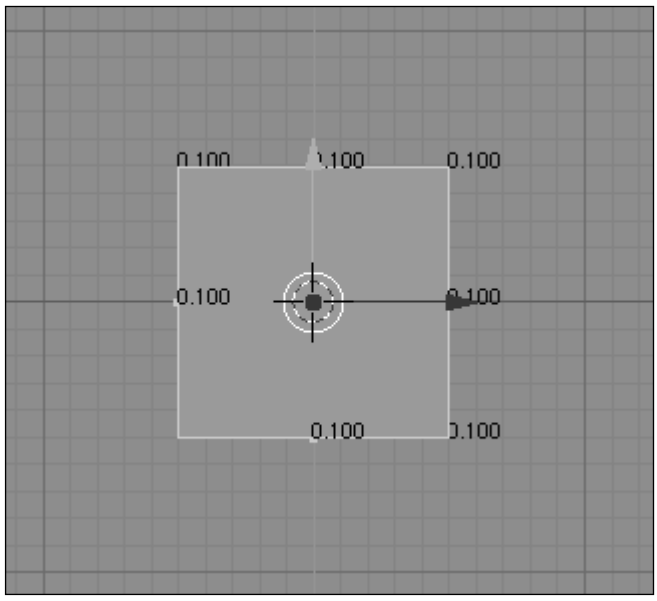


Remember, what we have to make here is a visual representation of this window. We don't have to follow all the visual aspects for the frame. But to make things right, we have to know the measurements for all the parts. If we don't follow these measurements, our window model won't look real.

Another thing to remember is that we will use subdivision modeling, which means that the first step is to choose a primitive shape, in this case a cube, and deform it until the shape of a window is achieved:

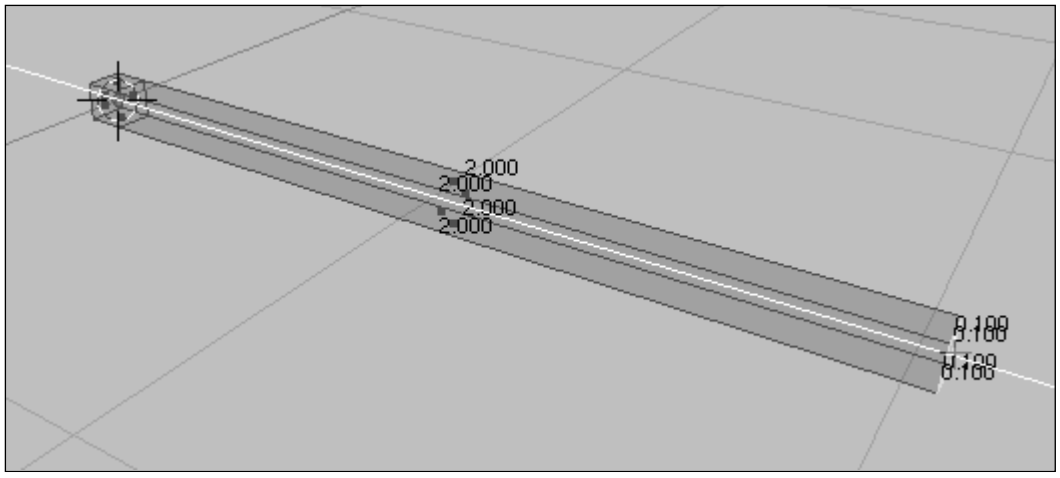
1. If you don't have a cube in your scene, press the *Space bar*, and create one with **Add | Mesh | Cube**.
2. If you already have a cube, change the work mode to Edit, otherwise, after adding a new cube, the work mode will automatically change to Edit. Now press the *A* key to select all objects. Then change the selection mode to Faces, to make the selection process easier.
3. Turn on the **Edge Length** button. It will make the editing process easier if we want to keep track of the measurements.

With all objects selected, press the *S* key, and scale down the cube until it reaches about **0.10** for all edges. Use the *Ctrl* or *Shift* keys while you move the mouse cursor, to have more precision for the scale:

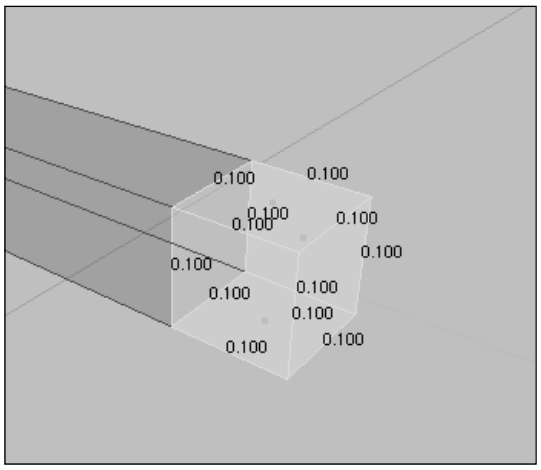


1. Because our window will have all it's parts based on this small cube, let's create a copy of it before we start to subdivide. Change the work mode to Object and press *Shift* + *D* two times to create two copies of the cube. Just place the copies somewhere near the original cube. We will use those copies later.
2. Now, we need to select one face of the cube. To select, just right-click on it.

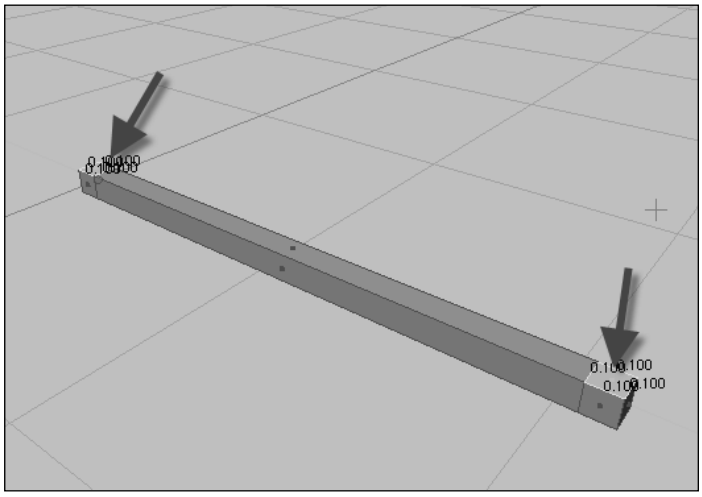
3. When the face is selected, press the *E* key to extrude this face, and move your mouse until the new face reaches **2.00**:



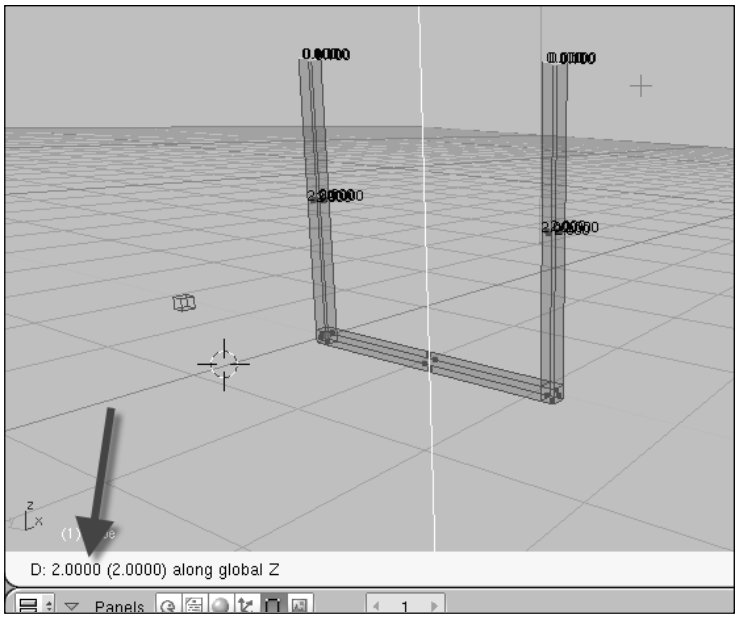
4. With the face still selected, press *E* again, and make another extrude until it reaches **0.10**:



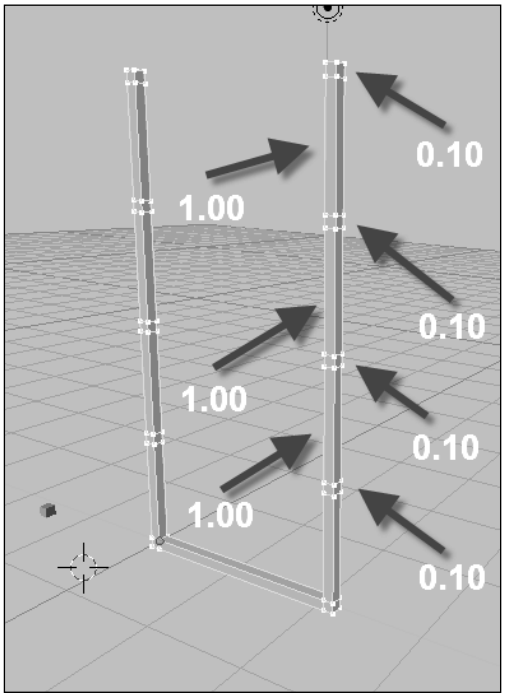
5. Remove any selected face, and then select the small top faces on the left and right sides:



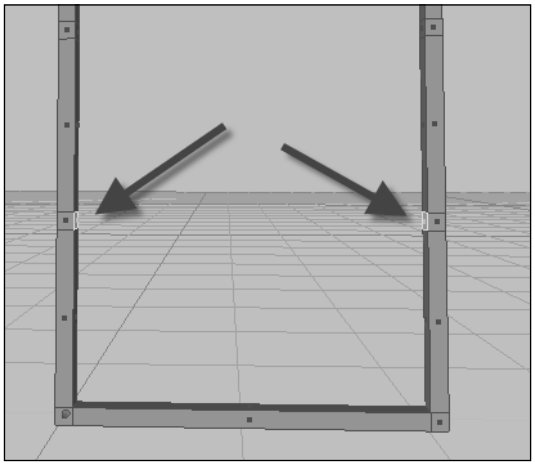
6. Extrude these faces until they reach about **1.00** units:



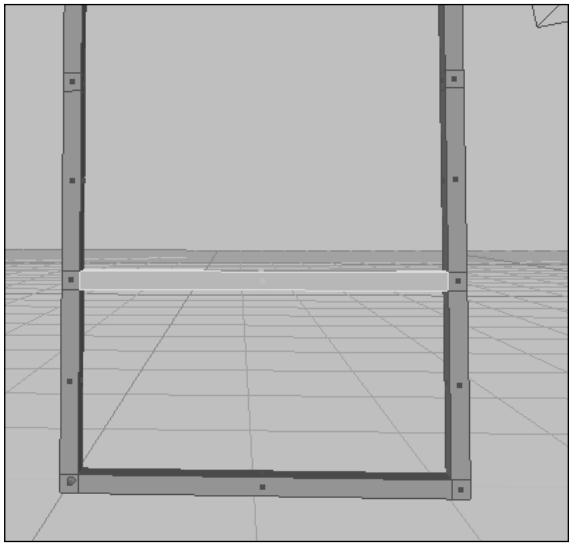
7. Repeat this extrusion seven times, and give each part these distances: **0.10**, **1.00**, **0.10**, **1.00**, **0.10**, **1.00**, and **0.10**:



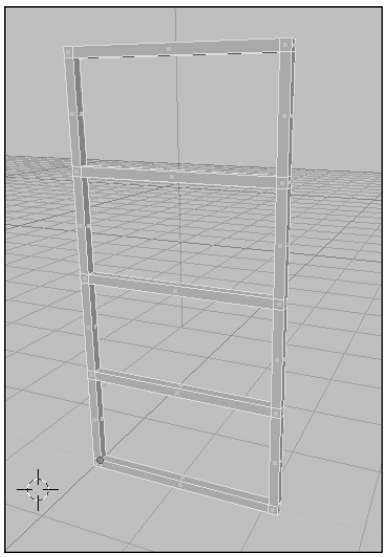
8. Now, we will make a connection between the middle faces, with the **Skin Faces/Edge-Loops** tool. Select the two faces pointed to in the following image and press the *F* key. A small menu will appear when the *F* key is pressed, and then choose the **Skin Faces/Edge-Loops** tool:



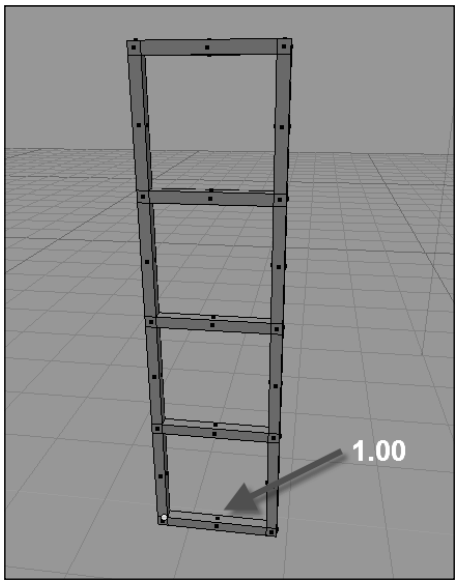
9. The following image shows the connected faces:



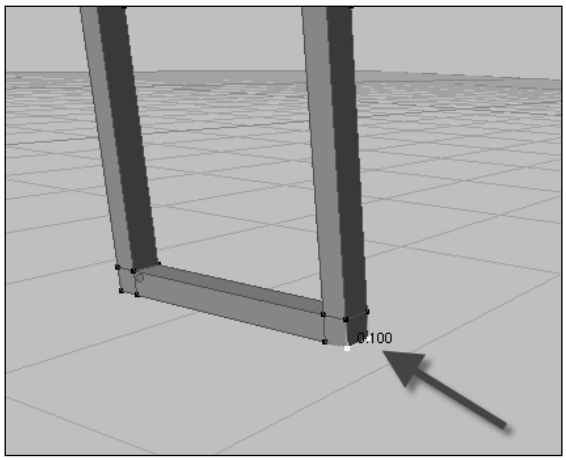
10. Repeat the same process with all the other small faces until we get a model such as the one shown in the next image:



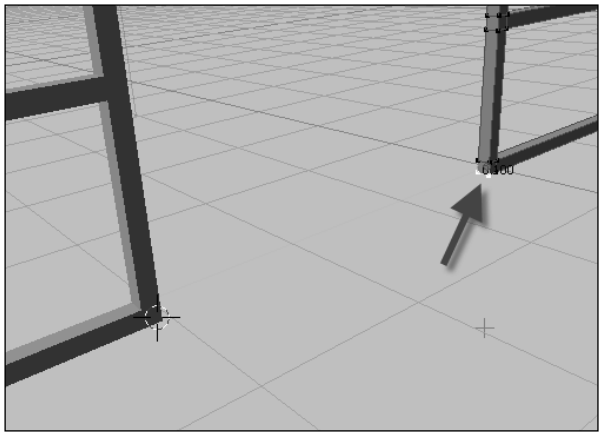
11. Remember the copied cube, which we created at the beginning? Let's apply the same modifications such as we just did. Repeat all the previous steps, until we get a model just like the one shown in the following image. Just use a smaller distance between the two main vertical parts for this model. A distance of **1.00** works fine:



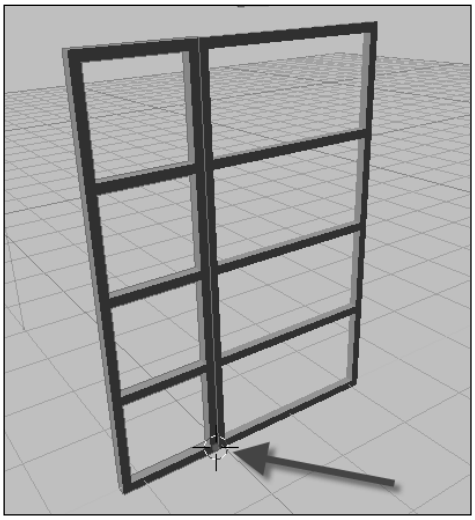
12. Now we will have to align both the models. Select the edge pointed to in the image, and then press *Shift* + *S* to call the **Snap** menu. Choose **Cursor | Selection** to make the 3D cursor jump to the middle point of the edge:



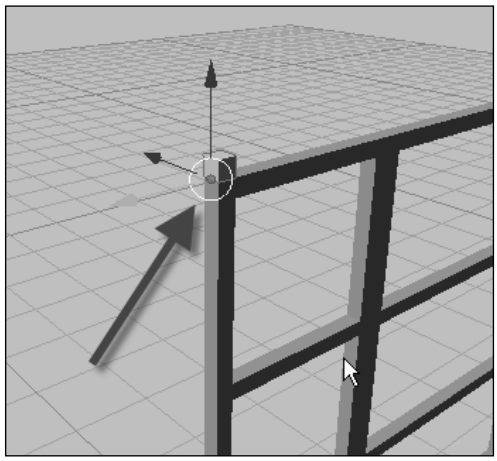
13. When the cursor is in the right position, change the work mode to Object, and in the **Editing** Panel, press the **Center Cursor** button. This will make the object center jump to the place where the 3D cursor is.
14. With the center of the first object placed in the right position, we have to do the same thing for the second one. Select the edge pointed to in the following image:



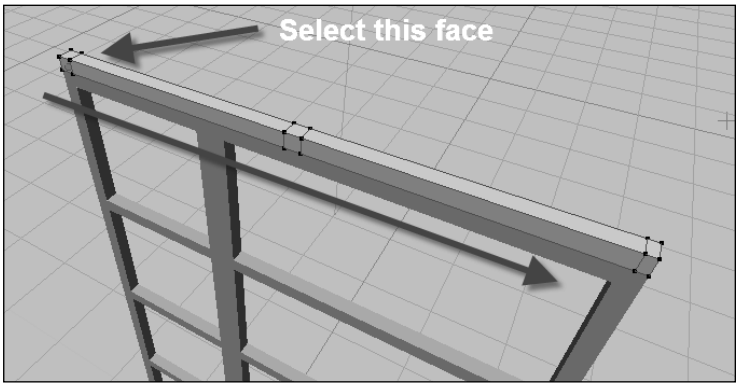
15. When the edge is selected, press *Shift* + *S* and choose **Cursor | Selection** again.
16. Finally, we have the objects with the right conditions for alignment. Change the work mode to Object, and select the object which had the center point edited first. When this object is selected, press *Shift* + *S*, and choose **Selection | Cursor**:



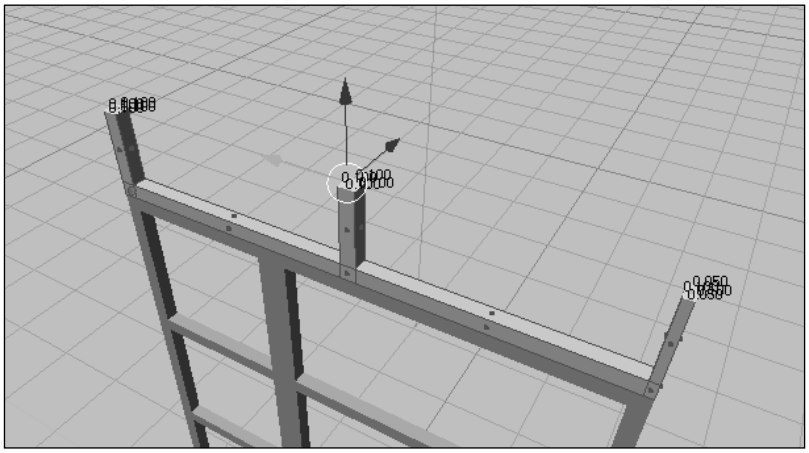
17. This will make the object jump to the right position. Using the center point and the 3D cursor is the most efficient method to align objects with precision in Blender. You may be asking yourself—why not make both objects as one? Creating two objects will make our life easier if we want to add some animation to the scene. But it's completely normal to build the models as just one piece.
18. We have the model now, so what's the next step? Before we move forward to make the other parts, let's build the upper part of the window frame. Select the other cube which we have copied in step number five. Move it until it reaches the top of the window at the right side, as shown in the following image:



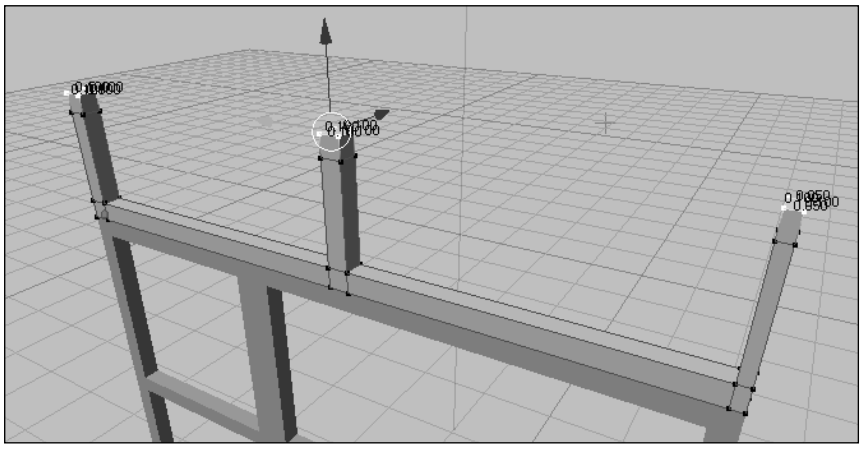
19. Enter into Edit mode, select the right face for this cube, and extrude it four times with the respective measurements: **1.60**, **0.10**, **1.55**, and **0.05**:



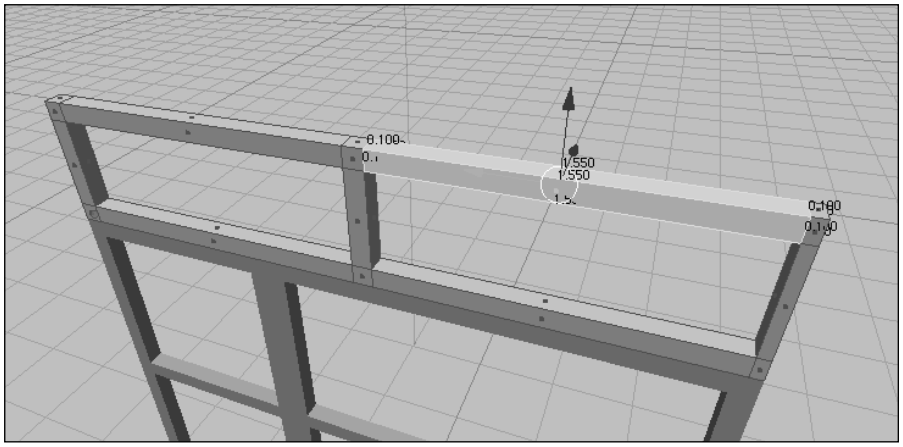
20. Now we have to extrude the three small faces pointed to in the following image. Extrude them until they reach **1.00** units high:



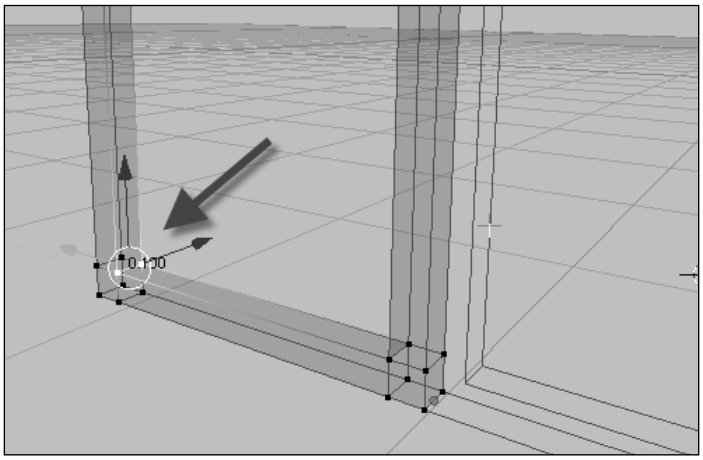
21. With the faces still selected, make another extrude of **0.10** units:



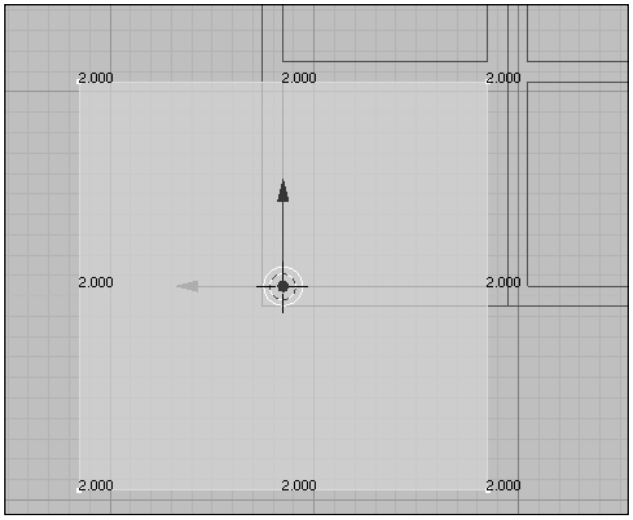
22. To connect the interior faces, just select two of them and press the *F* key to use the **Skin Faces/Edge-Loops** tool. Repeat the process two times, until the models looks like the next image:



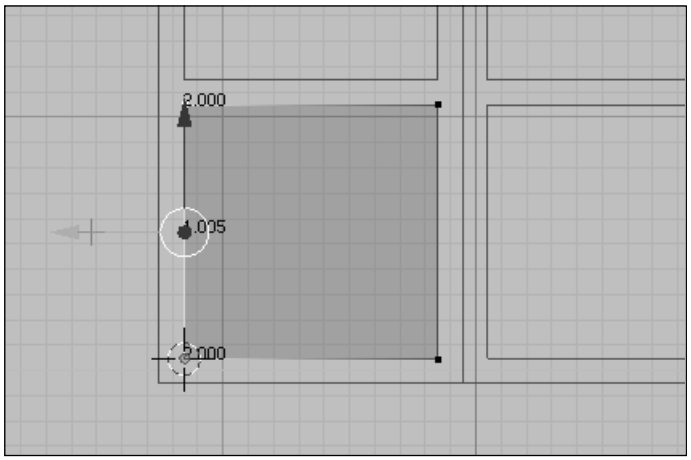
23. With most of the frame created, we can create the objects that will represent the glass. Select the edge at the bottom of the frame, pointed to in the image below. When the edge is selected, use the **Snap** menu to align the 3D cursor with this edge, and choose **Cursor | Selection**:



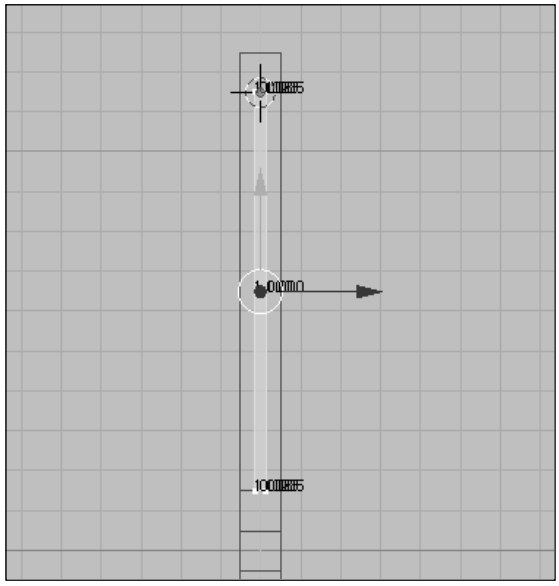
24. Change the view to Front. Depending on your model, it could be the *1* or *3* keys on the numeric keyboard. Make sure you are in Object mode, and create a cube:



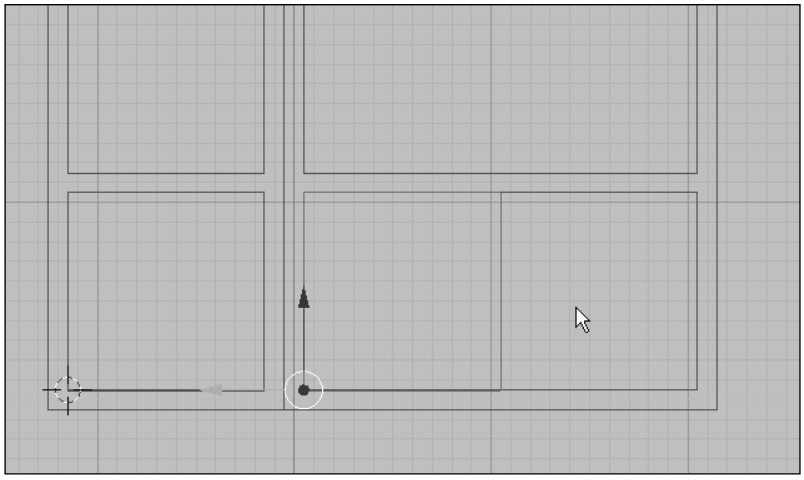
25. With the Box selection and the Grab transformation, move the vertices to adjust the size of this cube, until it fits at the right place in the frame:



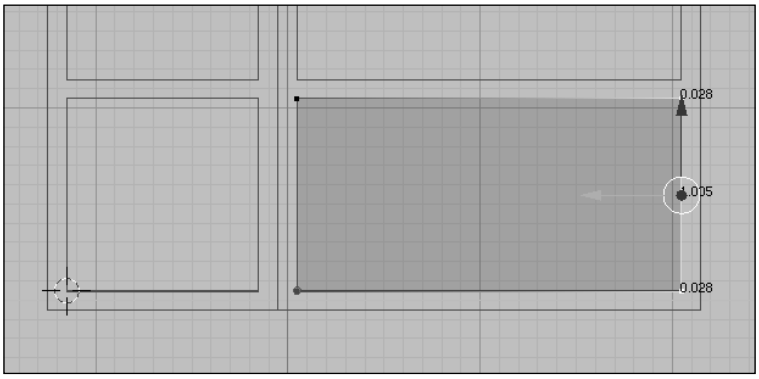
26. Change your view to Top, and with all vertices selected, press the *S* key to scale down the model. After pressing the *S* key, press the key corresponding to the X axis to make the scale only on this axis. Scale it until it reaches **0.03**:



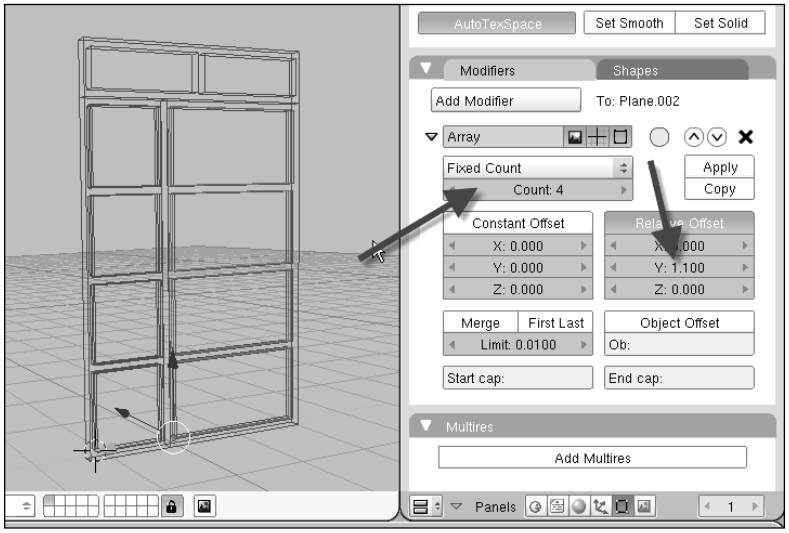
27. With the cube still selected, press *Shift* + *D* to create a copy of it, and move the copy to the next hole:



28. In Edit mode, select the vertices for this new cube and adjust its size to fit in the frame:

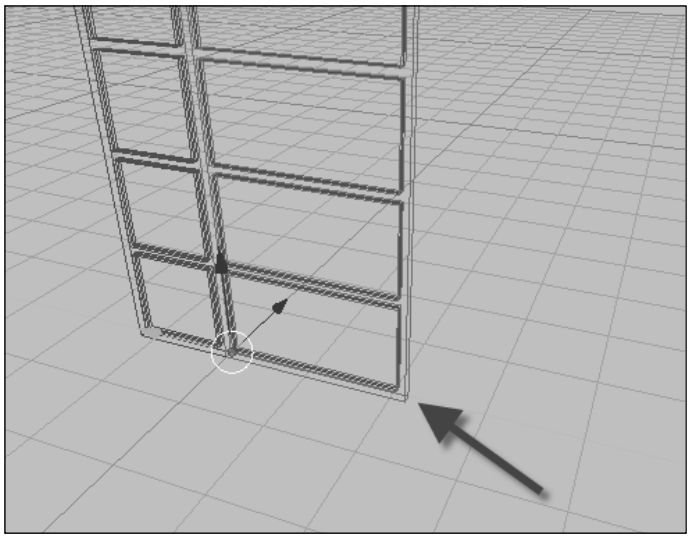


29. Now let's distribute the cubes in the other holes in the frame. To do this, we will use the **Array** modifier, and will use it to make four copies, which will use a relative offset of **1.10** units. Apply the modifier for each cube:

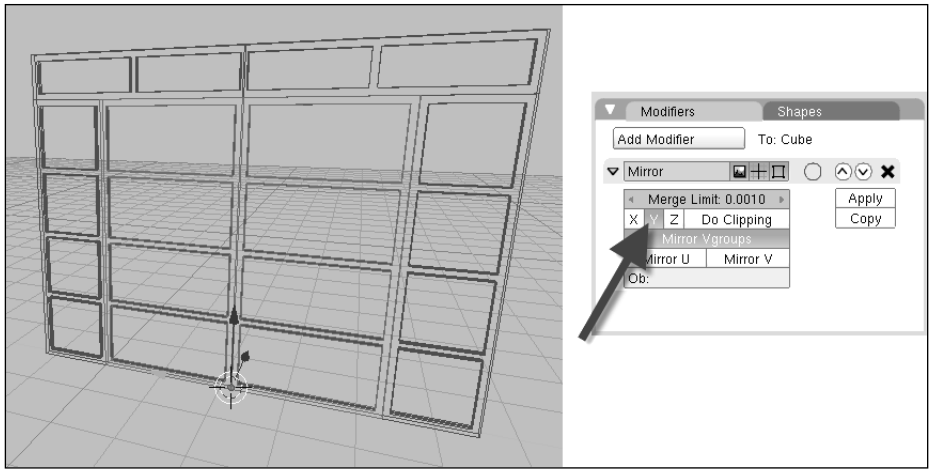


30. Use the same technique to make the cubes for the upper part. If we want, we can duplicate the cubes used previously and remove the **Array** modifier. Adjust their size to fit the frame.

31. To apply a **Mirror** modifier and finish the model, we must set the objects center to be placed at the position pointed to in the following image. Remember that you won't have to setup the position of the 3D cursor for every object. Just place it one time and then select one object, and press the **Center Cursor** button in the **Editing** Panel:



32. When the center is set up, just apply a **Mirror** modifier to all objects, and the window will be ready:



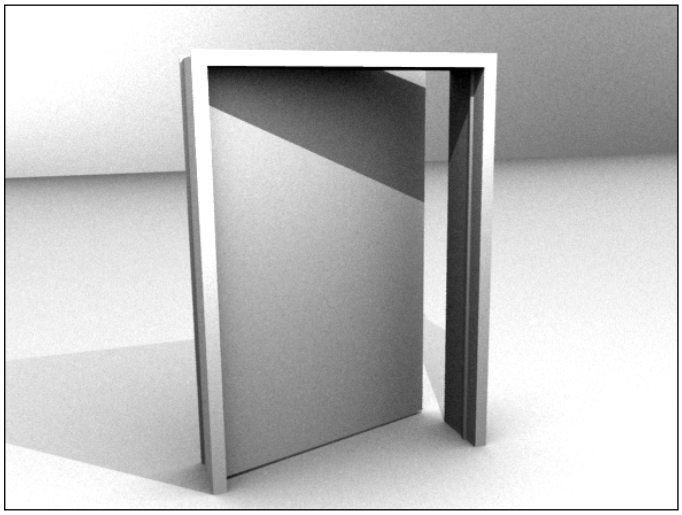
Level of detail

The camera view will determine how detailed the window must be. If the camera is near the window, we have to add more details. In most cases it won't be necessary, but it's good to know the camera view before starting to model anything. It can save a lot of time, especially if the camera is placed far away from the model.

Doors

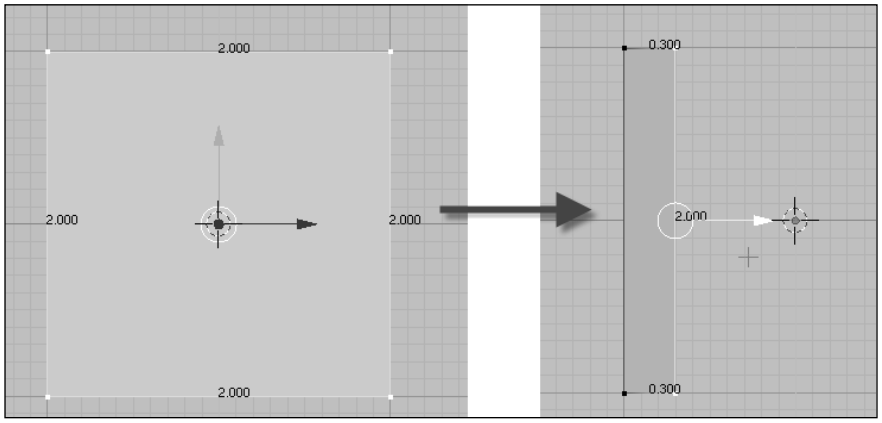
Modeling doors are a lot easier than windows, because most doors don't have the amount of details that windows have. But, depending on the type of the door, we can have pieces of glass as ornamental details. It will depend on your project.

Like we did with windows, let's create the door shown in the following image to understand how we can create a door model with a frame:

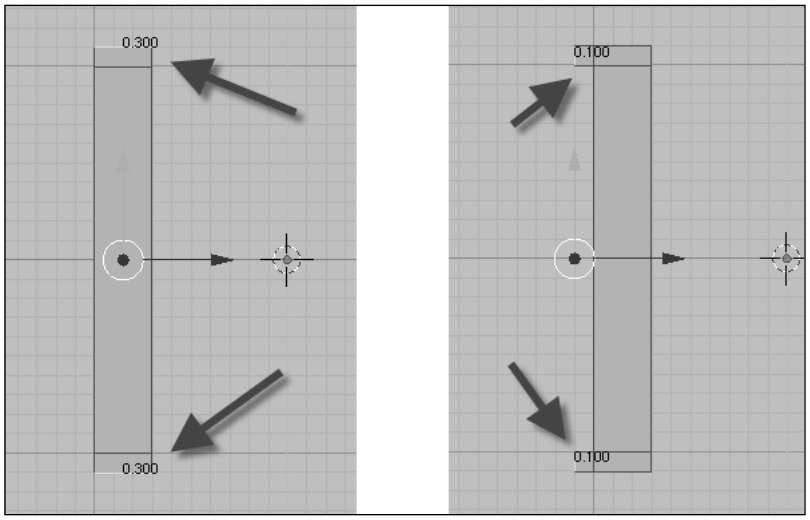


As you can see, the model is simple, but depending on the camera view, we may need to add more detail to the model. Let's get started. The first thing to do is to create a plane. Don't worry about the measurement now; by the end of the modeling process we will apply a scale to the model:

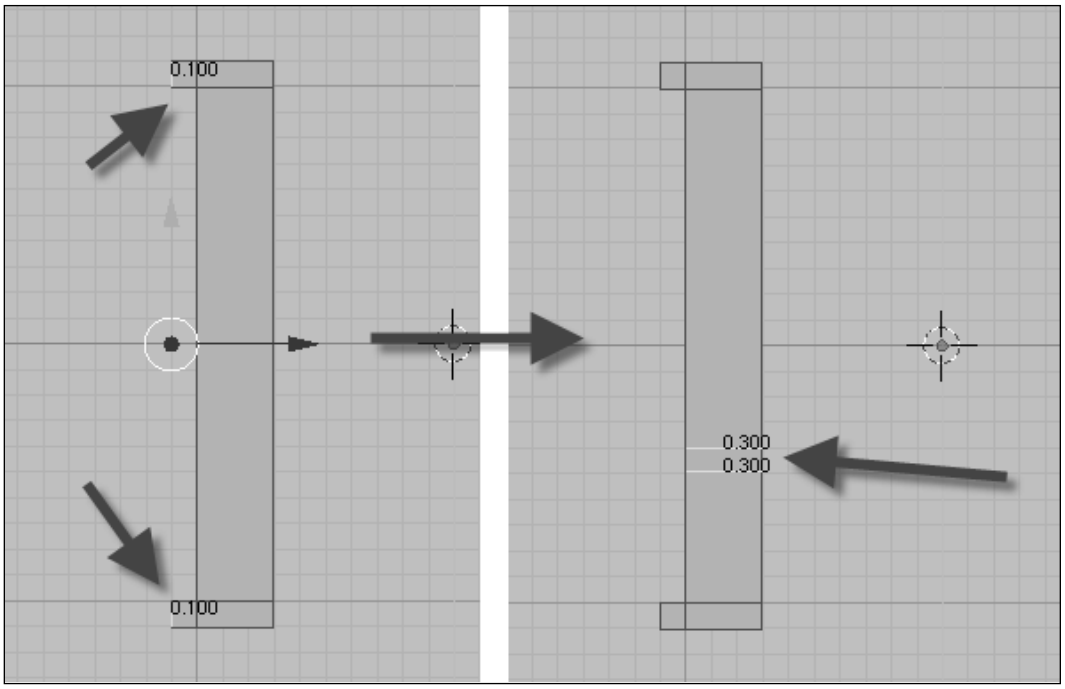
1. While you still are in Edit mode, select the right edge of the plane and drag it to the left. Drag it until we get a distance of **0.3** for the upper and lower edges:



2. Change the selection mode to **Edge**; it will make your work easier. Select the upper edge, and then extrude it until it reaches **0.10** units. Repeat the same editing for the lower edge. Now we have to select the small edges on the left, and extrude those edges until they reach **0.10** units. Remember to hold the *Ctrl* key to help place these extruded edges at their proper measurements:

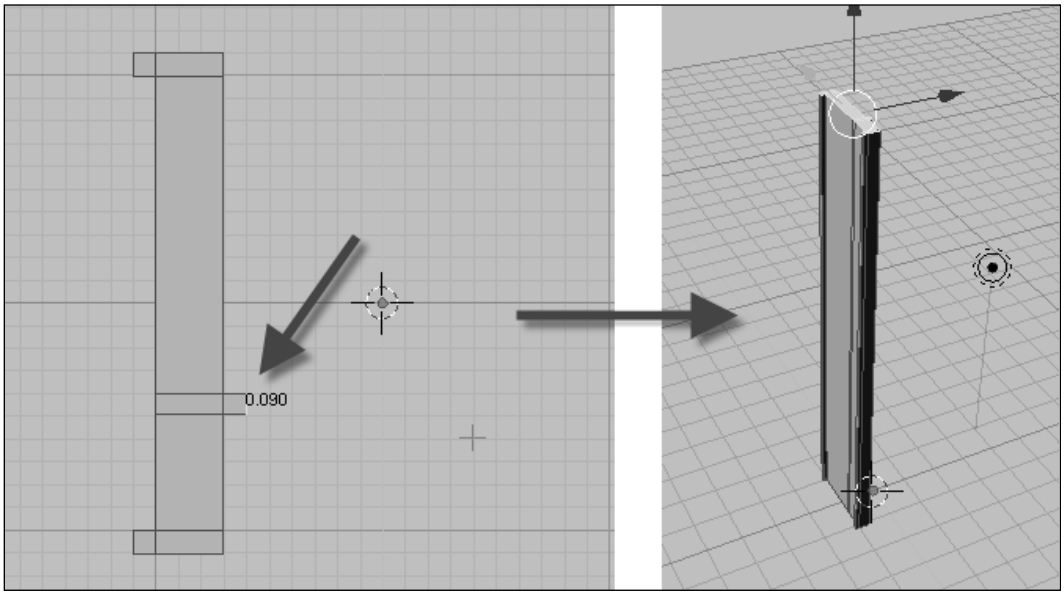


3. With the face loop cut, let's add two new edges to the plane. Just press *Ctrl* + *R* to activate the tool, and drag your mouse cursor to the point marked in the next image. Click two times to mark the cutting place. Repeat the same process to add another edge:

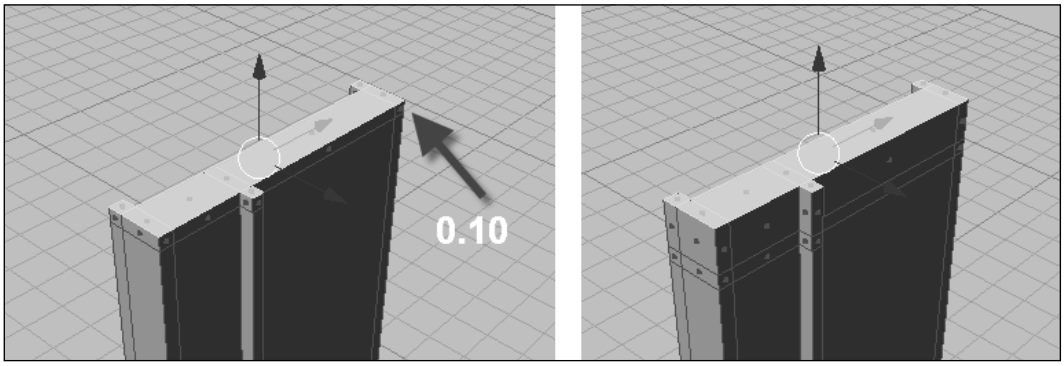


4. Select the small edge between the two new edges added with the face loop cut. When the edge is selected, extrude it **0.10** units. Now, we have the base plane to create the door frame. The next step is to select all the edges and extrude them.

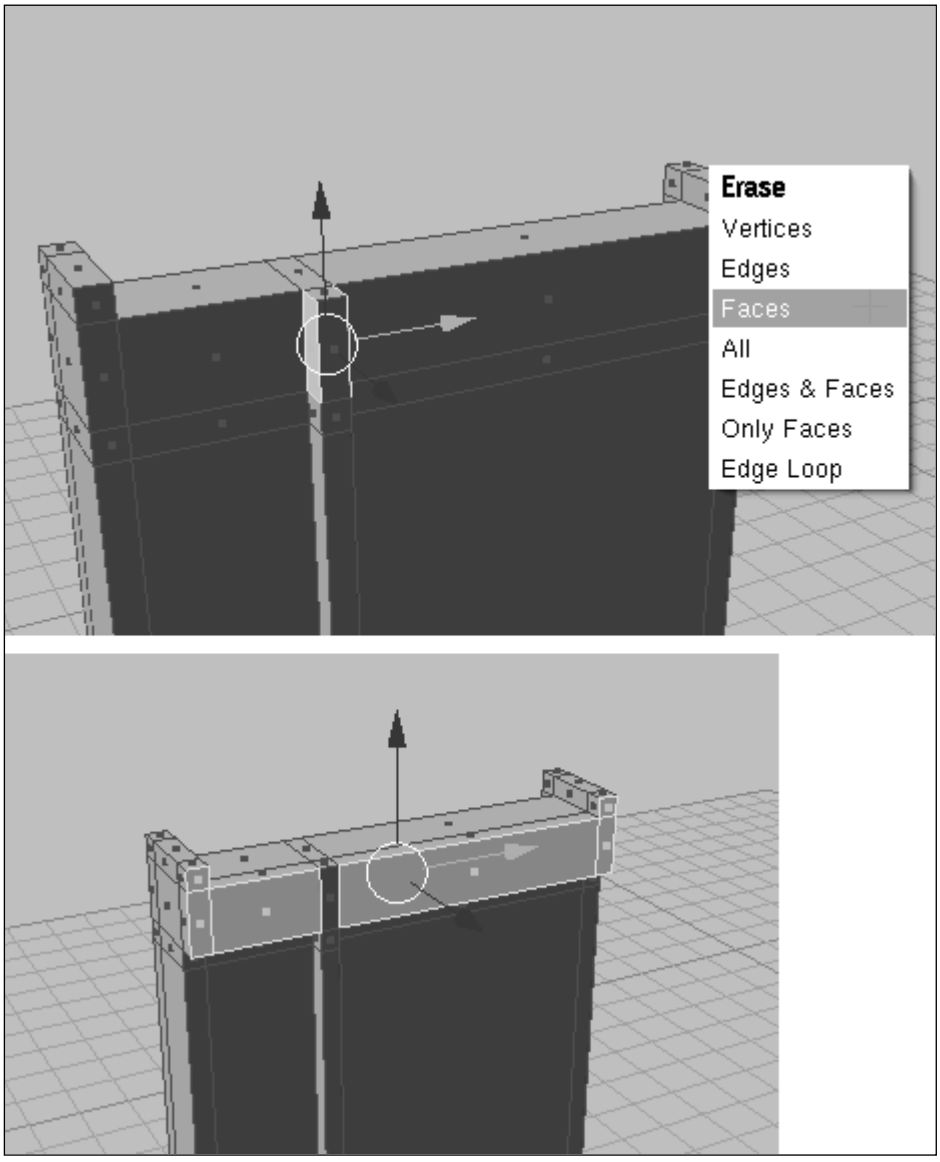
5. Before you extrude it, orbit the model to have more visual control over extrusion. When all edges are selected, extrude them until we get a measure of **2.00**:



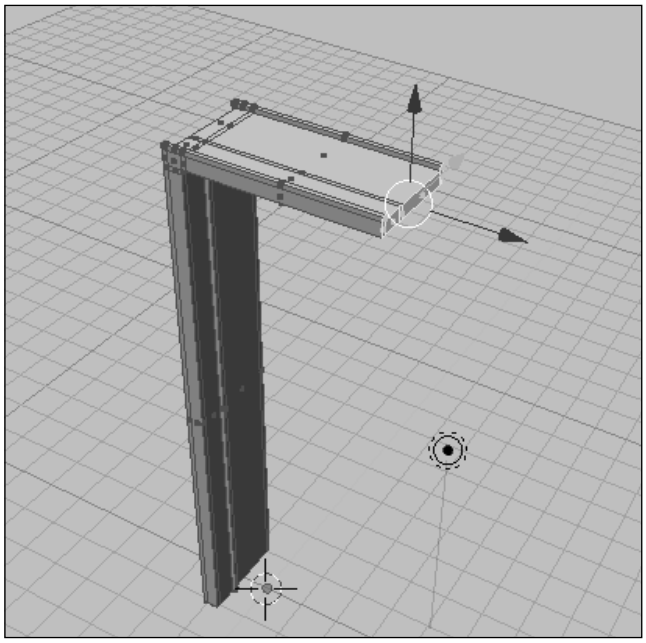
6. Change the selection mode to **Face**, and select all faces at the top of the model. When all faces are selected, extrude them until they reach **0.10** units. With the faces still selected, extrude them one more time, but now with **0.30** units. Now, we have to select only the small faces at each side of the top plane. With these faces selected, extrude them until we get a distance of **0.10** units:



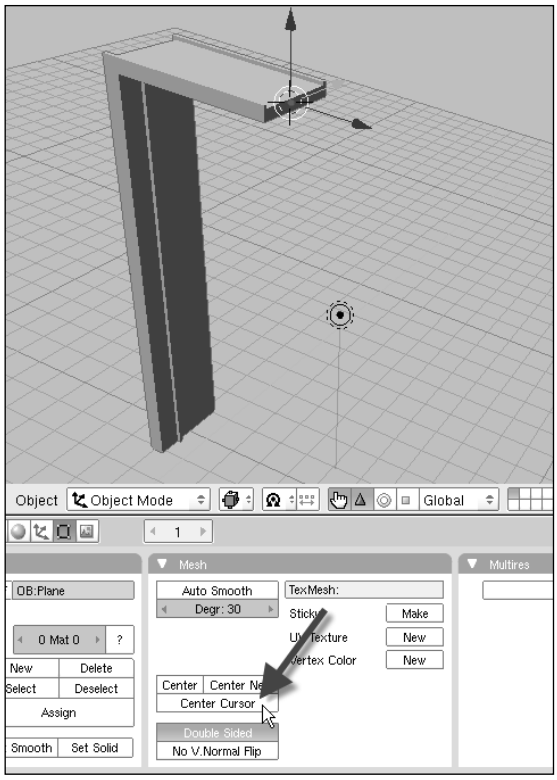
7. Select the two faces pointed to in the next image, and press the *X* key or *Delete* to erase those faces. We have to erase these faces; otherwise, they will be overlapping with the extrusion that we will be using next. Select the faces pointed to in the lower image and extrude them until they get a measure of **0.10** units:



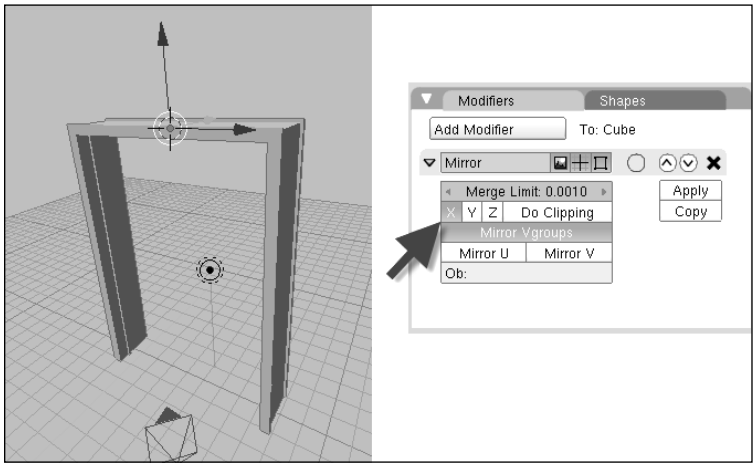
8. When the extrusion is done, select all the faces. Now extrude them until they get a measurement of **4.00**:



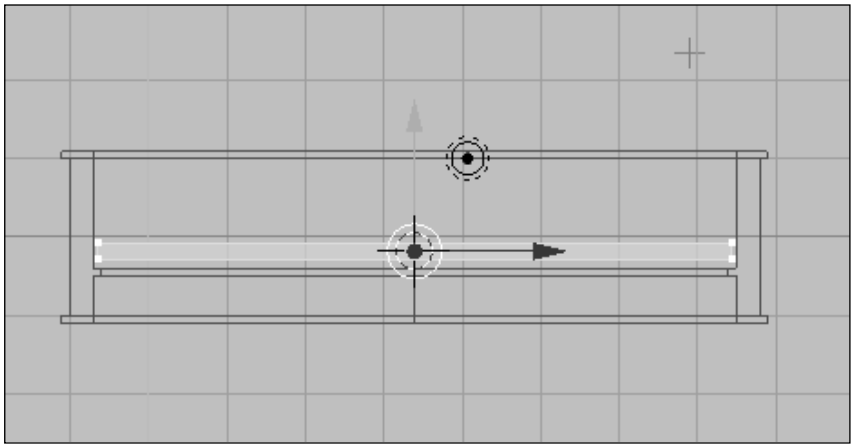
9. We have half the model ready. To create the other half, we can use a **Mirror** modifier. To use the **Mirror**, let's set the center of this model to the right position for the mirror. With the faces still selected, press *Shift* + *S* to call the **Snap** menu. Choose **Cursor | Selection** to make the 3D cursor jump to the center of the selected faces. Change the work mode to Object, and in the **Editing** panel, press the **Center Cursor** button to make the object center to be placed at the position of the 3D cursor:



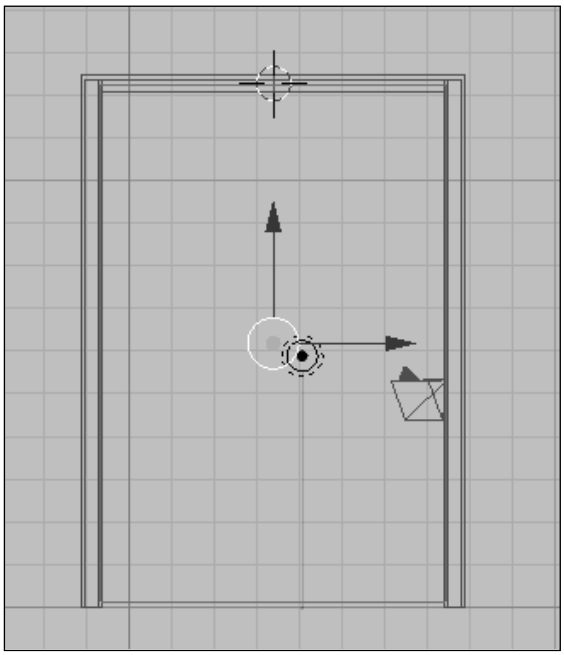
10. Now we can add a **Mirror** modifier. There aren't any special parameters to set up here, just select the right axis to make the copy. In this case, the axis is **X**, but depending on our model orientation it could be **Y** too. Just make a test. If the **X** axis doesn't give the result as shown in the following image, try **Y**:



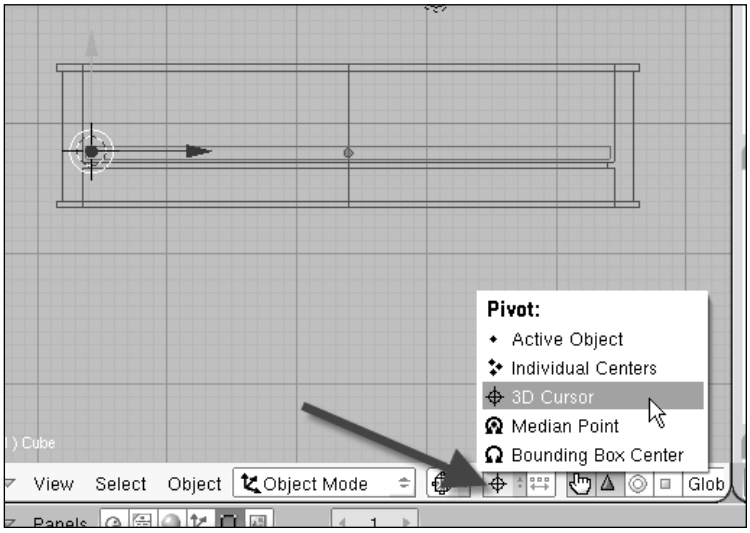
11. We have the door frame, now it's time to make the door. Change your view to Top, and create a cube. Remember to create the cube in Object mode. Use the *S* key to change the scale of the cube, to make it fit the frame:



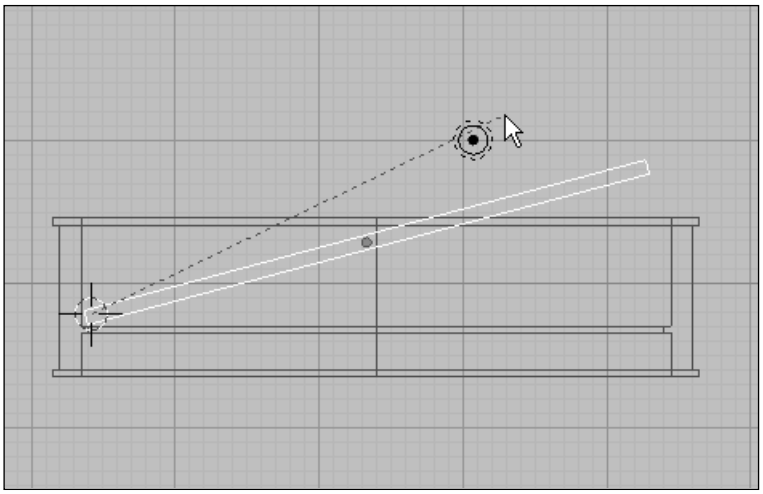
Change the view to Front, and adjust the cube again with the *S* key. Make it fit the door frame. Use the *Ctrl* or *Shift* keys to adjust it with greater precision:



1. Now that we have our model in the right size, we can rotate the cube just a bit to make the door look like it's getting opened. If we just select the cube and rotate it, the rotation will take place at its center, and won't give the effect that we want. To make it rotate at the right pivot, put the 3D cursor at the point that should be the rotation center and change the Rotation/Scale **Pivot** to 3D cursor, as shown in the following image:



2. With the pivot point in the right position, just press *R* to rotate the model. Don't rotate much, just enough to give the impression that it's half opened:



Removing doubles

Because we erased some faces in this model, it's a good practice to select all objects at the end, and with the **Specials** menu, remove all duplicated vertices. The **Specials** menu will appear when we press the *W* key.

And with that, we have our door model ready. Again, depending on the requirements of the project, we could add more detail to the frame and the door. The tools and techniques for that will be exactly the same as the ones we used to create this model. Just focus on this kind of modeling named edge modeling, which is based on extrusions of edges to build models.

Summary

In this chapter, we learned how to use the tools and techniques of Blender to create more details for our models, such as windows and doors. Even though being a complex subject, because of the many types of doors and windows used in architectural projects, we saw how to apply subdivision tools to create these models.

Some of the concepts learned in this chapter were:

- Rotation/Scale pivots
- Arrays to create multiple copies
- Applying **Mirror** modifier to create symmetrical models
- Level of detail for models

In the next chapter, we will start to model and create furniture to use in architectural visualization projects. And to create furniture, most of what has been already learned so far will be used again, but to create objects with a smaller scale.

6
Modeling Furniture

The next step for our scenes is to add some furniture, to increase the realism even more. And, the sense of realism and furniture is a key element. We can classify furniture into two main categories, internal and external furniture.

With the first type, we have all the objects that populate our interior scenes such as sofas, beds, and chairs. And the second type is about something like urban furniture, such as cars, fountains, and fences.

This kind of modeling deals with smaller scales, and because of that, sometimes, we have to work with more details than we normally do. It can make the modeling process a bit longer than usual, but only if we need to create a good level of detail for our models. In this type of modeling, we will use again the concept of level of detail. We have talked about this at the beginning of Chapter 5.

And as we mentioned there, to use the concept of level of detail effectively, we must begin our projects with good planning. Otherwise, it will be useless to do any kind of optimization without knowing the right place, where the cameras will be positioned.

Another interesting thing about furniture is that we can keep the models that we create, along with the ones downloaded from the Internet, to build a good library. With a good 3D models library, we can easily add previously created furniture into new projects, decreasing the time needed to fill scenes with furniture.

We even can download or buy models over the Internet. The only thing that we will have to do in this case is import the model into our scene.

Create models or use a library?

There are two possibilities to work with furniture; we can create models, or use pre-made models from a library. The question is—when should we use each type? The point is, some people say that using a pre-made model is not very professional, but they forget to say that most projects don't have an adequate deadline, and for that, we need a quick modeling process to be ready in time. So, what's most important for professionals? Get things done? Or, say to the client that all models were created just for his/her project?

If the deadline is the most important factor, your clients may not mind if you use pre-made models. Probably they won't even notice that. So, don't be ashamed to use pre-made models. It won't make your projects less professional. It's even recommended to use these models to speed the process, because it will make it possible to spend more time in lighting or texturing. Without enough time to work on lighting and textures, the final image may not achieve the quality and realism that we want, or even worse, the realism that the client needs.

Are there any situations that demand the creation of a furniture model from scratch? Well, there are some. First, if you can't find the model in any library that you know, then it's going to be necessary to model it from scratch.

If you are working with an architect who designs the spaces and furniture as well, you will probably have to model the furniture too, because it won't be available in any public library. Any project that deals with customized furniture will require that we create a customized model for the furniture.

Create your own library

A good practice for anyone doing architectural visualization consists of collecting a lot of 3D models from public libraries to use in future projects. Keep these models for use later, but don't forget to check if the author released this model with no restrictions for commercial use. Otherwise, you must get his permission to use it. If you want to create your own library with no restrictions, why not create your own models? This could be a good exercise. Take a few examples and start creating some furniture. With time, you will have a good number of models.

How to get started?

In most cases, we have to get used to the fact that all furniture modeling will have to start from scratch and with no blueprints available. All the references that we will have are photos provided by clients or provided from web resources.

If you have it available, take some time off to visit a real store, and take some pictures and take measurements on your own. Sometimes these stores will give you flyers and brochures, especially if you work with architecture. With time, you will get a lot of good reference material, and some of them come with measurements.

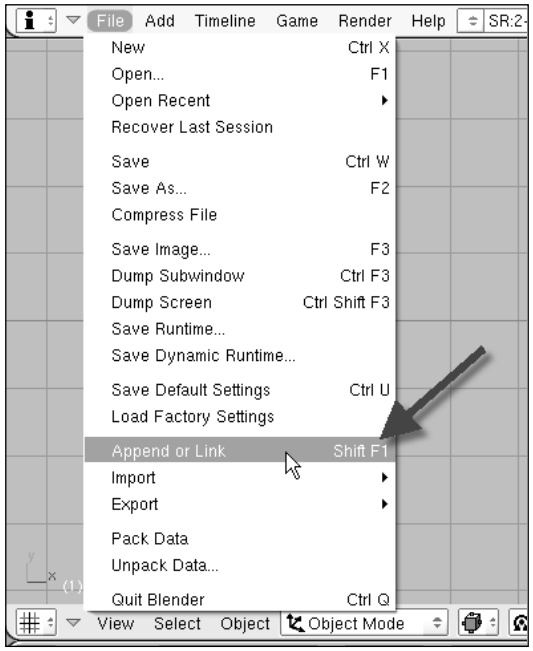
But if you don't know where to get started, let me point you to some great web resources:

- `http://www.e-interiors.net`
- `http://blender-archi.tuxfamily.org/Models`
- `http://www.katorlegaz.com`
- `http://sketchup.google.com/3dwarehouse`

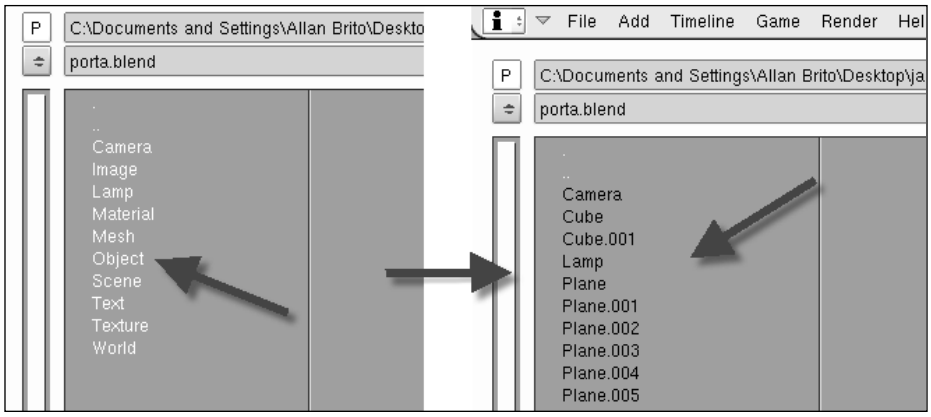
The first link has a lot of reference images, classified by furniture type and designer. And sometimes, they even provide a free 3D model. Most models there are saved in DXF or 3DS file formats, which could be imported to Blender.

Appending models

Before we get to model, let's see how we can import a model form an external library into Blender. The process is very simple. What we have to do is use the **File** menu, and access the **Append or Link** option. There is a shortcut for that too. Just press *Shift* + *F1* to call the same function:



With this option, we have to select another file already in the Blender file format. This option won't import files in different formats. When we select a file, a list of elements available in that particular file will appear for us to select what we want. In most cases, the models will be stored in **Object**:



When we click the **Object** option, all other objects available in that file will appear. If you know the name of the model which you want to import, just select the name and click **Load Library**. The object will be loaded into our scene.

Here, we have two options to handle this object; **Append** or **Link**:

- **Append**: If we choose this option, the object will be merged with our current scene.
- **Link**: With this option, an external link to the object file will be created. Any modifications to the original file will be reflected in our current scene.

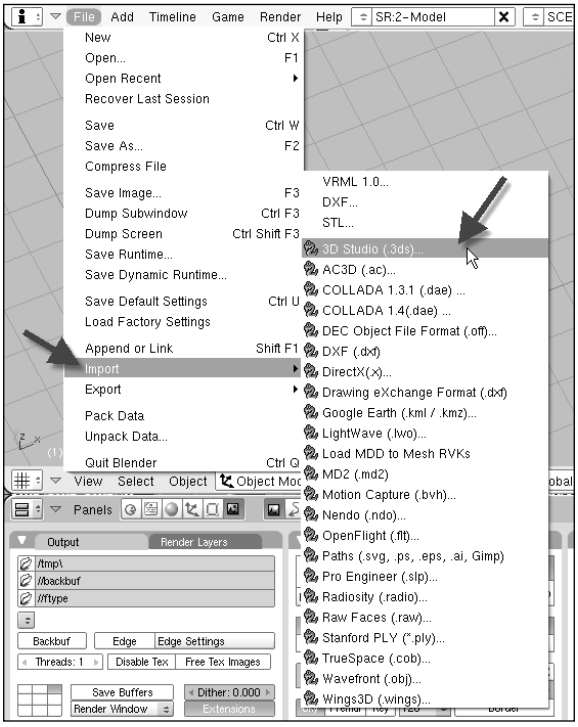
What is the best method to use? It will depend if you are willing to track all modifications applied to your furniture models. Using the Link method is a great way to keep the furniture updated, because every modification in the original file reflects immediately in any scene in which this model is placed. With the second method, we will have to take the original file with the scene file every time we need to put our scene on another computer. They always have to go together.

But if you choose to use the Append option, things will get a bit simpler, because the object will be incorporated in the scene file. We won't have to be worried about getting the furniture file along with the scene.

Always use the Append option when you want to use furniture or any other model saved in another Blender file. To open a furniture model saved in another file type and then "blend" it, we have to use the Import option.

Importing models

To import a model, the process is very simple. We use the **File** menu, and choose **Import**. Then, we have to select the proper file type from the list. The best file type, and most common to find furniture blocks, is the 3DS file format, which belongs to the old 3D Studio. There are some other good formats which work with Blender, such as OBJ and LWO:



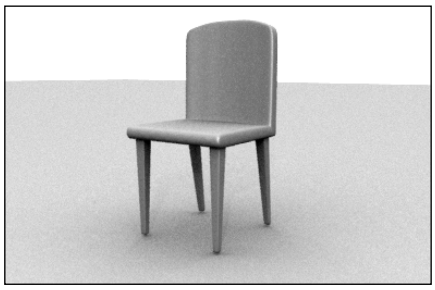
The 3DS file format can store lights, and it works well in Blender. The only thing we have to take extra care of is that most imported models come with triangular faces, which are a bit harder to edit. But, if you don't need to make any modifications to the model, it won't be a problem.

Append or Import?

Just to make things clear, if you download a furniture model from a website and it's saved in the Blender native file format (`.blend`), you will append the model. And if you download or get a furniture model in any other file format than `.blend`, you will have to import it. Because most models aren't saved in the Blender native file format, we can say that almost all furniture models that you will find will require an import action to be placed in our scenes.

Modeling a chair

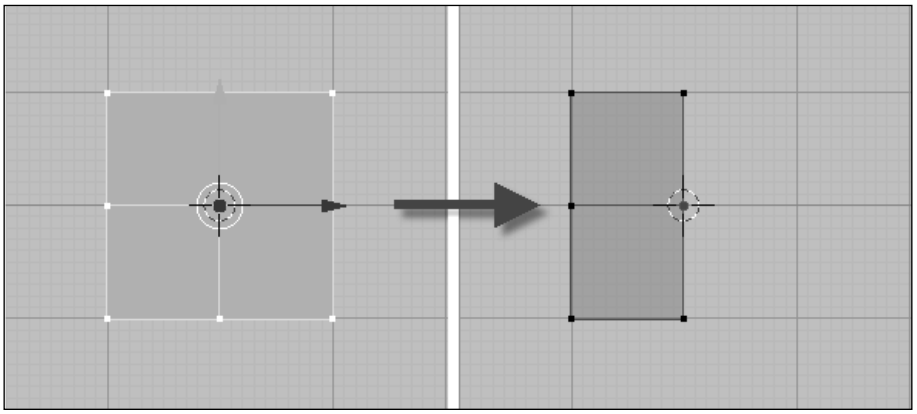
Let's start with something simpler, such as a chair. Even being a simple model, it will help us to deal with smaller dimensions and details. Here is an image of the model:



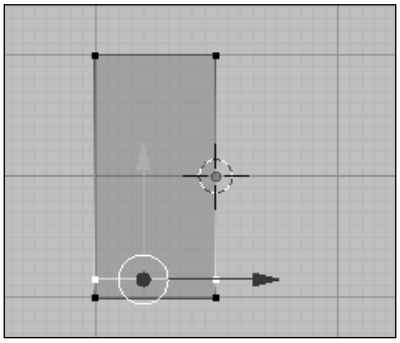
What's the main objective of this modeling? We have to create this chair with the minimum use of faces and vertices. A good amount of detail can be left for textures, and it's always a good choice to use a low number of vertices or faces in a model. If you consider one model, it won't matter much. But with a large number of chairs, such as a theater room, it can make all the difference in render time.

Let's get started in Top view. We can use a simple cube to start. Select this cube, and change the work mode to Edit. Select all vertices, and press the *W* key. This will open the **Specials** menu. Choose **Subdivide** just once in this menu; it will create new vertices and edges. When these new vertices are created, as shown in the following left image, press the *A* key to remove all objects from selection.

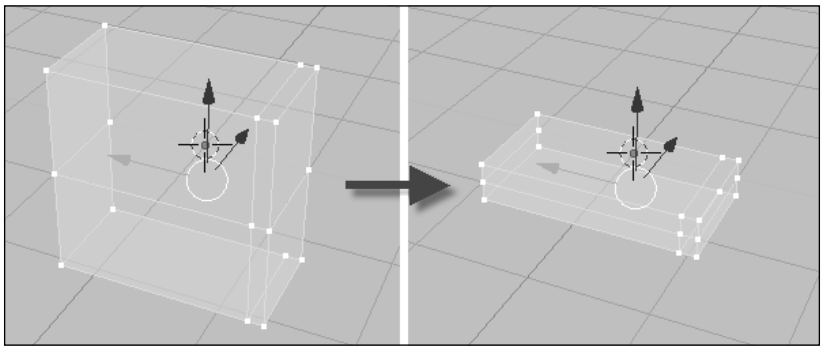
Now, select just the vertices to the right, with the *B* key. Remember to change the view mode to **Wireframe** before using the *B* key, otherwise we won't be able to select the vertices behind visible faces. When these vertices are selected, press the *X* key, and choose **Vertices** to erase only the selected vertices:



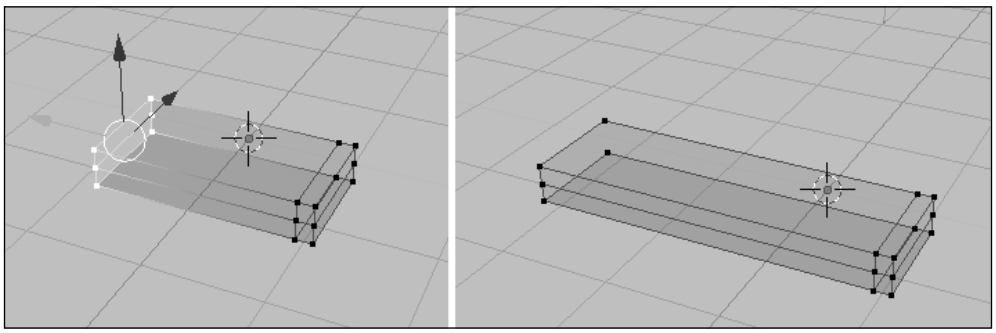
With the *Ctrl* + *R* key, add a new edge loop to the model, just as shown in the next image:



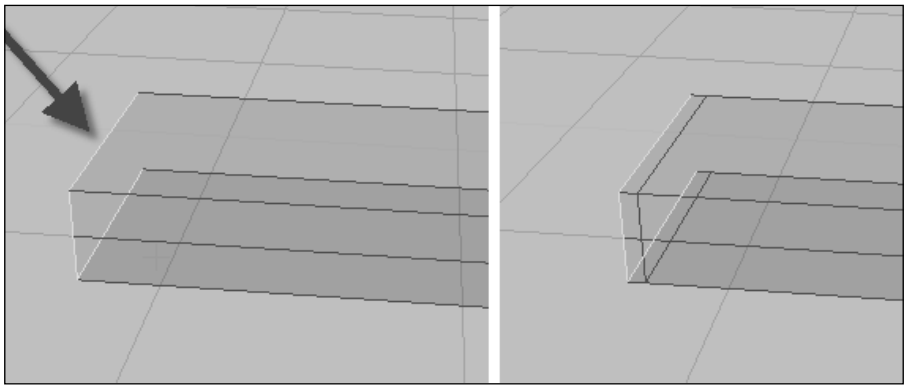
The next step is to change the scale of our model. Rotate the view with the mouse wheel to see the model with a perspective view. Select all objects, press the *S* key, and right after that, press the *Z* key. It will make the scale work only in the Z axis:



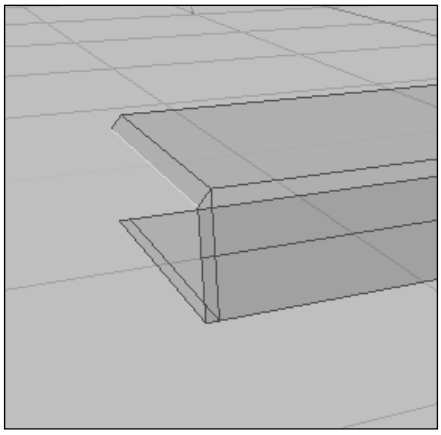
Now select the vertices pointed to in the following image and erase them with the *X* key. To remove only the faces and leave the vertices, we have to swap the **Select Mode** to **Faces** with the *Ctrl* + *Tab* key and erase the faces with the *X* key. The faces can be deleted in both ways, but with the **Faces Select Mode**, it will be faster:



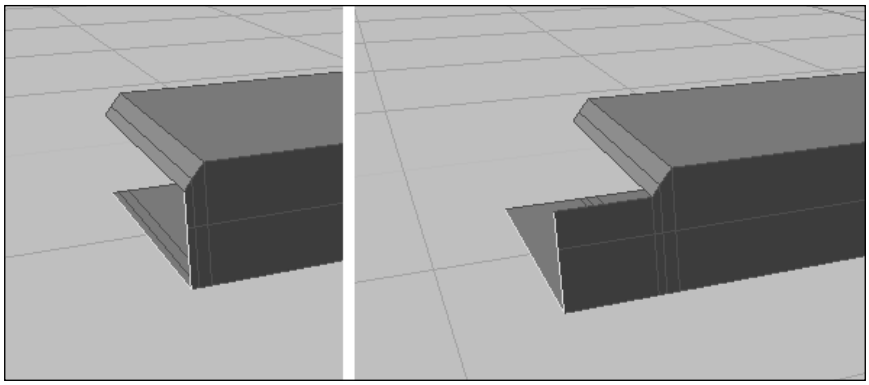
Change the selection mode to Edges, and select the edges pointed to in the following image. With the edges selected, press the *E* key to extrude them:



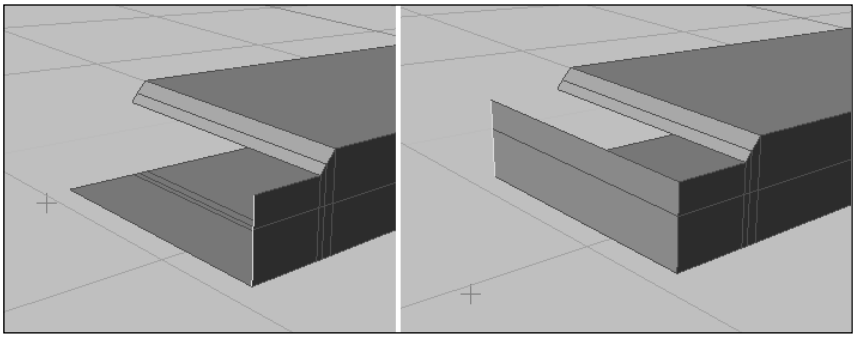
With the new faces created, we can now add detail to the model. Select only the top edge of the previously created faces. Move down this edge just a bit; it will add a small chamfer to the seat border:



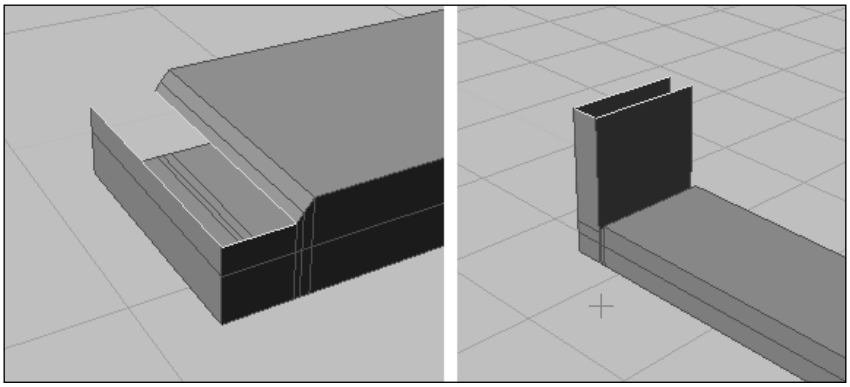
Now we can move on to the next extrude, which must be from the selected edges shown in the following image. I'm not using any kind of measure for this example, but if you like to work only with real measurements, remember to hold down the *Ctrl* key every time a new extrude or edge is moved. This way, all transformations will use the grid lines. The *Ctrl* key won't work if the **Vertex Snap** is turned on, so turn it off in case it was previously used. For this model, I'm not using **Vertex Snap**, which could use the vertices of the model to snap in key points such as a vertex, edge, or faces of the object:



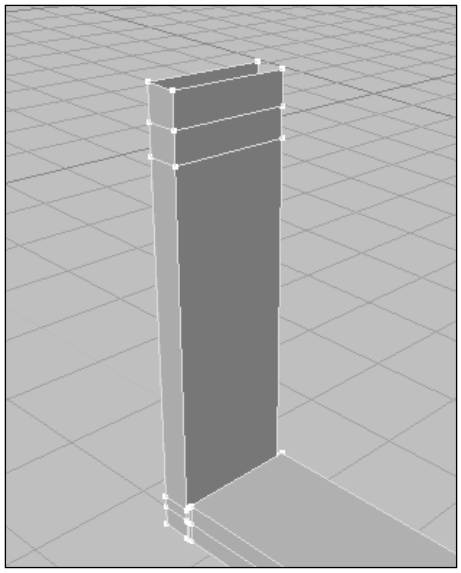
With the new faces created, select just the two edges shown in the next image. Extrude these edges until they reach the other side of the base model. Hold down the *Ctrl* key while you extrude them to help with the precision. If you already want to remove duplicated vertices, select all objects, and press the *W* key. Choose **Remove Doubles** to erase any duplicated vertices:



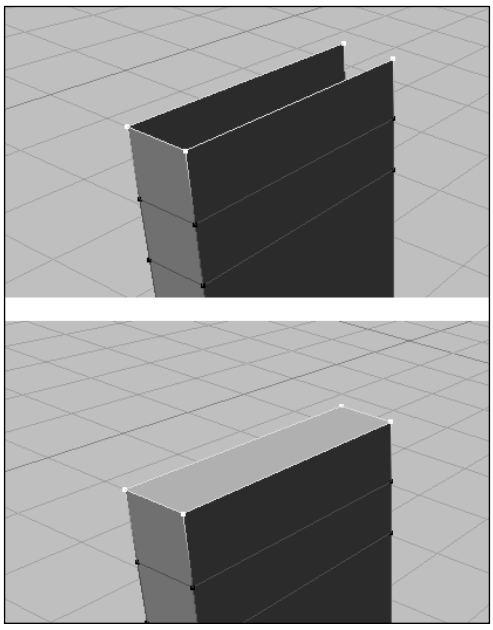
Select the edges shown in the image to keep adding more parts to the chair:



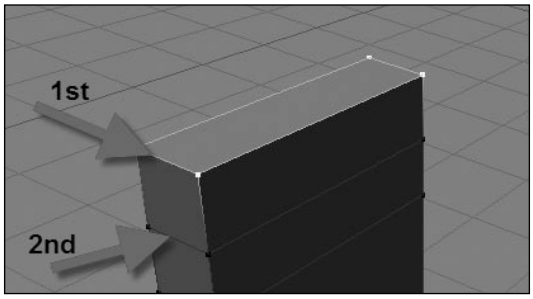
Extrude the edges three times, until you get the same structure as shown in the following image:

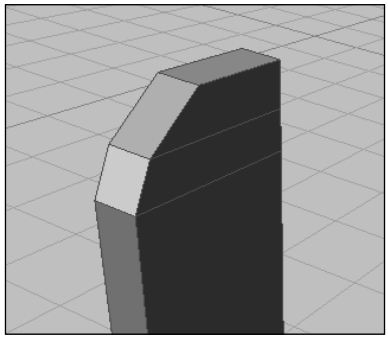


Now, we have to close the top with a face. To do that, we must select all four vertices from the top. When the vertices are selected, press the *F* key to create a new face:

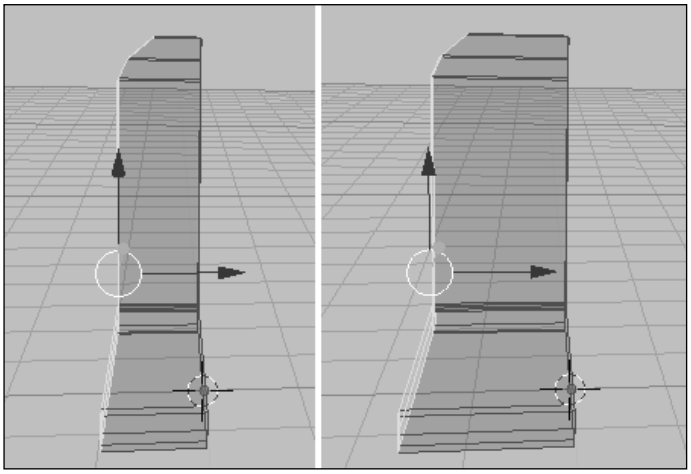


The next step is to select the small side edges to create detail. Select just one edge, beginning from top to bottom, and move it just a bit. Repeat this operation with the other edges, until we get the edges placed as in the following image:



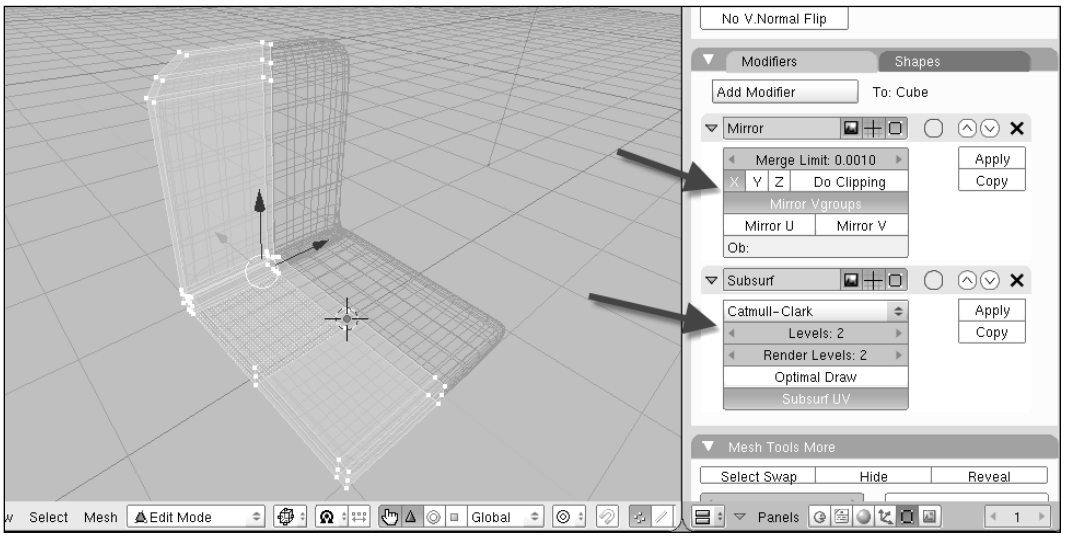


The basic shape for our chair is created. Now, we can make adjustments for improving the overall proportions. Select all edges or vertices on the left side, and move them a bit to the left. It will make the model wider:

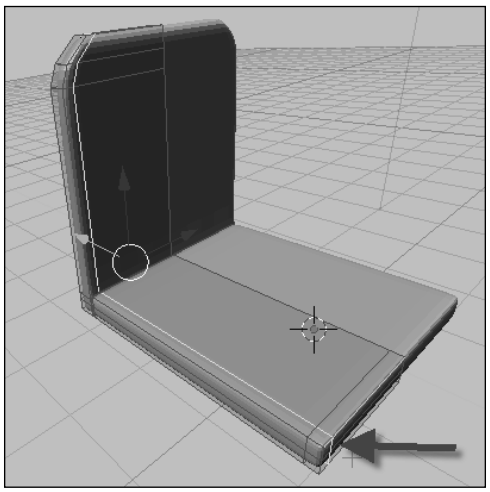


Did you notice that we have modeled only half a chair? Now we can make the other half with the **Mirror** modifier. Add the modifier, and choose the right axis to make a perfect copy. If the center point for the model has been moved, you might need to edit the model to create a perfect mirrored match. Don't worry if you have moved the model by accident, sometimes it can happen. Also, turn on the **Do Clipping** button to prevent the two sides of the model from running over each other.

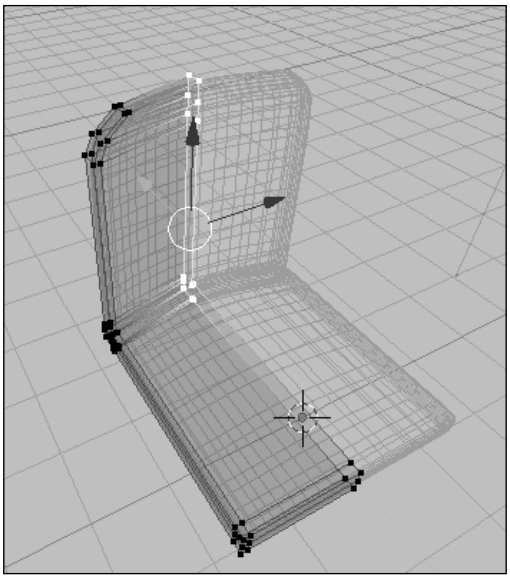
Along with the **Mirror** modifier, add a **Subsurf** modifier as well:



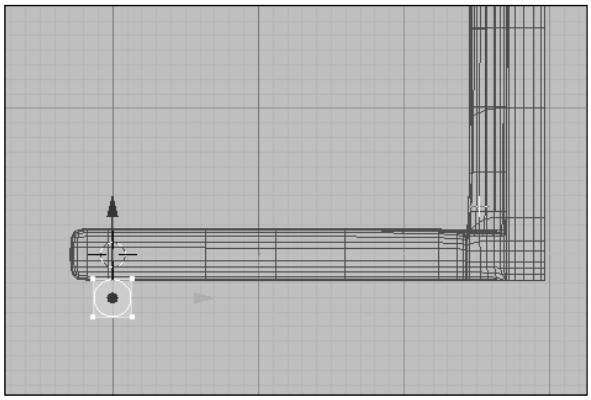
With the **Subsurf** modifier, we realize that this model needs a new edge loop at the left side. Just press *Ctrl + R*, and add a new loop, as shown in the following image:



We can make the back seat of the chair a bit curved. Just select the vertices pointed to in the next image. Select them, and move them just a bit to the back. With the **Subsurf** modifier, we will be able to create a curved surface. Along with this curve, we can make adjustments, such as sizing down the seat:

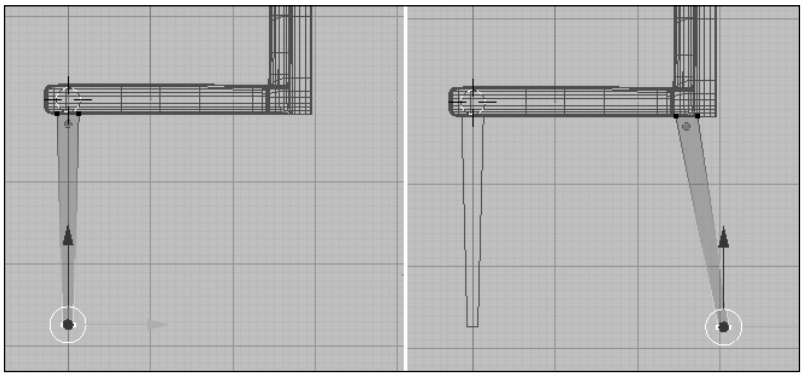


To finalize the model, we must add the support. Just add a cube, and size it down until it looks like the following image:

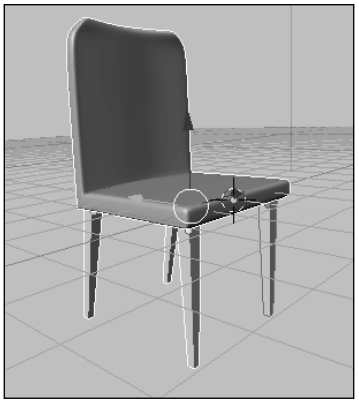


Change your view to one of the side views and press the *3* key on the numeric keypad. Select the vertices at the bottom of the cube and move them down. When you place the vertices at the bottom, press the *S* key to scale them down. Create a copy of this cube, and place it at the back of the chair.

With this copy, select the bottom vertices, and move them to the right:



And now we have a chair model! We use the same principles required to model more complex objects. All we have to do is use the tools wisely:



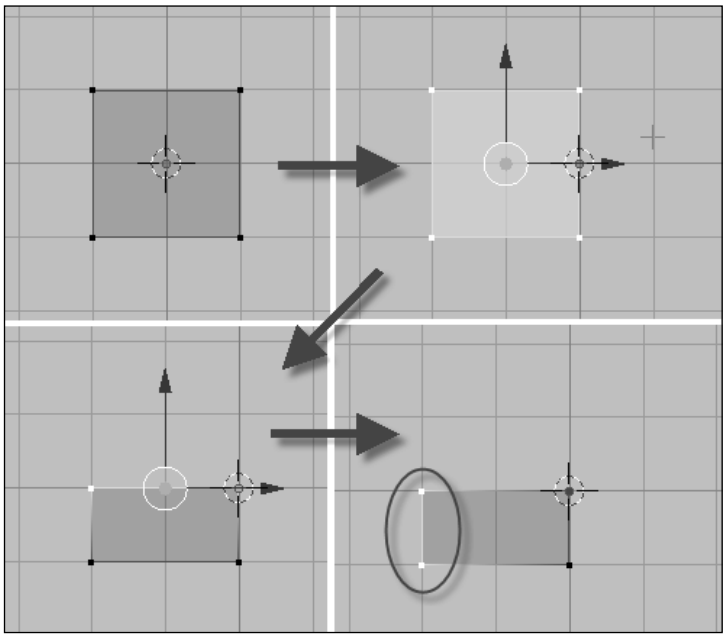
Modeling a sofa

It can be very simple to create some kinds of models, as we saw in the last example. Almost every object can be modeled from a cube, and with some extrudes and a few adjustments, generate a complex shape. But it only looks complex. It's the same way with our next example, which will be a sofa created from three cubes. The cubes will turn into the body, front seat, and back seat of the sofa.

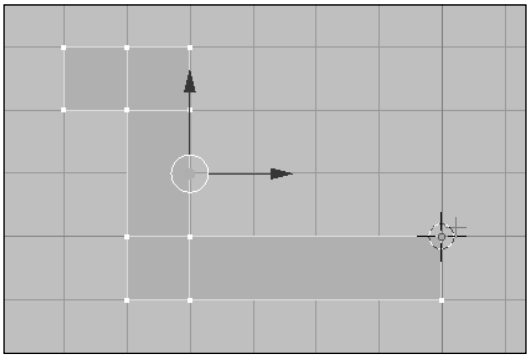
The first step to create the sofa is to add a cube if you don't have any object in the 3D view. Change your view to see the model from the front view. With the cube selected, change the work mode to Edit, and select all vertices of this cube. In this example, every transformation will be created holding down the *Ctrl* key. This way, we will have more control over the proportions of the model.

When all the vertices are selected, move the cube a bit to the left. See in the image that the 3D cursor will be placed at the right side of the model. It means that the object center is placed there as well.

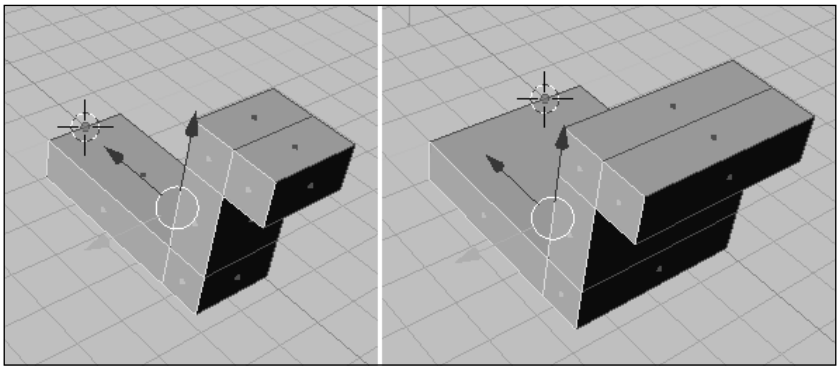
Remove all vertices from the selection, then select just the top vertices with the *B* key and move them halfway down. Hold the *Ctrl* key to make it easier. To finish this part, select only the vertices at the left side:



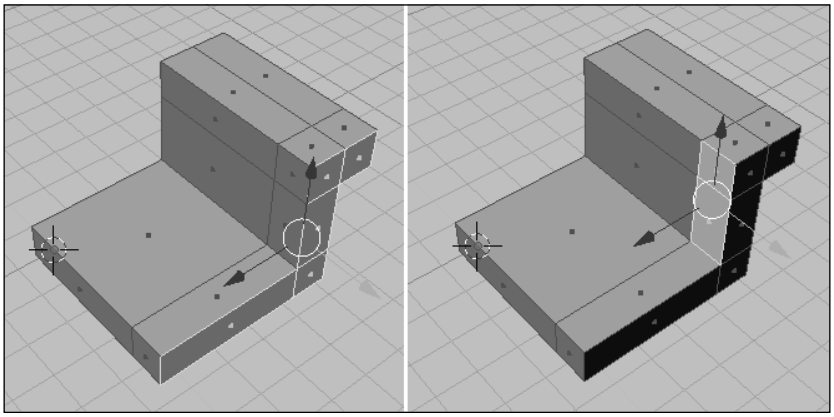
With the extrude tool, we create a structure like the one shown in the next image. The first thing to do is to move the vertices a bit to the left, and then with the *E* key, extrude the faces. If you notice, all the new faces and edges are perfectly aligned with the grid lines; this occurs because we are using the *Ctrl* key:



Change the select mode to Faces and rotate your view. Select the back side faces. When these faces are selected, just move them to make our model a bit bigger:

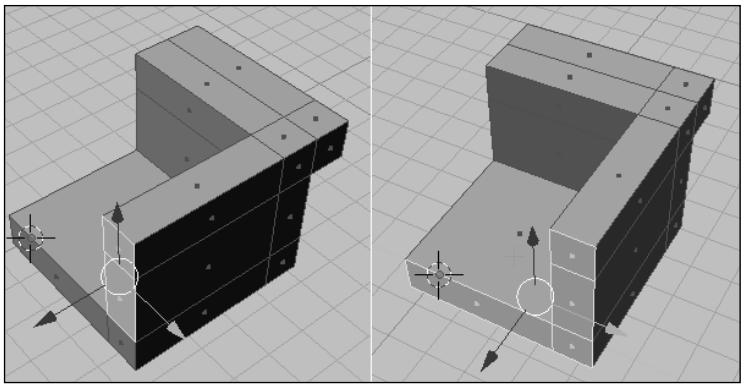


We must now rotate our view to select the faces pointed to in the following image. These faces will be extruded:

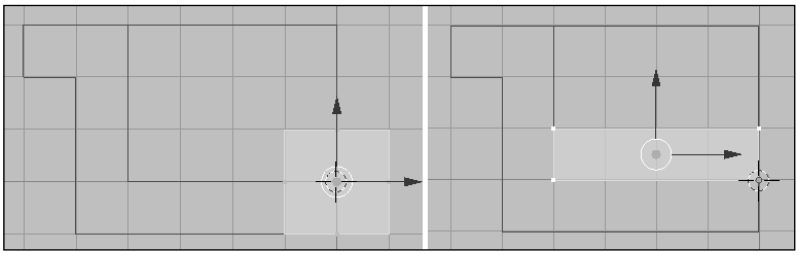


Press the *E* key to extrude the faces, again with the *Ctrl* key pressed. At the end of this process, select all the faces and press the *W* key, and choose **Remove Doubles**. We have to remove any duplicated vertices to clean up the model, because we created overlapping faces. But don't worry; with editing, it won't have much effect on the final model.

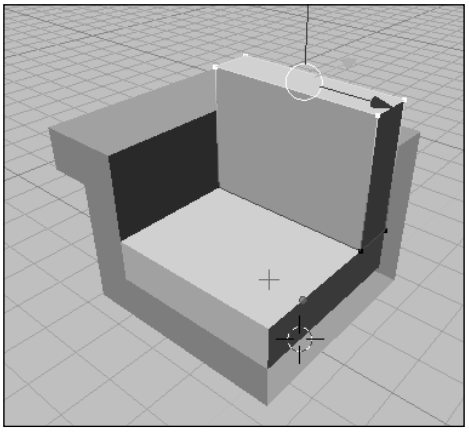
Another thing before we go ahead—select the faces pointed to in the image on the right. We have to erase these faces, because a **Mirror** modifier will be applied to the model. If we don't erase those faces, they will be used in the **Mirror** modifier. The result will be overlapping faces. By the time we apply the **Subsurf** modifier to the object, both borders will be smoothed, and they won't be visually connected. When the faces are selected, just press the *X* key, and choose faces:



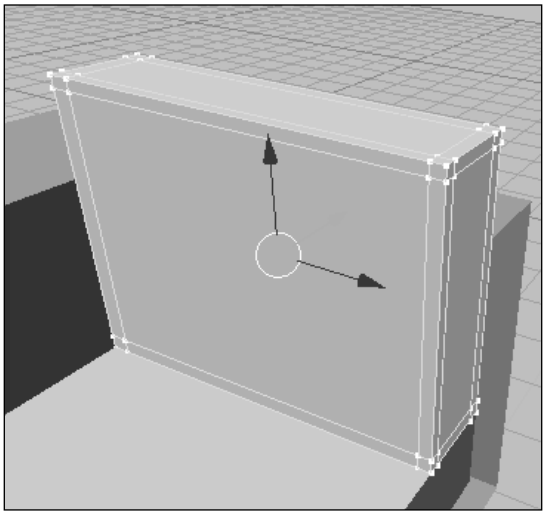
The bigger part of this sofa is created. Now let's add some more details. Change your view to see the model from the front. Then create a cube and change the size of this cube to fit the place for the seat, just like it's shown in the image to the right. If you need, change the view to adjust the model from the top:



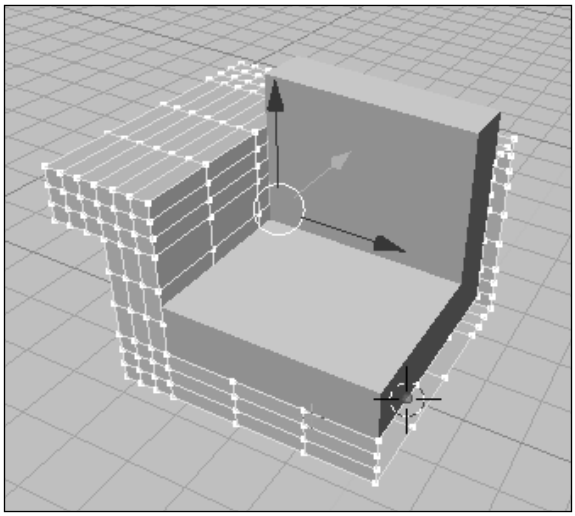
Create a copy of the cube, and adjust it's size to make it fit the back seat. A good thing to do here is to add more edge loops with the *Ctrl* + *R* key. It's needed because we will apply a **Subsurf** modifier. With more edge loops, we will be able to control the amount of smoothness added to the model with the **Subsurf** modifier:



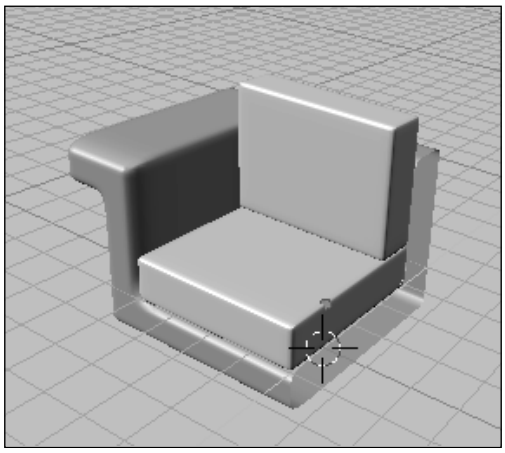
With the *Ctrl + R* key, add six new loops to the cube. Repeat the same process for the other cube. We have to add these new loops close to the existing edges to control the curvature created by the **Subsurf** later:



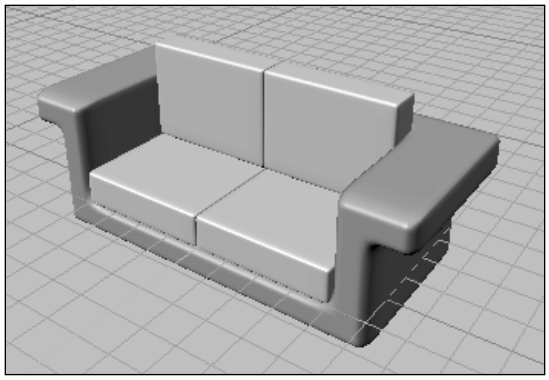
Select all vertices from the bigger part of this sofa and press *W*, and then choose the **Subdivide Multi**. As factor to the subdivision, choose **2**, and we will have the next image shown as the result:



To complete this phase, add a **Subsurf** modifier to all objects. Because we added a lot of new edges loops, there won't be any major problems in the resulting object. But, if you think there is something else to adjust, feel free to add more detail:



With a **Mirror** modifier added to each object, we will see the final result of this modeling—a simple sofa, which can get more detailed with a good set of textures or lighting:



Building a library

You don't have to model all the furniture in the same scene of a complex model. For instance, for a scene of a dining room, the walls can be modeled in one scene. The table and chairs can be modeled in different scenes. And to gather them all together in one single scene, we will use the **Append** option. If you follow this simple action (model single objects in individual files and append them to the main scene), after a few projects, you will build a significant number of individual furniture files, which will become your own furniture library.

Summary

We learned in this chapter how important using furniture in our models and scenes is, to give all environments more details and realism. Some of the aspects that we have learned are:

- We can model our furniture and use a pre-made library as well
- Sometimes it is better to use a pre-made model
- How to build your own furniture library for reuse in future projects
- How to model a chair
- How to model a sofa

7
Materials

After passing through the entire modeling process, now we can work on materials for our objects and the overall scene. But what are materials? And what they can do for us? With materials, we can set up an object or surface, and determine how it interacts with light. This is the main concept. Materials will be used to control how a surface reflects or doesn't reflect the light. If we start to think about it, this concept makes a lot of sense. The main difference between a stone surface and a wood surface is how it reflects visible light. Of course, I'm not talking about physical properties! Actually, the colors of all the objects that we see are the colors of the reflected light from the object's surface. This is how we see an object in the real world, and also in the 3D world.

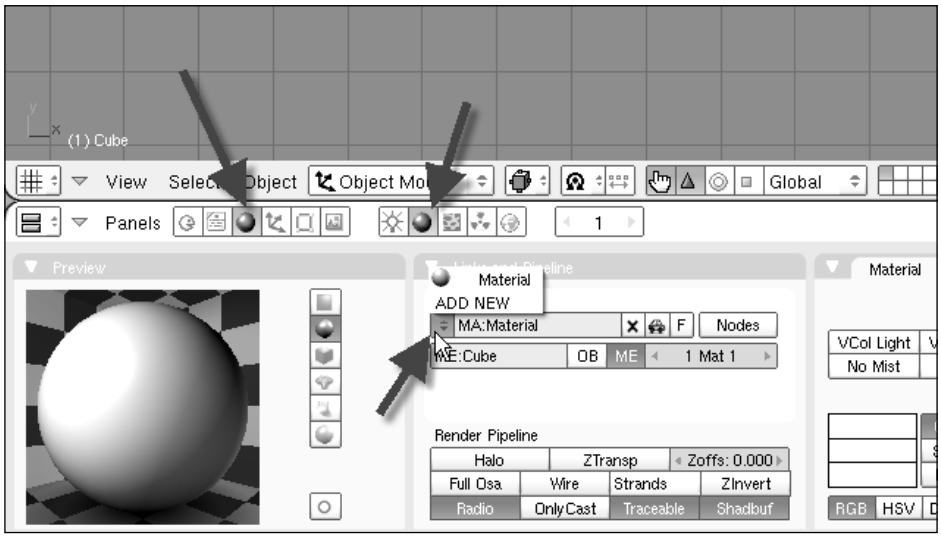
This is very important to understand, because a lot of parameters of materials are based on how the material will react to light.

Because of this interaction with lights some artists usually set up the environment lighting before setting up the materials. This will make the adjustments a lot easier in some cases, but for the overall process, you can start to work on the materials before lighting. Don't take it as a rule; try to set up materials before and after, and see what works best for you. There is nothing like trying it yourself!

Now that we know more about materials and how important they are to improve the realism of your scenes, let's start working with them.

Creating and organizing materials

Before we start to use materials, let's see how to create and organize them. To apply a material to an object, we have to first select the object, and choose the material from the **Shading** panel:



To select a material, we use the combo box pointed to in the image. If we don't find any material that matches the properties of this particular object, we can always create a new one with the **ADD NEW** option. Sometimes, it can seem more practical to create a new material for every object, but avoid this kind of behavior. Try to use existing materials as much as possible to optimize the resources of a scene. If you already have a glass material created, there is no point in creating a new material for another object if they have the same configuration. Unless they have different properties, try to use the same material.

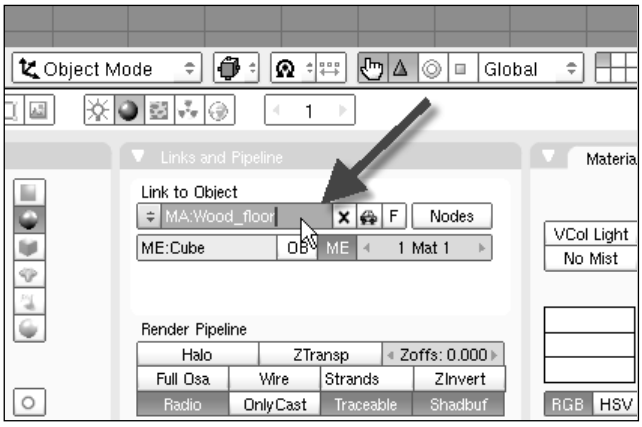
Shading Panel

To open the **Shading** panel, we use the hotkey *F5*.

Besides creating these materials, another good practice is to set unique names for all materials. If we take a closer look, Blender always gives materials an auto-generated name, such as **Material.001** and **Material.002**. If you don't want to pick a name for the material, use the **AutoName** to generate a new name for the material automatically.

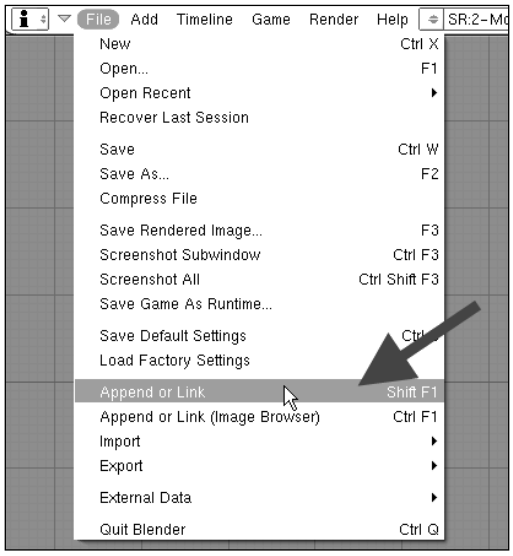
If a material has a name that represents the properties of a surface, it will make the process of searching for a specific material a lot easier. For instance, a material named **Window_Glass** means a lot more then something named **Material.005**.

Changing the name of a material is very easy; just select the material and type new name in the text box, pointed to in the following image:



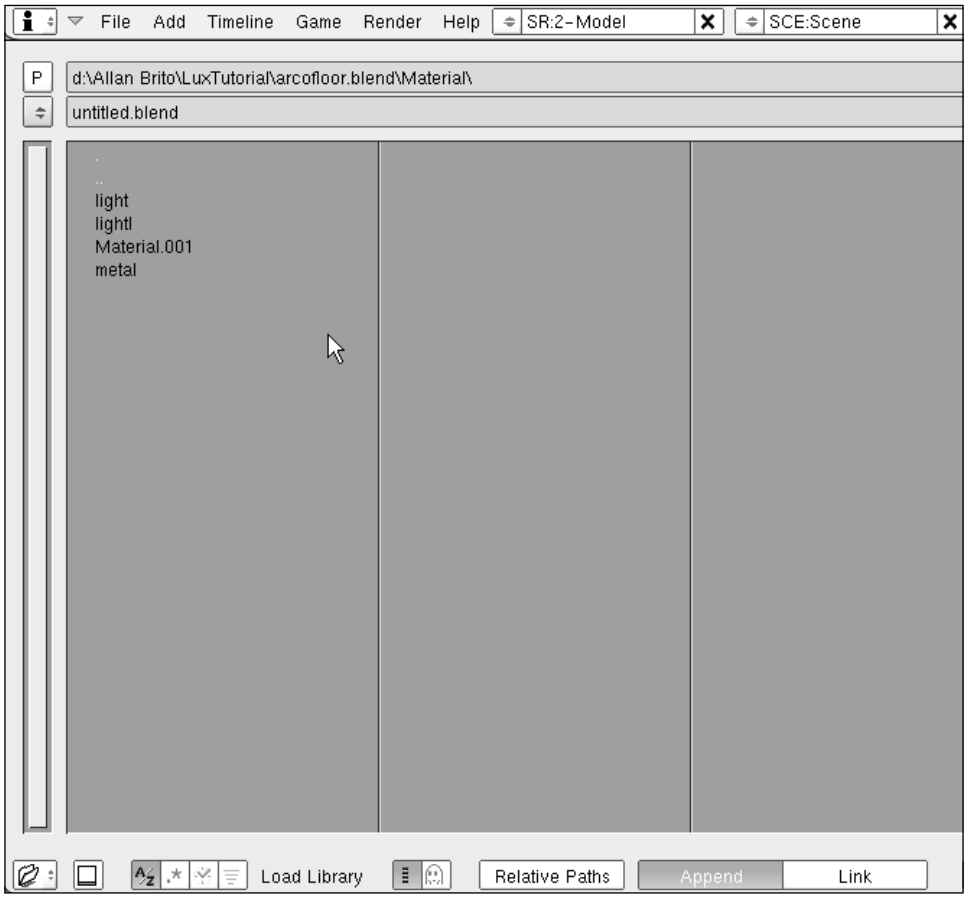
By the end of a big project, we will have a lot of materials organized and classified with semantic names. The best part of organizing the materials is that we will be able to use them again in future projects. And yes, we can import materials from other files with the **Append or Link** option in Blender.

To use this, we use the **File** menu, and choose **Append or Link**. The same option can be called by using the hotkey *Shift* + *F1*:



When we choose this option, a new window will appear. Then, we have to find the file that contains the materials that we want to import. Select the file, and browse through it's contents. Of course, we have to select **Materials** to see the list of materials available to import.

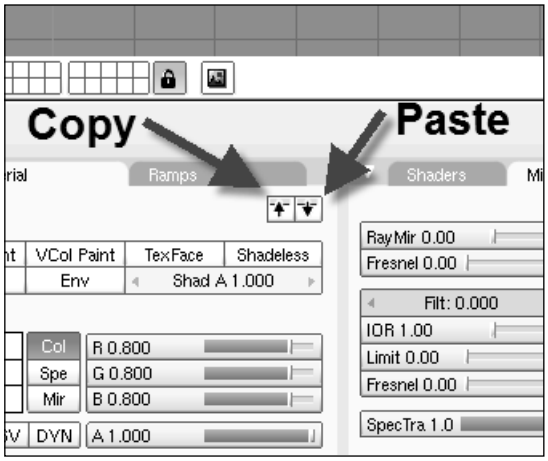
This list shows the name of all materials available in that file. See how important it is to give materials a good name? In this situation, if you don't remember what kind of surface the material represents, a good name can be very helpful:



When you finish selecting the material, press the **Load Library** button.

And what if we have a material? For a particular object, can this material be used with almost all properties? Well, suppose that you have a scene where two objects that will be represented with glass have almost the same properties. We can create one material, and then make a copy of this glass material, and paste its properties into another material slot. Then all we have to do is make the necessary adjustments.

To create these copies, we use two small buttons. The button on the left copies the material's properties, and the button on the right pastes the information into another slot:

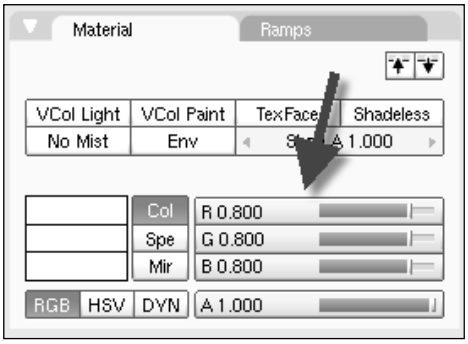


Material color

Now that we know how to set up names and import materials from other scenes, let's start to work with the look and feel of materials. The first and most basic aspect of any material is its colors. To give a material a particular color, we have two options to set up the color—a solid color or a gradient color.

Solid color

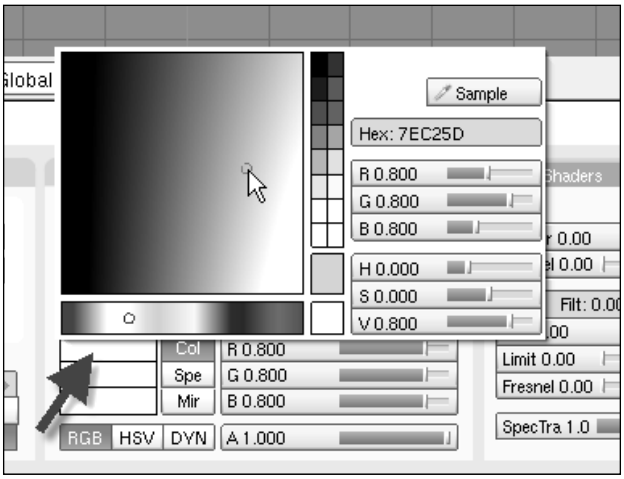
The first one is the simplest, and we choose it with the color picker:



Just click on the small rectangle, and choose the desired color. If you find it more useful, we can use a mixer and choose the color with the blend of different indexes. In Blender, we have three different color systems available from which to choose **RGB**, **HSV,** and **DYN**. Here is a description of each of those color systems:

- **RGB**: With this option, we will choose a color with the mixture of Red, Green, and Blue colors. For instance, to get a light blue color, we must use **0.7** of Red, **0.9** of Green, and **0.9** for the Blue.
- **HSV**: To choose a color with the **HSV** model, we will use the Hue value. The light blue color used to demonstrate the **RGB** model has a Hue of 0.55. Along with the Hue, we will set the Saturation of the color and the Value, which works like a brightness control of the color.
- **DYN**: This option is related to Dynamics used in the Game Engine of Blender.

Looking with an artist's point of view, the color picker is the best choice, but feel free choose the color any way you like it:



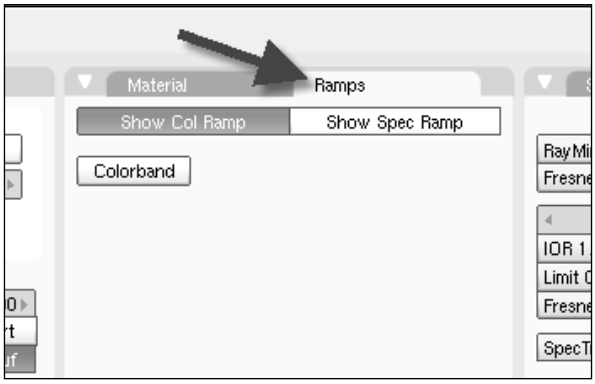
If we take a closer look, there are three color pickers, which can set up the color with three different aspects of the material:

- **Diffuse Color**: This is the color reflected by the material when illuminated by pure white light. In other words, this is the color of the material.
- **Specular Color**: With the specular color, we will set how the highlights in a surface will be represented. The specular color should be set with the same color as the main light source on the scene, to achieve better results on hightlights.

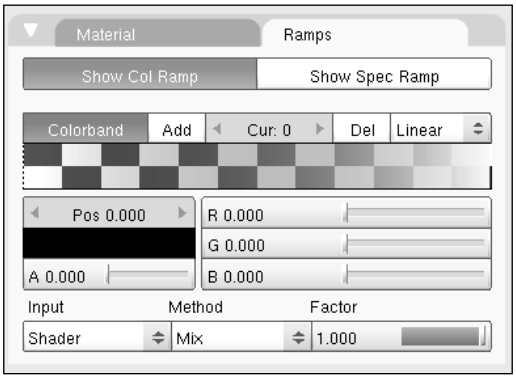
- **Mirror Color**: The mirror color will be used in reflections, to add color to the reflected images. This is useful to represent some kinds of metals and plastic, when the surface of the object is not chrome or a mirror.

Gradient colors

Besides the solid colors, we can choose gradient ramps for our materials. To use this category of color, choose the **Ramps** tab:



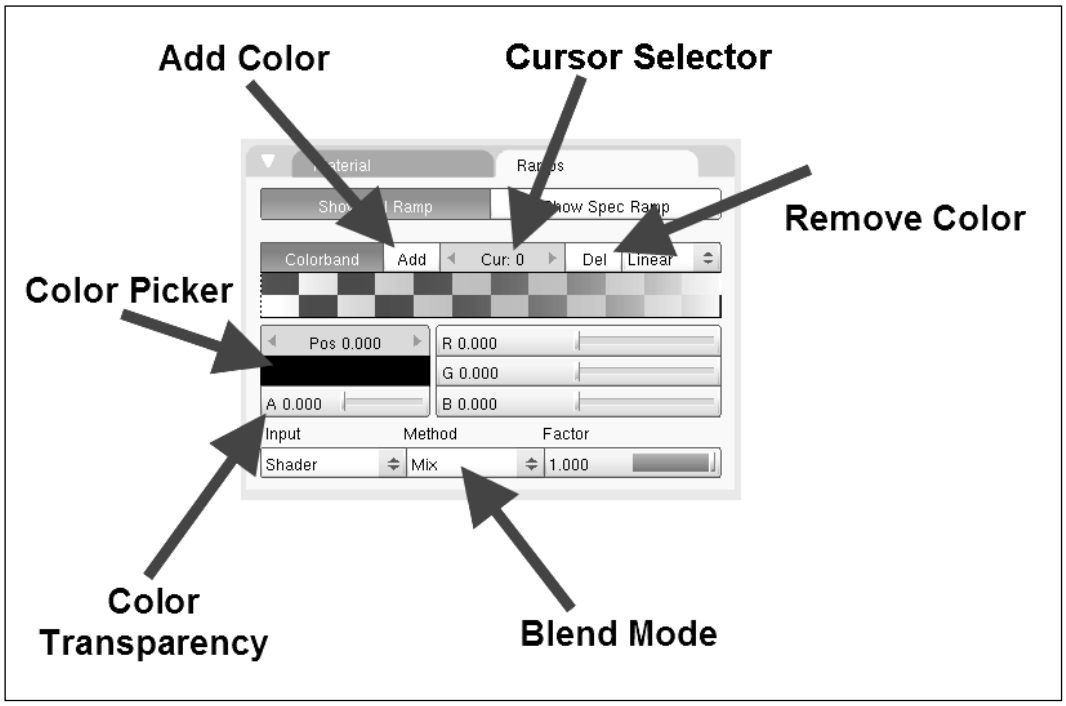
In this menu, we can turn on the use of color ramps for materials. With these color ramps, we can add even more realism to the material by making them sensitive to the amount and angle of the light that hits the surface of each object. This is exactly what happens in the real world, where the diffuse and specular reflections will suffer minor changes depending on the way light behaves on the scene. There are two options—choose **Show Col Ramp** to diffuse, and **Show Spec Ramp** for specular color. When we turn any of these options on and press the **Colorband** button, the **Ramps** menu will show all options available to set up these colors:



To work with these menus, we have to understand how they manage colors. First, there is a horizontal bar, which represents the gradient. All gradients are created by a collection of color indexes, which are represented by numbers. At the beginning, only two indexes are created. The position of an index is represented by a vertical white line, representing the relative position of the index with a fraction between 0 and 1. We can easily move this line by clicking on it and dragging the mouse.

When the line is moved, the color position and the gradient changes.

Let's see a few more important options for these gradients:

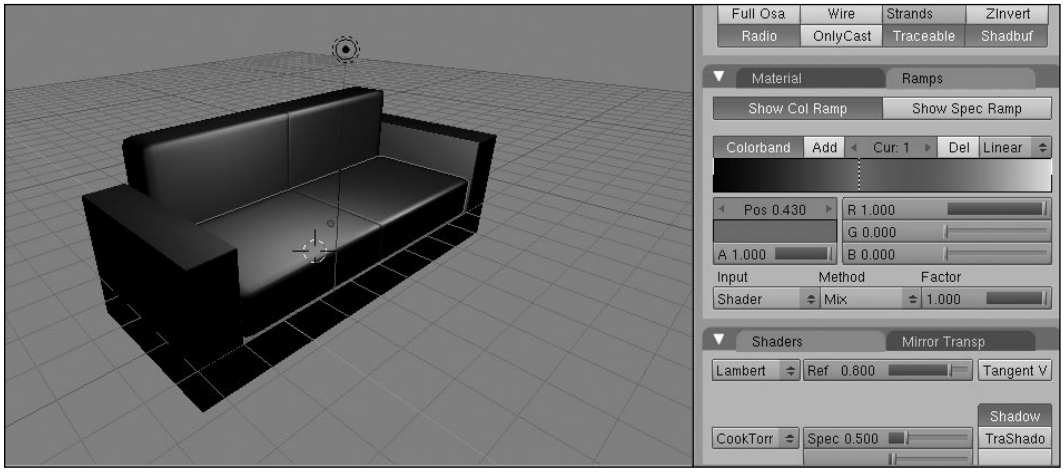


- **Cursor selector**: With this option, we select the active index.
- **Add Color**: Here we can add a new color index if we need more than two indices.
- **Remove Color**: If you want to remove an index, use this option. Before hitting this button, set the index that must be removed as the active index.
- **Color transparency**: Here we can set up how transparent the color will be. The **A** represents the alpha, and for a value of **0**, we have total transparency, and **1** signifies full opacity. The background of the material will be visible at the rendering.

- **Blend mode**: This option determines how the ramp will blend with the material color and background.
- **Color picker**: Here we have a color picker to choose a different color in a more visual way.

With these options, we can set up a good gradient color for our materials, and give them a very organic look. Which are the best situations where we can use these ramps? Here are some objects and surfaces which can easily be represented by ramps:

- Vegetation
- Velvet
- Skin
- Fabrics



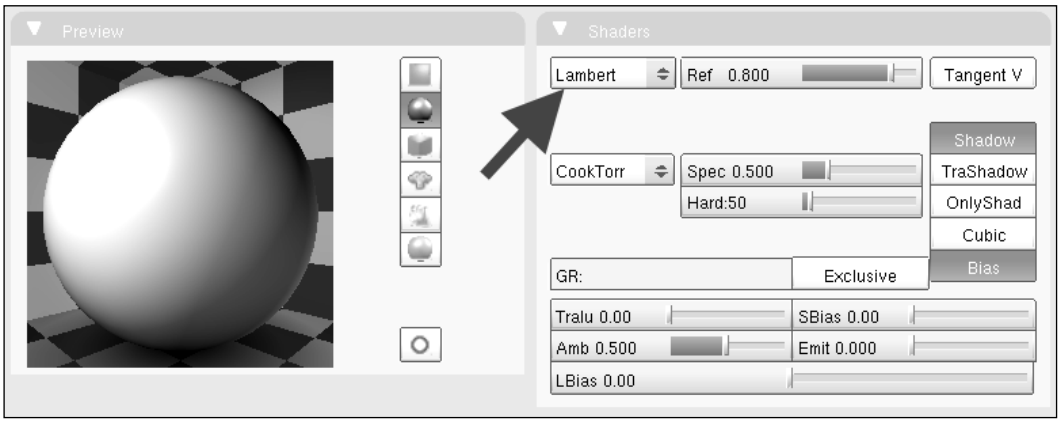
Shaders

Shaders can be used to set up how a material reacts with visible lights. In the material panel, we can choose two types of shaders, which are diffuse and specular shaders.

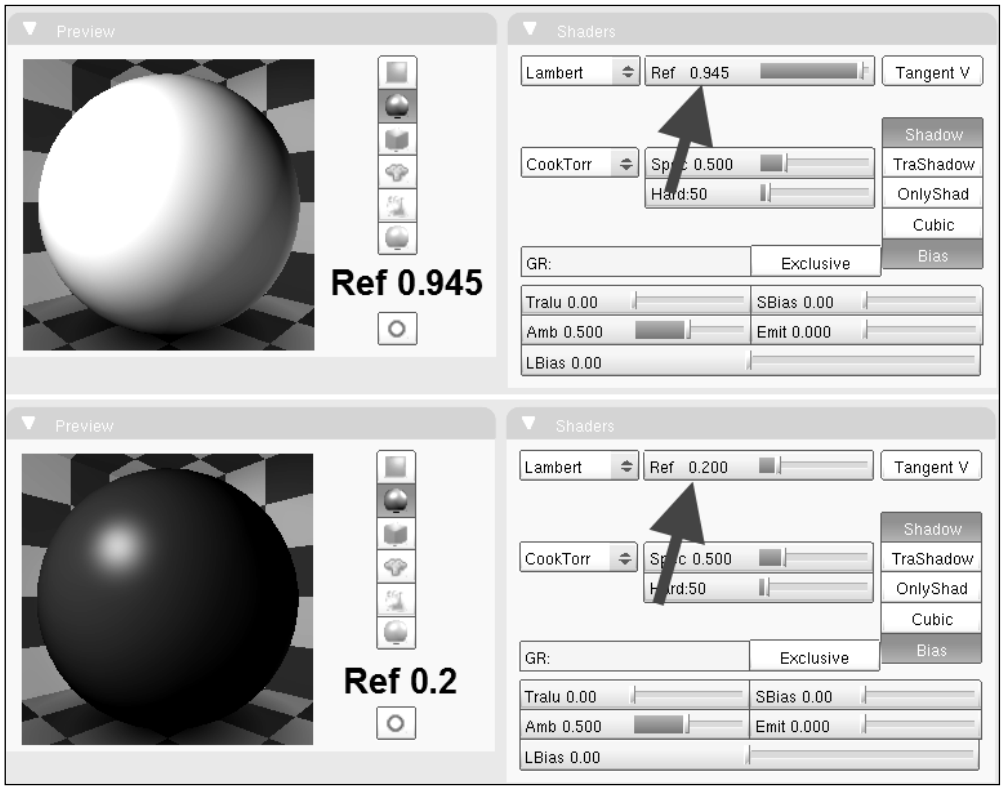
Diffuse

A diffuse shader determines how much light is reflected by the surface. Let's make things clear here—it doesn't mean that the object will generate illumination. If we set up a high reflection value for the diffuse shader, it will mean that the object will be brighter.

We have five different kinds of shaders in the Blender materials panel:



All these shaders share common parameters, like the **Ref**. This option sets how much light is reflected. So, if we set up a high value, the material will look very bright, and lower values will make the material darker:



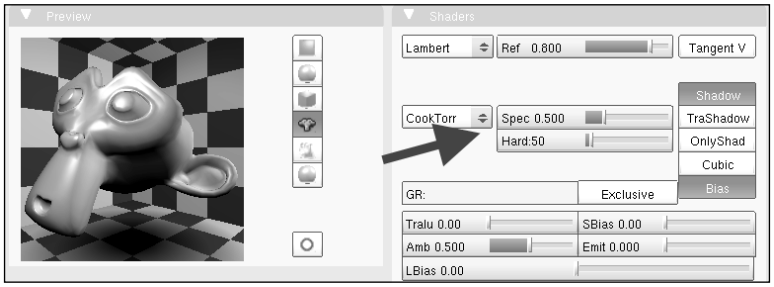
What is the best setup? It will depend on the type of surface that you want to simulate. The key to set up a good diffuse shader is the **Ref**, along with several other options available in each shader type. Here is a description of each shader:

- **Lambert**: This is the standard diffuse shader for Blender, and it is the simplest of them all. The **Lambert** shader will always reflect the visible light with the same intensity. It is independent of light or viewer angle. The only parameter available for it is the **Ref** slider, which will determine how strong the reflection of diffused lights will become. A good example of material that can use the **Lambert** shader is unfinished wood or a few types of stone.
- **Oren-Nayar**: In comparison with the **Lambert** shader, the **Oren-Nayar** adds more realism to the light reflection, because it uses rough surfaces and takes into consideration the small imperfections of the surface to reflect light. It is a great choice for concrete and wood. We have the **Ref** parameter and a **Rough** parameter that control how much of the roughness of the surface will be used to reflect light.
- **Minnaert**: With this shader, we have an effect that makes parts of the surface dark when pointed towards the viewer. There is a **Ref** parameter, like all other shaders, and a **Dark** parameter that controls how dark a surface will be when pointed to the viewer. The materials that can be created with **Minnaert** are sand, dirt, oxide, metal, and others.
- **Fresnel**: Here we have a shader that will be entirely dependent on the light angle over the surface. The side perpendicular to the light will become darker, and the opposite side will be bright. There are two other parameters besides the **Ref**, which are **Fresnel** to control how strong the **Fresnel** effect will be, and a **Fac** parameter to set the blending factor of the **Fresnel** and diffuse reflection. With this shader, we can create organic materials or surfaces, such as velvet, that depend on the light angle, such as velvet.
- **Toon**: If the objective is to create a stylized material, we can use the **Toon** shader to achieve that. The highlights on this shader can be created using a hard-edged shape, similar to an animation cell.

Specular

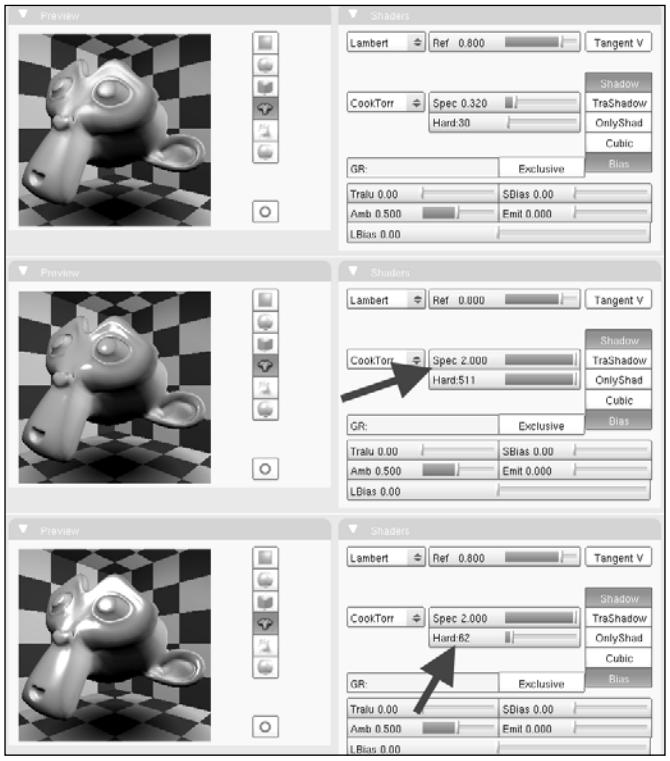
The Specular shader controls a kind of reflection of light, which depends on the point of view and the angle at which the light hits the surface. This shader is very important to set up materials such as glass, metals, and other reflective surfaces. All those surfaces have a particular way of behaving with reflections. In the following image, we can see the Specular shader creating highlights on the monkey image.

Just like the Diffuse, we have five different shaders here. All those shaders share common parameters. The most important are **Spec** and **Hard**:



- **Spec**: This option will set up how hard the Specular reflection is. High values will make the reflection stronger.
- **Hard**: With this option, we can make a soft or hard border for Specular reflections.

With these two options, we can simulate a many materials. For instance, surfaces formed by glass objects have a high **Spec** and **Hard** values. Surfaces like concrete have a median **Spec** value and a very low **Hard** value:



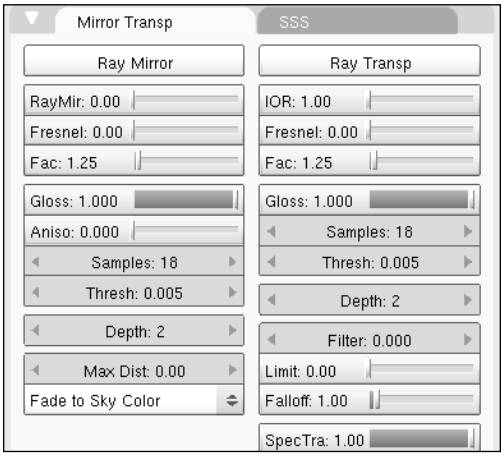
Here is a description of how each shader works:

- **CookTorr**: To control the way a material will generate highlights on the surface of objects, we can use the **CookTorr** shader. This is one of the simplest types of shaders, and it can generate small and well-defined highlights, great for plastic materials. There are two parameters available to control the intensity of the highlight (**Spec**) and how smooth the highlight border will be (**Hard**).
- **Phong**: This shader works in a very similar way to the **CookTorr**, but it can generate more fuzzy highlights. Materials that can use this type of Specular reflection include some kinds of rough stones and wood. The parameters of the **Phong** material are the same as **CookTorr**.
- **Blinn**: With this shader, we get an evolution from the **Phong** Specular, because it adds an index of refraction. In fact, the main visual difference between both shaders is that with **Blinn**, we get a highlight on surfaces at low angles. It is more realist than the **Phong** shader, and is used along with the **Oren-Nayar** to create concrete or wood.
- **WardIso**: Here we have a shader that is generally used to create metals and plastic, because it can produce hard edge highlights. In addition to the **Spec** parameter, we have **rms** that sets the size of the highlight.
- **Toon**: In the last Specular shader, we have the same properties as the **Toon** Diffuse shader, with hard edge highlights aimed at creating an animation cell effect.

Ray tracing

Materials that use ray tracing parameters can have an extra level of realism, because they can show reflections and transparency based on more sophisticated calculations. With these options, we can make more realistic materials, such as glass and mirrors.

To use these options, we must select the **Mirror Transp** menu. This menu holds all options to set up ray tracing materials:

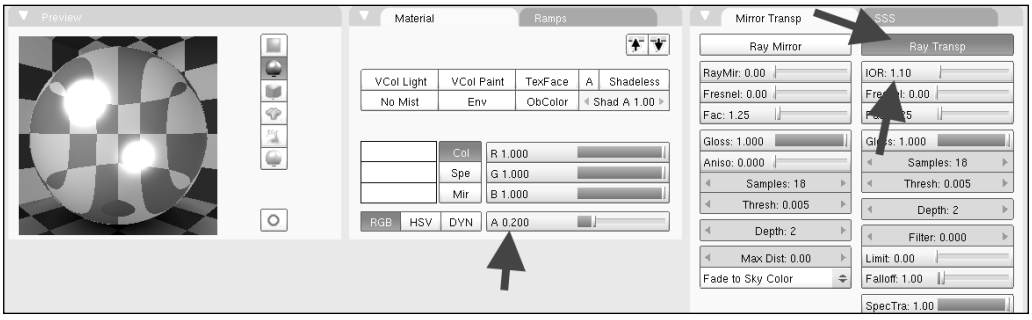


The difference between **Ray Mirror** and **Ray Transp** is that we can create reflections with the first and transparency with the second. The **Ray Mirror** is controlled by several settings, but it is with the **RayMir** that we set how reflective a surface will be. Both options share a few settings, similar to the **Fresnel** and **Fac**. The **Fresnel** controls what is named the **Fresnel** effect, which can set how much transparency or reflection a surface can have, based on the camera and face normals. With this effect, we will have areas of the surface with weak and strong ray tracing effects.

And, with the **Fac** setting, we can set how strong the transition between areas will become with weak or strong ray tracing effects.

Creating glass

To create a good glass or transparent material, we have to turn on the **Ray Transp** button. When this button is turned on, all materials will have a better transparency. Along with this option, we have to use the **Alpha**, to make the material transparent:

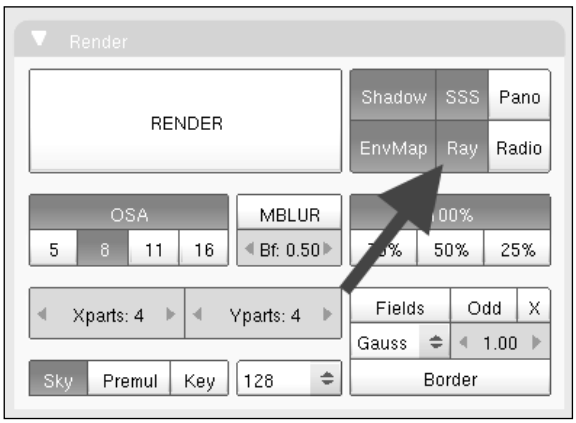


If the **Ray Transp** is turned on, and the **Alpha** is set to any value below one, the material will be transparent. Another aspect of transparency that we can set up is the **IOR**, which can determine how the object changes the ray light's trajectories.

The base value for some materials can be preconfigured, according to these values:

- Glass: 1.5
- Water: 1.33
- Diamond: 2.4
- Plastic: 1.46

If you want to render a material with this configuration, just make sure the **Ray** button is turned on in the rendering options:



Together, these options can produce almost any kind of transparent material; we just have to find the best options for any particular case.

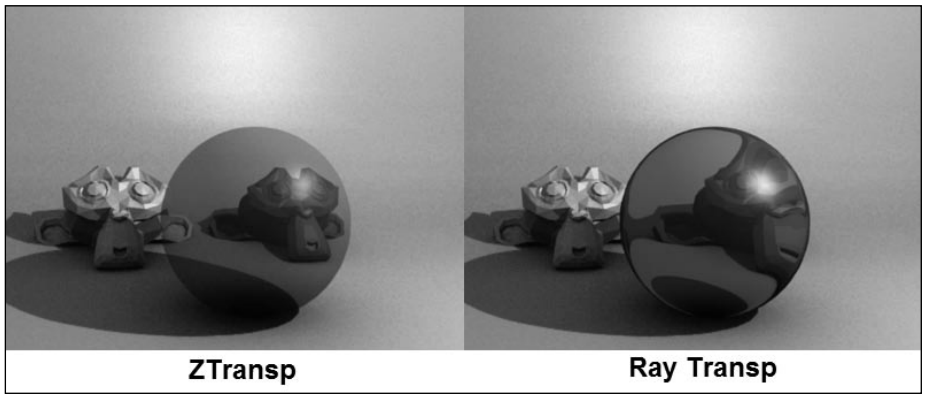
Turn off Ray for test renderings

If you want to speed up the rendering for test purposes, always turn off the **Ray** button. When you do that, all properties that use ray tracing will be off. So, the render time will drop. When you feel that the rendering is fine, turn it on again.

Simple glass

If the objective is to produce a simple glass, just decrease the **Alpha** value, and turn on the **ZTransp** button. This button is right below the text area where we can set up a name for the material.

The difference from the **Ray Mirror** option is that with **ZTransp**, we will see through an object based on the **Alpha** value. There is no calculation for color or anything else. It's simpler and faster, but not very realistic:

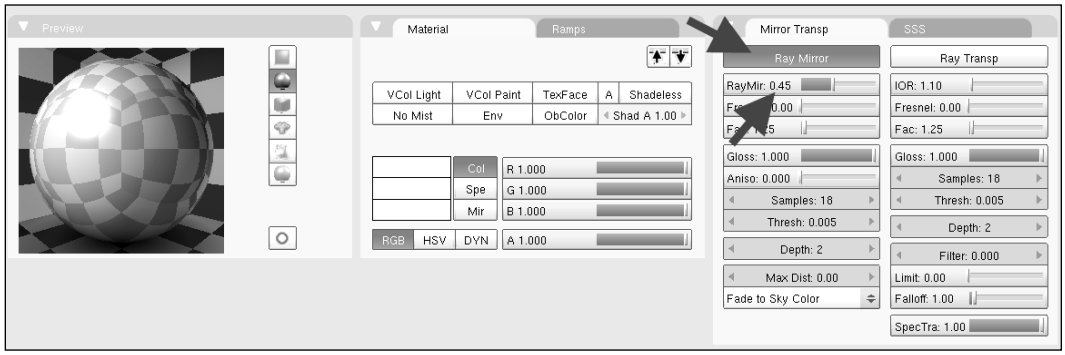


Mirrors and reflections

The other options for ray traced materials are reflections, which allow us to create mirrors and materials with reflections. To use reflections, we must turn on the **Ray Mirror** button, and set up the **RayMir** slider. Higher values will produce a strong reflection, and make the material act like a mirror.

To control the amount of reflection, we can use the **Fresnel** slider. It controls how reflective an object is, based on the angle at which we view the surface. With a **Fresnel** of **0**, we have a perfect mirror, but with higher values, the reflection will decrease, based on the angle that we view the surface. For instance, if we have a varnished wood floor, even being reflective, it's not a perfect reflection. Depending on the view angle, it's more or less reflective.

The **Fresnel** can be used for reflections and transparency as well, with the same objective—control the amount of ray trace by the viewing angle. Right next to **Fresnel** is the **Fac** option, which controls the blending amount for the **Fresnel** effect:



Almost all material has some level of reflection, especially materials such as glass, water, stones, and some kinds of wood. If you have a scene where these types of materials are used, applying some kind of reflection to them will increase the realism significantly.

Again, the best level of reflection will depend on a number of different factors in your scene. So, don't be afraid to test and play a bit with this slider.

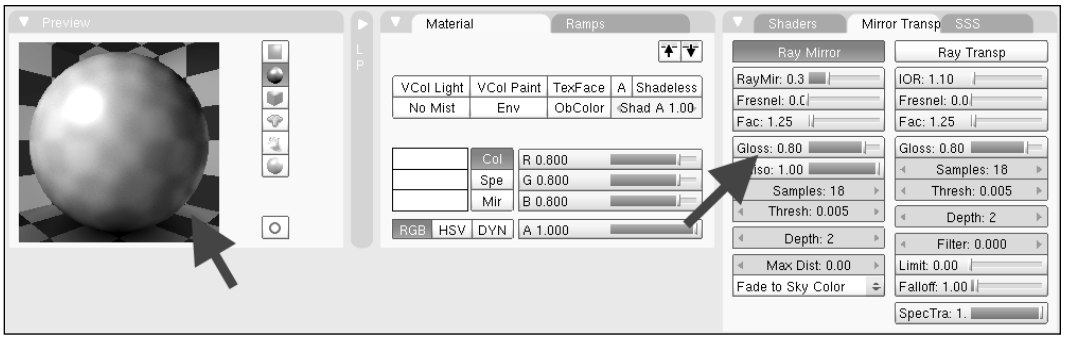
If you want to create mirrors, don't forget to change the material color to something near black.

Reflections and transparency

Beware that all these options can make our scene look more real, but can make the rendering process take a long time to finish. Use these options, but remember to not use them everywhere, unless you have very fast processors.

Glossy reflections

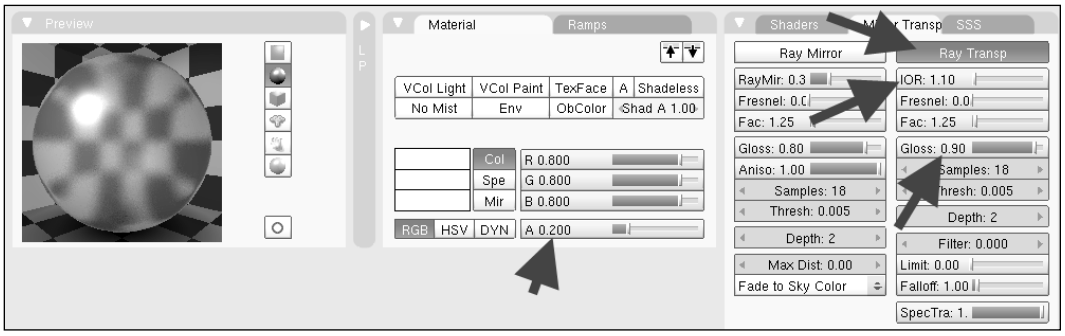
There is an option in the **Mirror Transp** menu that can add blur to all reflections created with ray tracing. If we use only the **Ray Mirror** button to represent surfaces like floors, the result will be a reflection created by something close to a glass material. To avoid this effect, we can use the **Gloss** parameter to control the shininess of the reflection. The default value for this parameter is **1**, but if we change it to any value below **1**, we will get a blurred reflection:



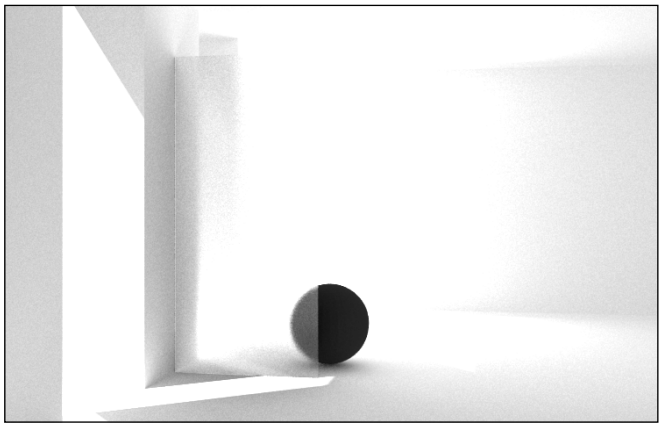
To change the quality of the reflection, change the **Samples** value below the **Gloss**. There are a lot of materials and surfaces that have this type of reflection, especially the ones representing wood and stone.

Glossy transparency and frosted glass

In this chapter, we have learned how to create a perfect and transparent glass material for our projects, but in addition to a perfect glass, we find another common glass material used for architecture named frosted glass. In Blender, we can create frosted glass using the **Gloss** value of the **Ray Transp** property. Just as with the glossy reflections, we just have to change the **Gloss** value in a glass material:



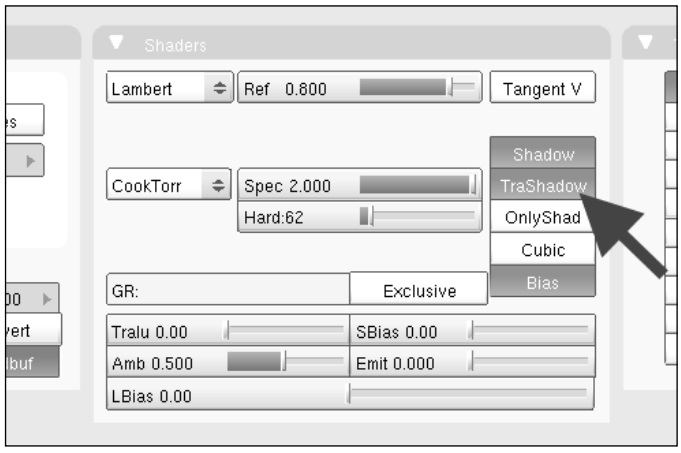
In the following image, we have an example of how a frosted glass material will look like in a scene:



Ray traced shadows

If we work with transparent materials, such as glass or plastic, we may want these objects to cast shadows based on their respective colors. For instance we may have a window, and the glass of this window is green. In the real world, all shadows that are cast by this glass won't be one hundred percent black, but will turn their color to green. We can achieve the same thing in Blender.

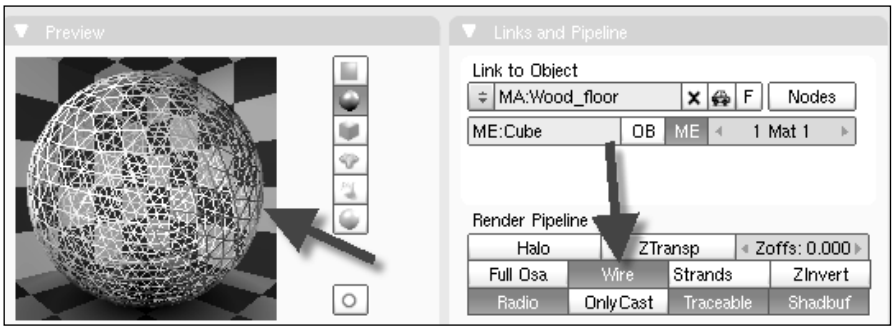
To use it, we must turn on the **TraShadow (Transparent Shadows)** parameter in the **Shaders** menu. This option must be turned on for the material which has to receive the shadow, and not the one that will cast it:



For instance, if a glass object is casting a shadow this shadow, may be projected on the floor. The floor material must have the **TraShadow** turned on to receive a shadow based on the material color and transparency level.

Wireframe materials

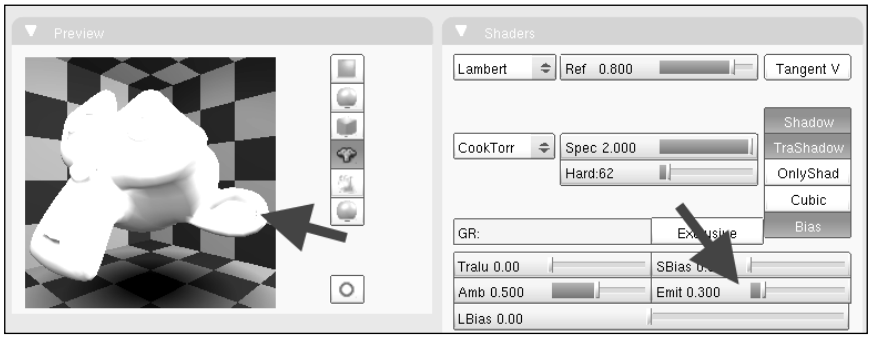
With an option named **Wire**, we can determine that some materials should be rendered only with the wire frame, if you want to show a more structured view of a 3D model. This could be a very interesting option; just select the material, and turn on the **Wire** option:



Self-illumination

There is a very interesting option in the materials setup, which allows us to set up a material to emit light. But it won't generate any kind of illumination for the scene. It will only make the material brighter.

To use it, we set up the **Emit** parameter in the **Shaders** menu. Make the adjustments to get the desired look, but remember that higher values will cause the material color to be very saturated:



Why can't this option generate light? The Blender internal render can't do global illumination. With Blender, light energy can only be cast by light objects, such as lamps. There is nothing wrong with it, and that's the way most 3D packages work. If you want to do global illumination, which is another way to work with 3D materials and lights, Blender works with some nice external renderers that can do global illumination, such as YafaRay, LuxRender, and Indigo.

In Chapter 12, we will take a detailed look on how YafaRay works to use global illumination in our scenes.

Summary

In this chapter, we have learned what materials are, and how they can give more realism to our scenes. Let's see what we have learned:

- How to create materials
- How to organize materials
- How to import materials between scenes
- Setting up a material color
- Determining how the material reacts to light
- Using ray tracing materials
- How to create transparent materials
- How to create glossy reflections
- How to create frosted glass
- How to create materials with reflections

With materials, we can create more realistic scenes. What we have to do now is practice, and apply these materials into a real scene. The next chapter will push materials to the next level with textures.

8
Textures

In the last chapter, we talked about materials and how they can increase the level of realism of our scenes. Well, with textures, we can take this realism to a higher level. With textures, the "magic" really happens!

Before we start to dig into textures, let me say that the biggest problem of working with them is actually finding or creating a good texture. That's why it's highly recommended that you start to create your own textures library as soon as possible. Textures are mostly image files, which represent some kind of surface, such as wood or stone, based on the photography. They work like wallpaper, which we can place on a surface or object. For instance, if we place an image of wood on a plane, it will give the impression that the plane is made of wood. That's the main principle of using textures; to make an object look like something in the real world. For some projects, we may need a special kind of texture, which won't be found in a common library, so we will have to take a picture ourselves or buy an image from someone.

But don't worry, because often we deal with common surfaces that have common textures as well.

Procedural textures vs. Non-procedural textures

Blender basically has two types of textures, which are procedural textures and bitmap textures. Each one has positive and negative points; which one is the best? It will depend on your project needs:

- **Procedural**: This kind of texture is generated by the software at rendering time, just like vector lines in software, such as Inkscape or Illustrator. This means that it won't depend of any type of image file. The best thing about this type of texture is that being resolution-independent, we can set the texture to be rendered at high resolutions with minimum loss of quality. The negative point of this kind of texture is that it's harder to get realistic textures with it. The advantage of using procedural textures is that because they are all based on algorithms, they don't depend on a fixed number of pixels.
- **Non-Procedural**: To use this kind of texture, we will need an image file, such as a JPEG, PNG, or TGA file. The good thing about these textures is that we can quickly achieve a very realistic look and feel with it. On the other hand, we must find the texture file before using it. And what's more, if you are creating a high-resolution render, the texture file size must be as well.

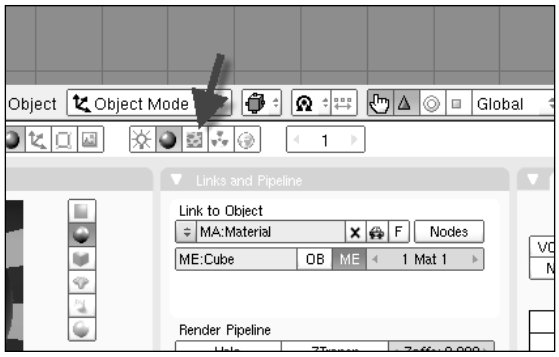
Texture library

Do you remember the way we organized materials? We can do the exact same thing with textures. Besides setting names and storing the Blender files to import and use again later, collecting bitmap textures is another important point. Even if you don't start right away, it's important to know where to look for textures. So, here is a small list of websites, which provide free texture downloads:

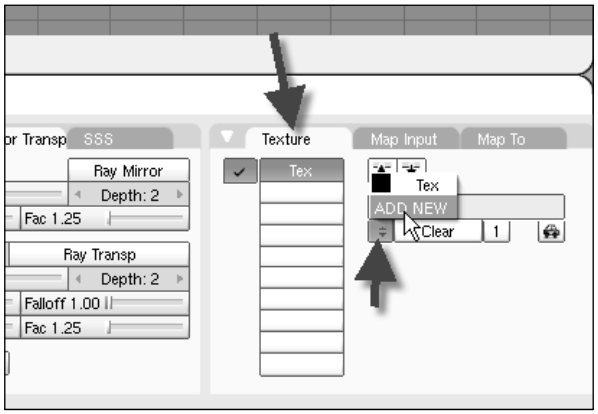
- `http://www.cgtextures.com`
- `http://blender-archi.tuxfamily.org/textures`

Applying textures

To use a texture, we must apply a material to an object, and then use the texture with this material. We always use the texture inside a material. For instance, to make a plane that simulates a marble floor, we have to use a texture, and set up how the surface will react to light to give the surface a proper look of marble. To do that, we use the **Texture** panel, which is located right next to the **Materials** button. We can use a keyboard shortcut to open this panel; just hit *F6* to open it:

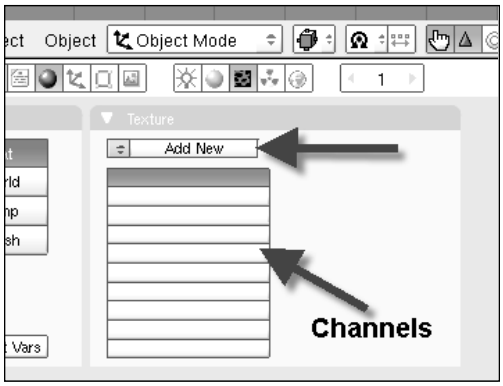


There is a way to add a texture in the **Material** panel also, with a menu named **Texture**:

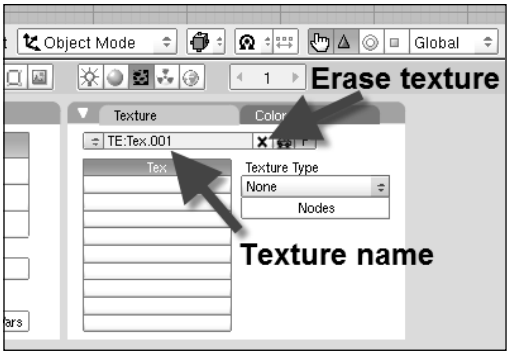


To get all the options, the best way to add a texture is with the **Texture** panel. In this panel, we will be able to see buttons, which represent the texture channels. Each one of these channels can hold a texture. The final texture will be a mix of all the channels. If we have a texture in channel 1 and another texture in channel 2, these textures will be blended and represented on the material.

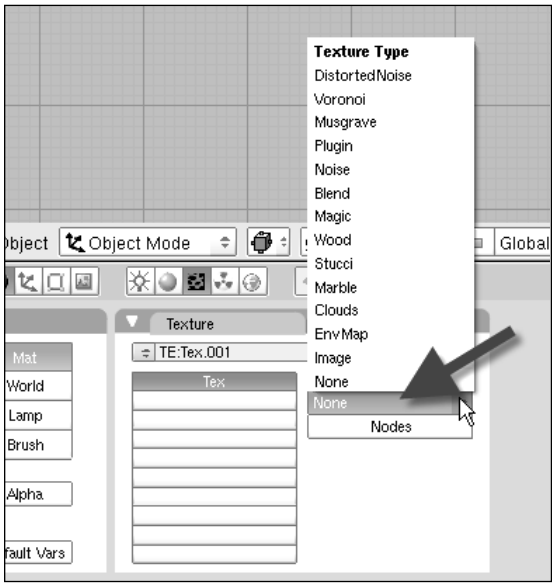
Before adding a new texture, we must select a channel by clicking on one of them. Usually the first channel will be selected, but if you want to use another one, just click on the channel. When the channel is selected, just click on the **Add New** button to add a new texture:



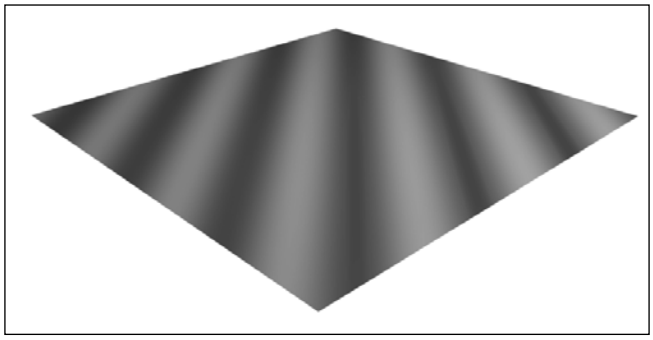
The texture controls are very similar to the materials controls. We can give a name to the texture at the top and erase the texture if we don't want it anymore. With the selector, we can choose a previously created texture also. Just click and select it:



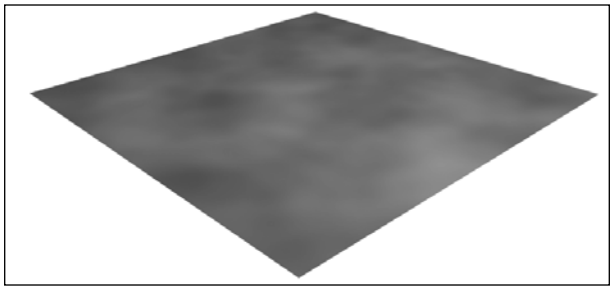
Now, here comes the fun part. With a texture added, we have to choose a texture type. To do that, we click on the **Texture Type** combo box:



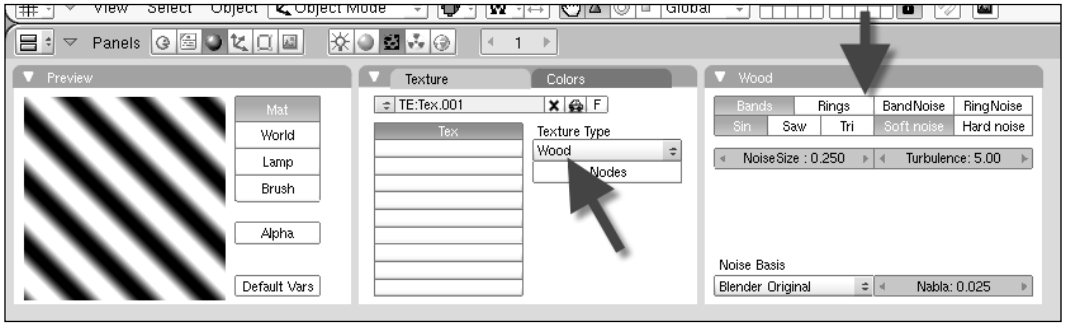
There are a lot of textures, but most of them are procedural textures, and we won't use them frequently. The only texture that isn't procedural is the **Image** type. We see an example of a procedural **Wood** texture in the following screenshot:



We can use textures such as **Clouds** and **Wood** to create effects and give surfaces a more complex look, or even create a grass texture with dirt on it. But most of the time, the texture type which we will be using will be the **Image** type:

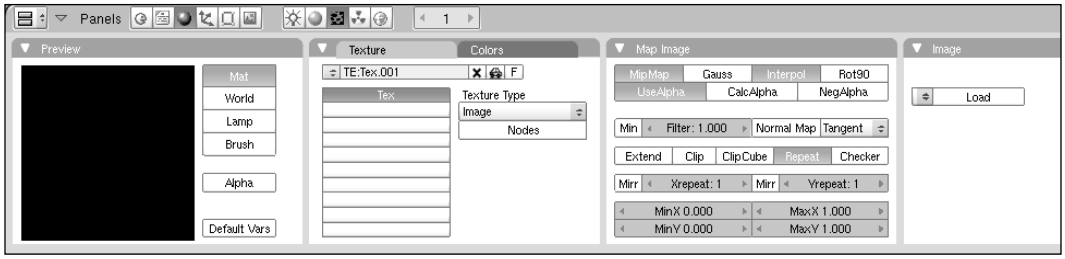


Each texture has it's own set of parameters to determine how it will look on the object. If we add a **Wood** texture, it will show the configuration parameters to the right:

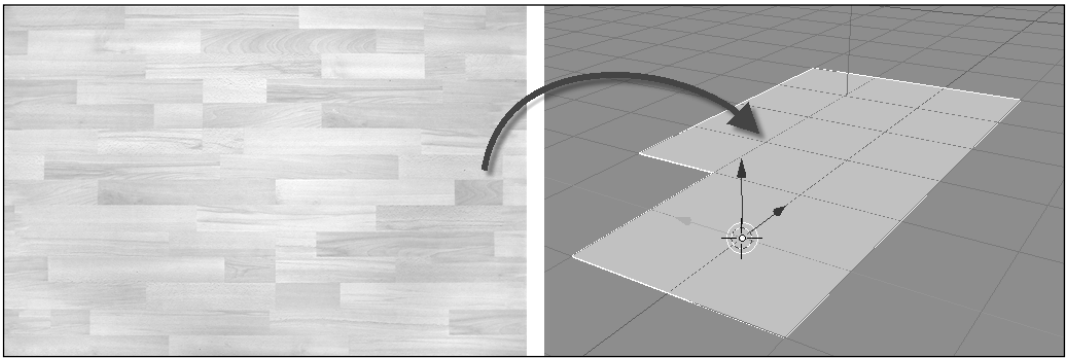


If we choose texture type as **Clouds**, the parameters shown on the right will be completely different.

With the **Image** texture type, it's not different. This kind of texture has its own type of setup. Following is the control panel:

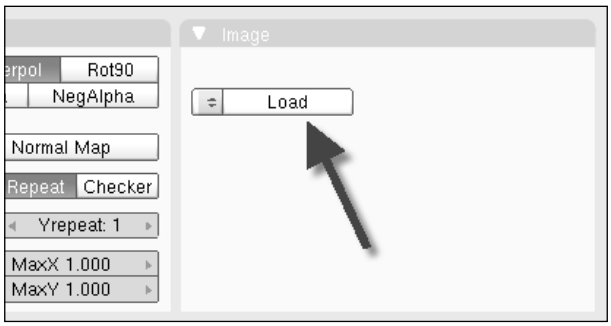


To show how to set up a texture, let's use an image file that represents a wood floor and a plane. We can apply the texture to this plane and set up how it's going to look, testing all parameters:



The first thing to do is to assign a material to the plane, and then add a texture to this material. We choose the **Image** option as texture type. Blender will show the configuration options for this kind of texture.

To apply the image as a texture to the plane, just click on the **Load** button, located in the **Image** menu. When we hit this button, we will be able to select the image file:



Locate the image file, and the texture will be applied. If we want to have more control on how this texture is organized and placed on the plane, we need to learn how the controls work. Every time you make any changes to the setup of a texture, these changes will be shown in the preview window. Use it to make the required changes.

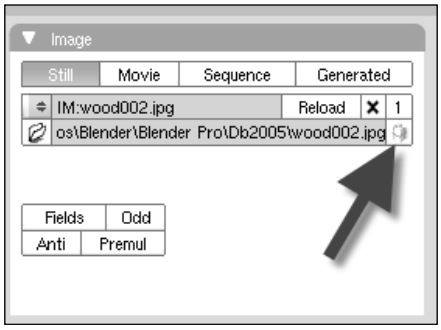
Here is a list of what some of the buttons can do for the texture:

- **UseAlpha***:* If the texture has an alpha channel, we have to press this button for Blender to calculate the channel. An image has an alpha channel when some kind of transparency is stored in the image. For instance, a PNG file with a transparent background has an alpha channel. We can use this to create a texture with a logo, for a bottle, or to add an image of a tree or person, to a plane.
- **Rot90***:* With this option, we can rotate the texture by 90 degrees.
- **Repeat***:* Every texture must be distributed on the object surface; repeating the texture in lines and columns is the default way to do that.
- **Extend***:* If this button is pressed, the texture will be adjusted to fit the entire object surface area.
- **Clip**: With this option, the texture will be cropped, and we will be able to show only a part of it. To adjust which parts of the texture will be displayed, use the **Min/Max X/Y** options.
- **Xrepeat / Yrepeat**: This option determines how many times a texture is repeated with the repeat option turned on.
- **Normal Map**: If the texture will be used to create Normal Maps, press this button. These are textures used to change the face normals of an object.
- **Still**: With this button selected, we will specify that the image used as texture is a still image. This option is marked by default.
- **Movie**: If you want to use a movie file as texture, press this button. This is very useful if we need to make something similar to a theater projection screen or a tv screen.
- **Sequence**: We can use a sequence of images as a texture too. Just press this button. It works the same way as with a movie file.

There are a few more parameters, such as the **Reload** button. If your texture file is updated outside of Blender, you must press this button to make Blender update the texture in your project. The **X** button can erase this texture; use it if you need to select another image file.

When we add a texture to any material, an external link is created to this file. This link can be absolute or relative. Suppose we add a texture named `wood.png`, which is located in the root of your primary hard disk, such as `C:\`. A link to this texture will be created like this—`c:\wood.png`. So every time you open this file, the software will look for that file at that exact place. This is an absolute link, but we can use a relative link as well. For instance, when we add a texture located in the same folder as our scene, a relative link will be created.

Every time we use an absolute link and we have to move the `.blend` file to another computer, the texture file must go with it. To imbue the image file with `.blend`, just press the icon for gift package:

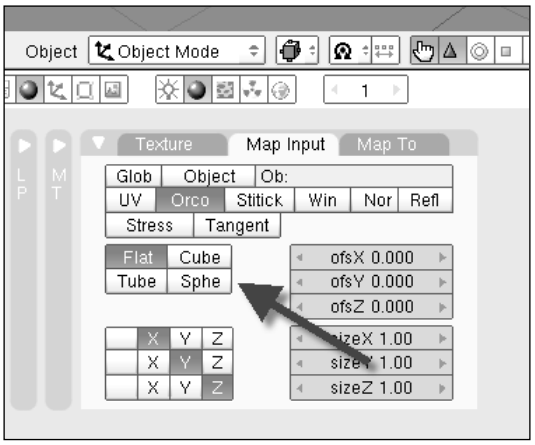


To save all the textures used in a scene, just access the **File** menu and use the **Pack Data** option. It will cause all the texture files to get embedded with the source `.blend` file.

Mapping

Every time we add a texture to any object, we must choose a mapping type to set up how the texture will be applied to the object. For instance, if we have a wall and apply a wood texture, it must be placed like wallpaper. But for cylindrical or spherical objects or even walls, we have to set up a way that makes the texture adapt to the topology of the surface, to avoid effects such as a stretched texture.

To set up this, we use the mapping options which are located in the **Map Input** menu:

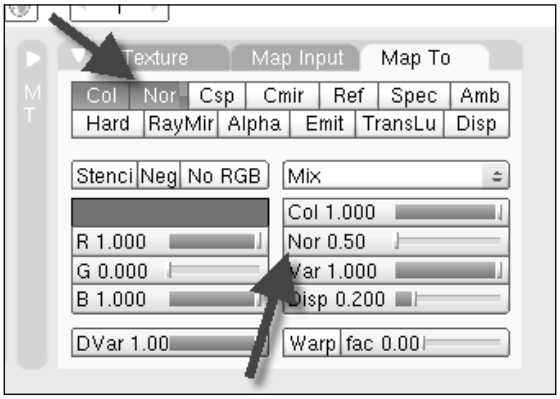


In this menu, we can choose between four basic mapping types which are **Cube**, **Sphere**, **Flat**, and **Tube**. If you have a wall, choose the option that matches the topology type with the model. In this case, the best choices are **Cube** or **Flat**.

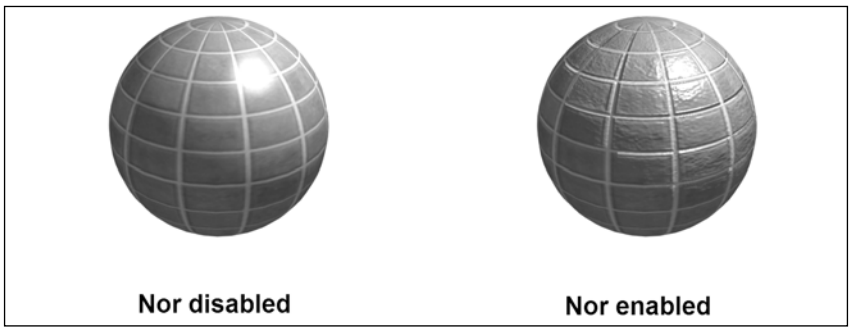
Another important option here is the **UV** button, which allows us to use another very powerful type of texturing, based on UV Mapping. We will talk about this technique later in this chapter.

Normal map

This is a special and useful type of texture that can change the normals of surfaces. If we have a floor and a texture of ceramic tiles, with this kind of map, the surface can be represented with all the little details of that tiling. It's almost as if we model the tiles. But everything is created just with a normal map:

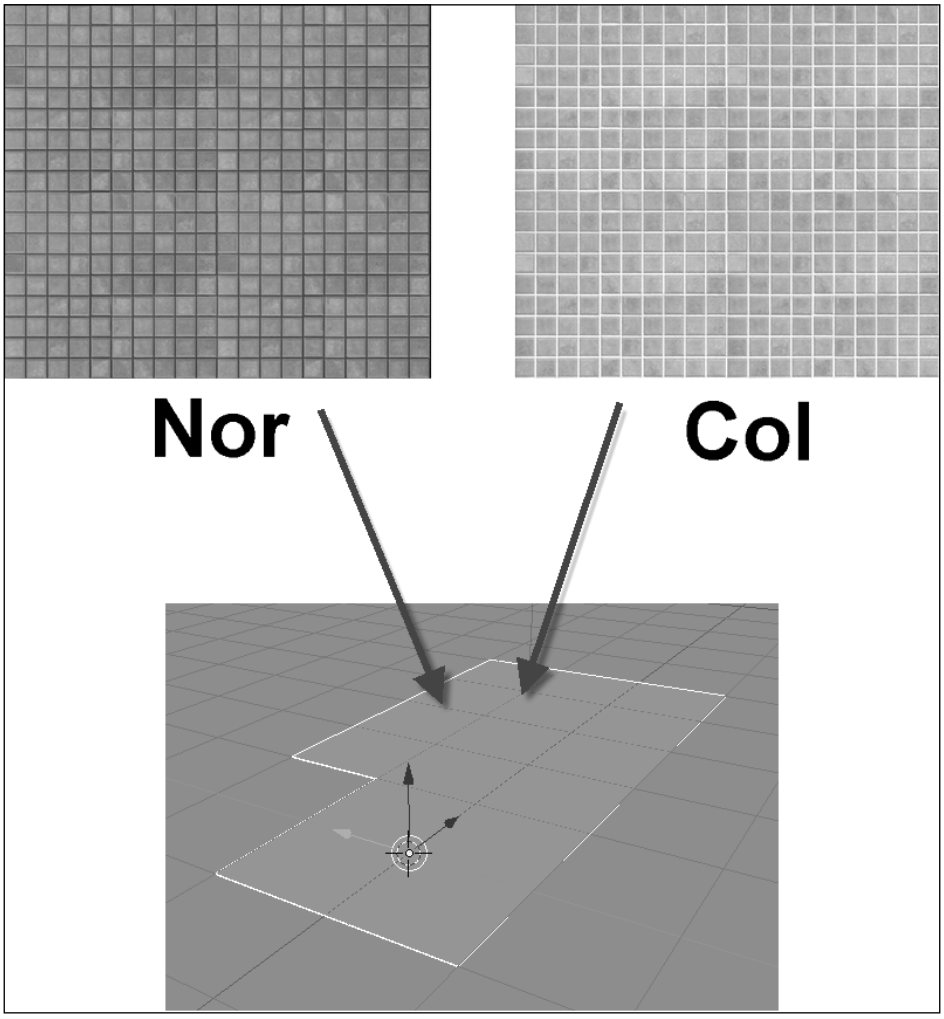


To use this kind of texture, we turn on the **Nor** button in the **Map To** menu. When this button is selected, we can set up the **Nor** slider to determine the intensity of the normal displacement. We see a comparison between a surface without and with the **Nor** option enabled in the next image:

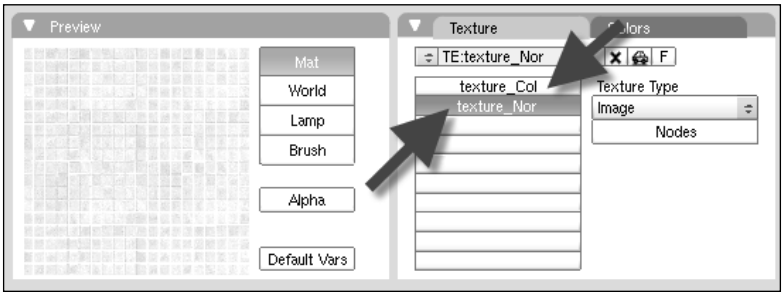


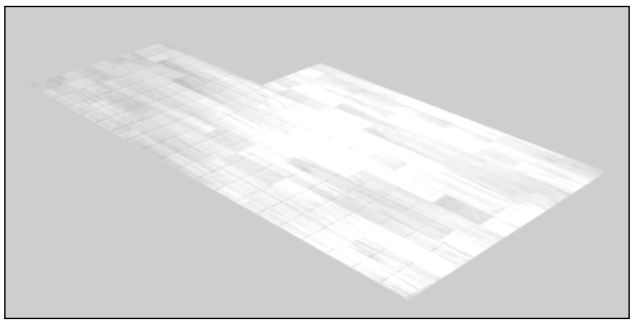
It works based on the pixel color of the texture. With white pixels, the normals are not affected, and with black pixels, the normals are fully translated. If you want to optimize the normal mapping, using a special texture for that is recommended. Some texture libraries even have these types of normal maps ready for use in projects.

Here is an example of how we can use them. We take a stone texture and a tiled texture with a white background and black lines:



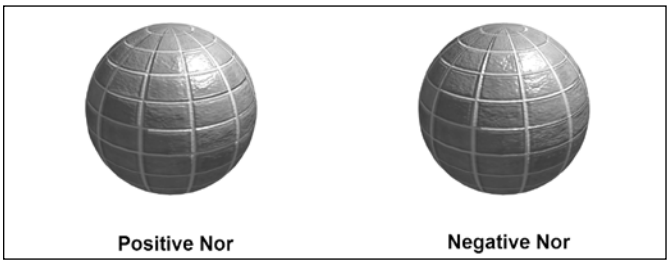
The stone texture is applied to the floor, and the tiled texture is used to create a tiling for the floor. The setup for that is really simple. Just apply the texture at a lower channel, and turn off the **Col** button for this channel. Turn on the **Nor** button, and this texture will affect only the normal's and not the material color. If you want, any image can be used as a normal map. Just set up the **Nor** intensity with the slider and check the render:





Turn on positive and turn on negative

Some of the buttons in the **Map To** menu can be turned on with positive and negative values. For instance, the **Nor** option can be turned on with one click. If we click on it again, the **Nor** text will turn yellow. This means that the **Nor** is inverted, with negative values. Some other buttons may present this same option:



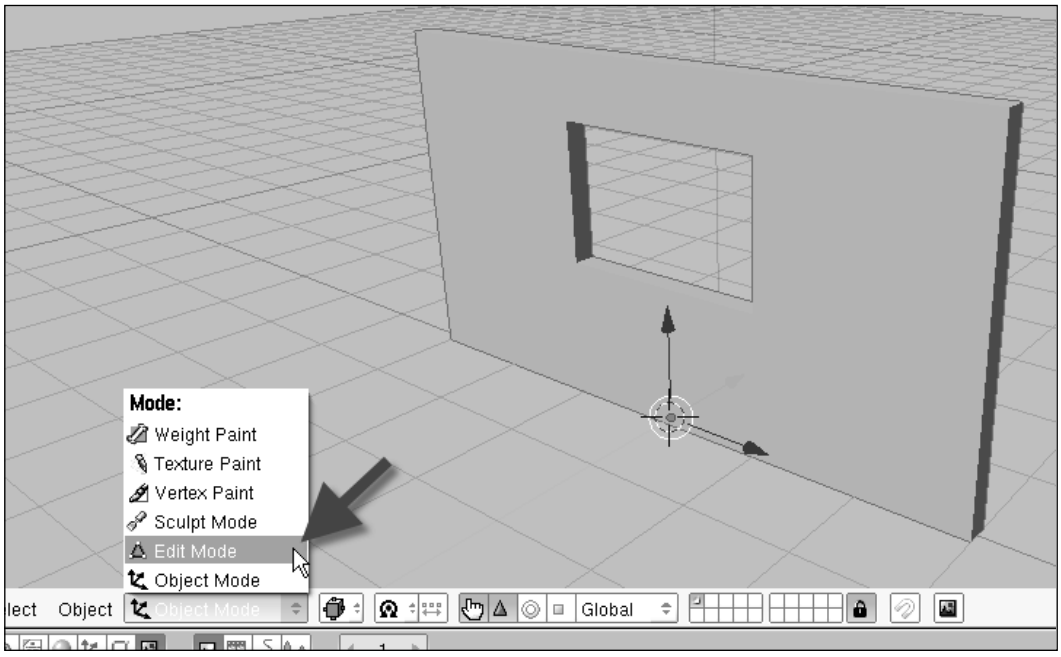
UV mapping

For some models, just placing an image on a surface is not enough. We have to take more control over all textures, and even create a more personalized texture for a model. With UV Mapping, we can create a texture image that fits exactly to all surfaces of a model and with the possibility to add details such as dirt and small imperfections to the texture image. Some of the painting of a texture can be done in Blender; we will take a look at that in the next chapter.

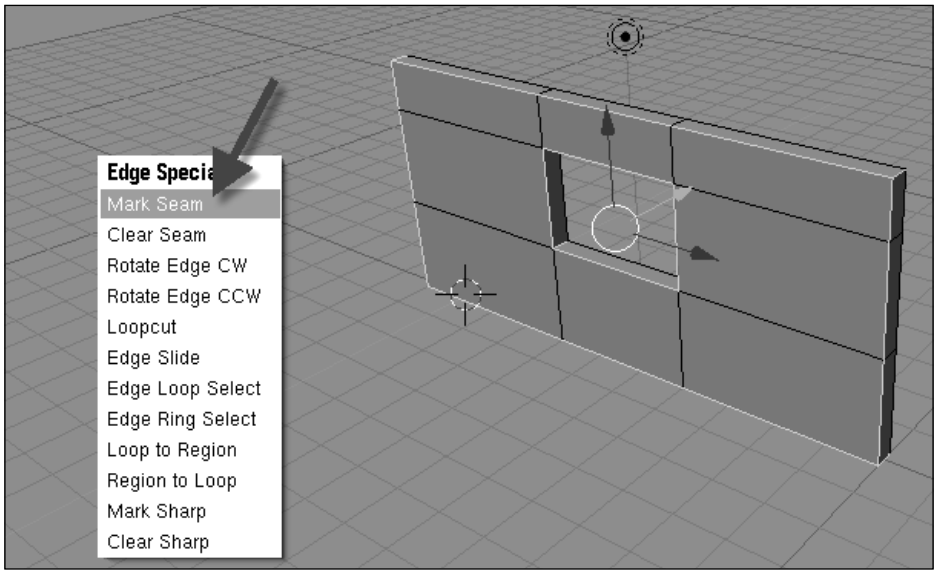
This kind of editing has to be done outside Blender, with painting software such as Gimp or Photoshop. Once this editing is done, we just have to apply the new texture again to the model.

What do we have to do to create a texture like this? The process for using this kind of texture is simple, but the task can demand a bit of editing. We must mark the model with some lines named seams.

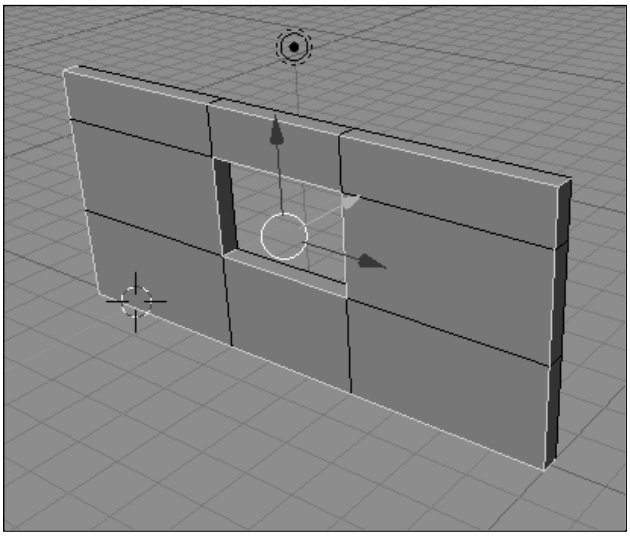
Let's see how it works with a wall. The first step is to select the model, and change the work mode to **Edit Mode**:



Change the selection mode to Edge, and select a few edges. Press the *Ctrl + E* shortcut, and choose the **Mark Seam** option:



With this, we will be marking the edges of the model that will be break during the unfold process. Which are the best edges to mark? Well, here we will have to use a bit of imagination:

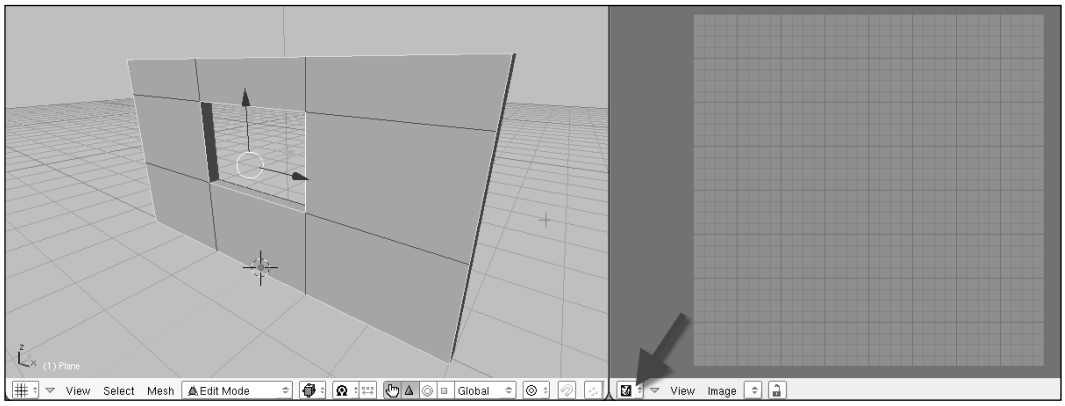


To choose the best edges, we must imagine the best places to mark and unfold the model. When the seams are marked, open a new window and change the window type to **UV/Image Editor**:

When this new window is opened, press the *U* button in the 3D View with all the objects selected. It will call the unwrap function, and create a flat layout with the unfolded model. Sometimes this operation requires a lot of testing and adjustment to produce a good result, but with the right seams, it will produce a nice flat image with all surfaces aligned and ready to be converted into a texture file.

Now, we have to export this layout as an image. To do that, we use a very good script named **Save UV Face Layout...** Just access the **UV** menu in the **UV/Image Editor** window, and choose **Scripts | Save UV Face Layout...**

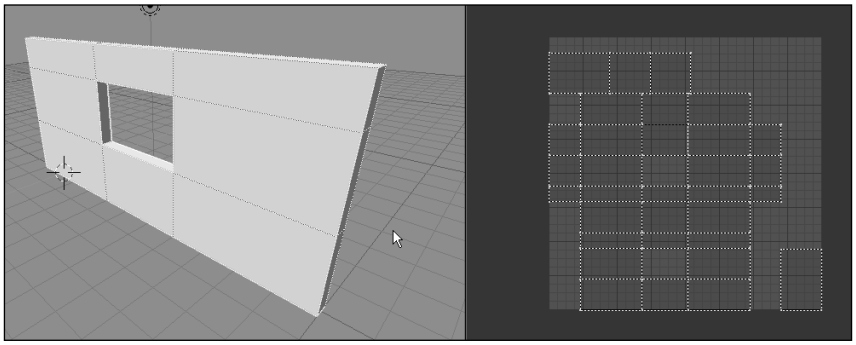
It will call a small menu, with the options for this script:



To set up the layout, we must change a few parameters, such as:

- **Size**: Here we can set up the size of the image. Always use a large value to create high-resolution textures. Use 1024, 2048, or higher.
- **Wire**: The layout will be saved with the wires from all faces. This option will determine the width of this wire.
- **Object**: With this button turned on, the object name will be used for the file name.
- **All Faces**: This button sets up the script to export all faces, and not only the selected ones.
- **Edit**: With this button turned on, right after exporting the layout, an image editing software package will be opened.

After editing the layout, we can apply the layout as a texture. But we must turn on the **UV** button in the **Map Input** menu. It will make the material look for a UV Mapping image to display:

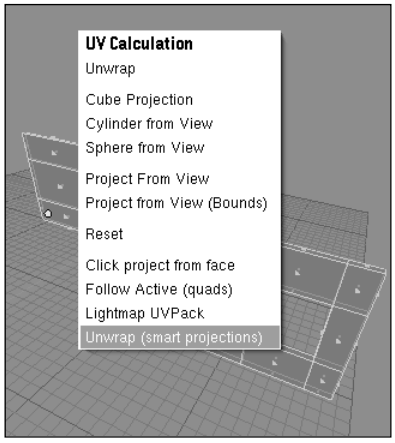


Now, all we have to do is press *F12* to render the image, and the texture will be applied to the object.

Unwrapping scripts

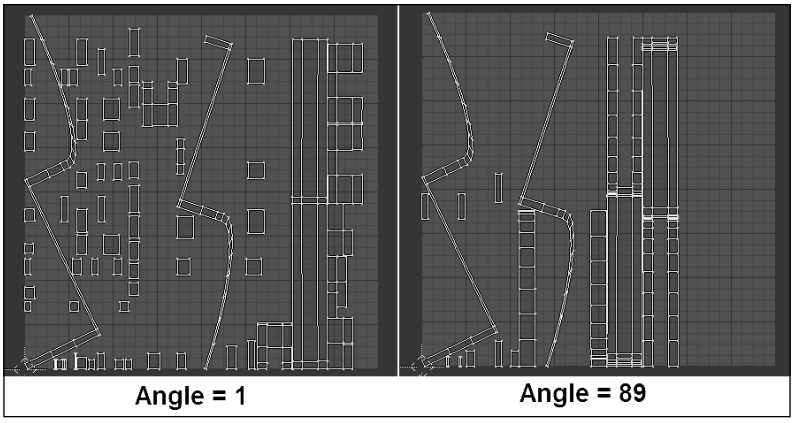
The operation of creating a UV Mapping can be very annoying for some people, because we have to imagine the unfolded model to mark the seams. To help with this task, there are a few scripts to create the seams and unfold the model automatically. One of them is created especially for architectural models.

The name of this script is **Unwrap (smart projections)**, and it's very easy to use. Just open a new window, and choose a **UV/Image Editor** as the window type. Then, in the 3D View, and in Edit mode, press the *U* key, and choose **Unwrap (smart projections)** in the **UV Calculation** menu:



The **Angle Limit** will determine how the faces will be unfolded. Bigger angles will generate layouts with grouped faces, and smaller angles will create layouts with more islands, which mean groups of faces generated by the unwrapping process. The **Selected Faces** button makes the script unfold only the selected faces, if turned on:

We can use the **Stretch to Bound** to make the layout fit the UV layout. And the **Bleed Margin** determines a limit for the UV layout to grow outside the limits of the window. And with the **Fill Quality**, we can set up the overall quality of the filling for the faces, if the **Fill Holes** button is pressed. We can see an example of how an angle set to **1** (minimum) and to **89** (maximum), can change the look of an unwrapped model in the following image:



Summary

In this chapter, we have learned how to work with textures to give our materials more realism. There are basically two types of textures, which are procedural and non-procedural textures. For us, the bitmap textures will be used most often, to allow us to create scenes with more realism.

In addition, we learned how to:

- Choose and organize textures
- Apply and setup a bitmap texture
- Map a texture around a model
- Use normal maps
- Create UV Layouts to create more complex textures

In the next chapter we will see in greater detail how to work with UV Mapping for architectural visualization.

9
UV Mapping

If you try to use textures on any kind of 3D model, you probably know that one of the biggest problems of textures, other than making them, is the adjustment process required to apply the image over the surfaces. Most people simply tile the textures, using a small image to fill up a big surface. It may even look faster, but the end result is very artificial and poor.

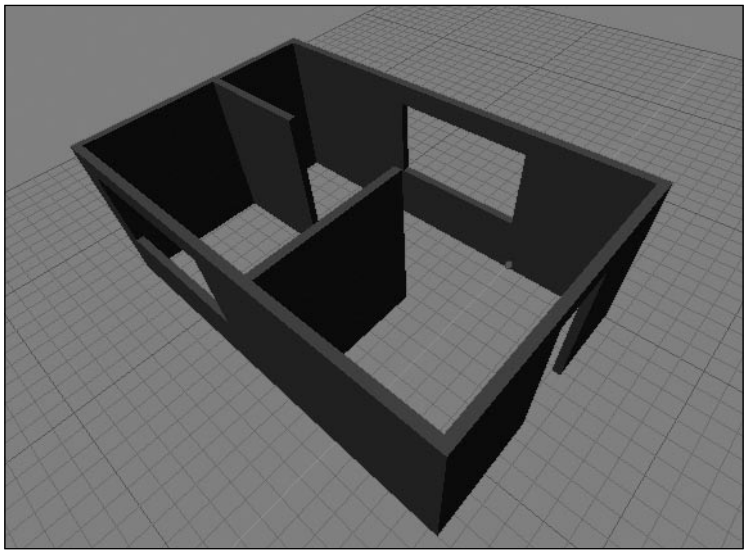
This occurs because the viewer will be able to see a lot of repeating patterns, making it very clear that this image was produced with a computer. Repeating patterns are associated with computers and artificiality.

How do we solve it? Avoiding the use of tiled textures is the first step. But, if we avoid them, we still have to fill up the surfaces of our models with textures. This is where UV mapping enters, letting us customize and have full control over the textures, without the use of tiles.

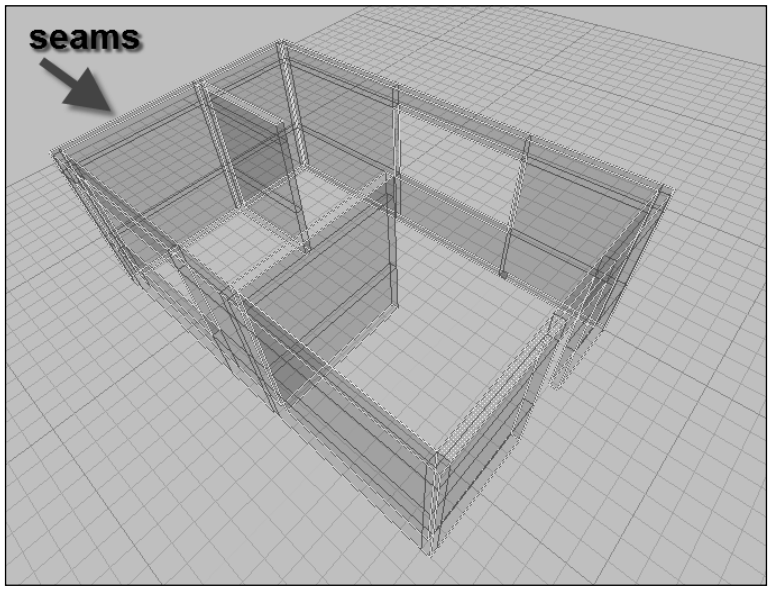
What is UV mapping?

With UV mapping we can tell the software to place textures at an exact location. The UV is a smaller set for X and Y coordinates. When we create a UV map, the computer has all the instructions to place the textures where the artist wants—on a face.

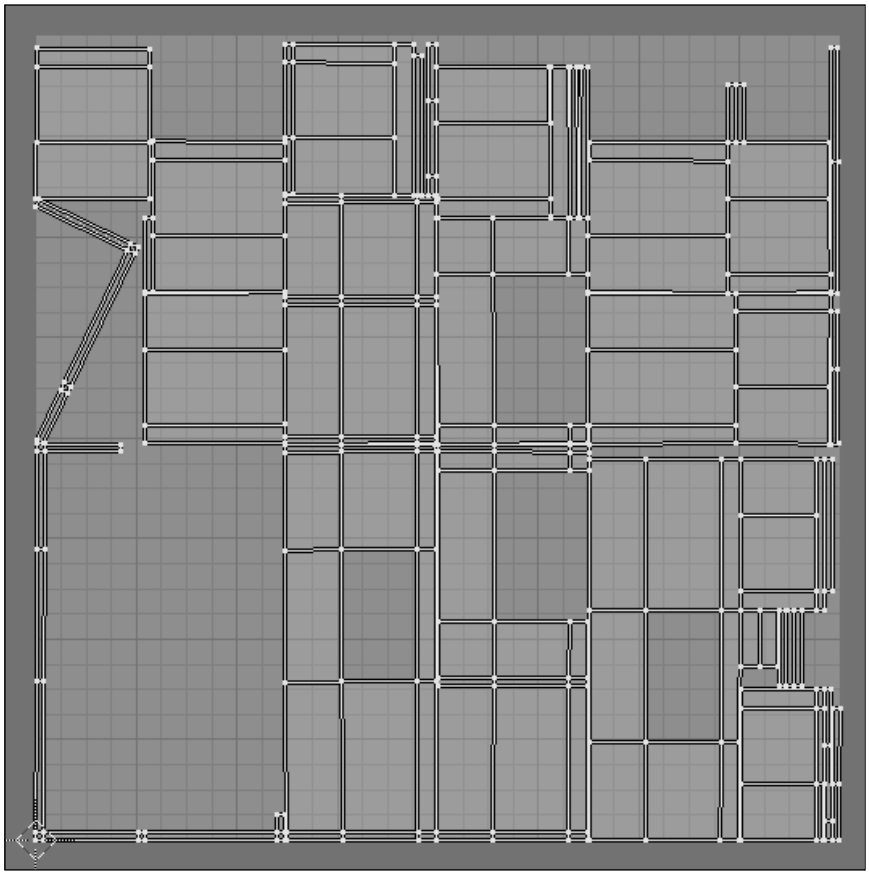
At the end of the process, all faces of the 3D model will be mapped with the texture, no matter what type or method of unwrap we use:



The process of UV mapping is very simple. One way to create it is by marking the edges of a model into several parts in order to unfold the 3D model as though it was made out of paper. This is named Unwrap. These cuts have to be made at the key edges of the model, otherwise it won't unfold correctly:



After that, we have to generate an image which contains all faces of the model. And with this image, we go to an image editing software such as Gimp or Photoshop to add and edit the texture map:



When the texture is placed, we save the image and take it back to Blender, where the same image can be aligned with the faces. Then, we will have all textures placed and adjusted correctly.

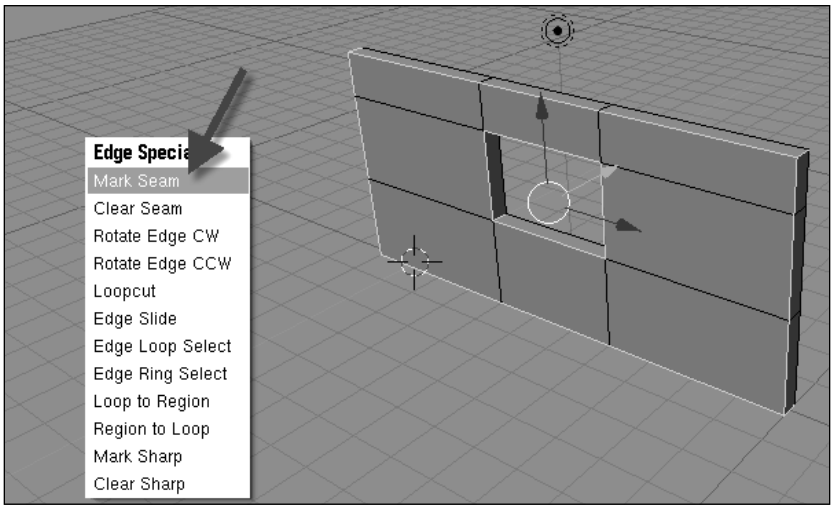
Why UV mapping?

If you are asking yourself right now—is this the best way to texture a model? Well, it's certainly not the easiest way, but it's the tool that gives us more control over textures and avoids the use of tiling. Work on the textures can be a key element in the success or failure of a project involving architectural visualization. Even if you manage to create an incredible illumination for the model, giving it a realistic look, a poorly textured wall or floor can give away that feeling.

Use the UV mapping technique only when you need full control over textures, or for the Interactive Animations created with the Blender Game Engine. Depending on the type of surface or project, a tiled texture may be the best choice.

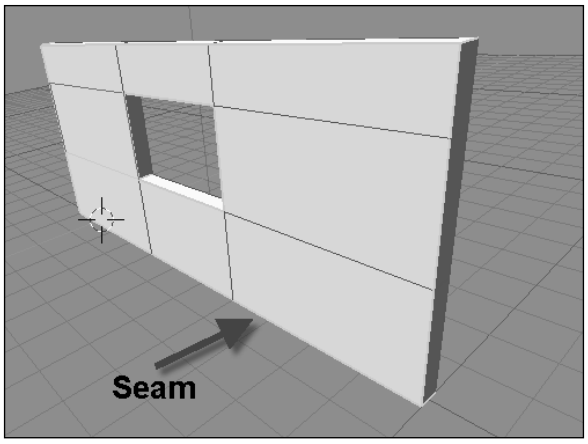
Marking the model

The first task we have to accomplish to use UV mapping is the slicing of the model. Here, we won't use tools such as the Knife or the Loop Cut, but a different tool which will allow us to mark a seam on the model. To do that, we use have to change the Select Mode to Edge, and press *Ctrl + E*. A menu that groups all edge operations will appear:



When we press this shortcut, a menu will appear giving us several options. One of these options is **Mark Seam**, which we will use to mark the edges of our model. These marks will be used to unfold the faces of the model.

In order to use this option, we have to select an edge, either by selecting two vertices or selecting the actual edge. Once you select it and activate **Mark Seam**, an orange line will mark the edge, making it easy to identify where the seams are located:



To remove a Seam, use the same shortcut, and choose **Clear Seam**. As you may be wondering, making only one seam won't help us much. Sometimes, a model, depending on the complexity, may require a lot of seams to be unwrapped before you start to mark them. Take some time to plan—which edges will require seams for your model?

It won't be a waste of time, if we consider that this process can actually save time later on revision and rework. So take a few minutes, before you start to visualize where the seams will be required to unfold your model.

Don't change the model!

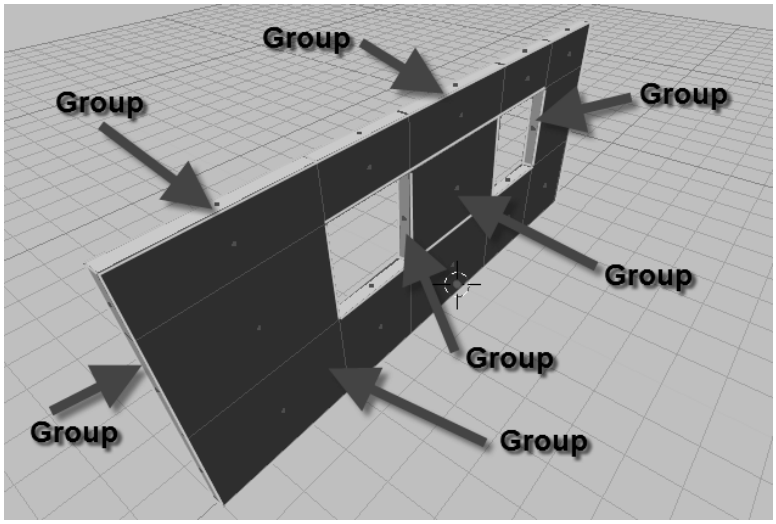
If you decide to use UV mapping, it's very important to use only the final version of your models. Any change on the topology of the model will require that all the work on the UV mapping has to be redone. So, don't start to mark seams if you know that your model may require changes. And if your client or project requires changes on the topology of the model, be ready to start all over again.

What makes a good seam?

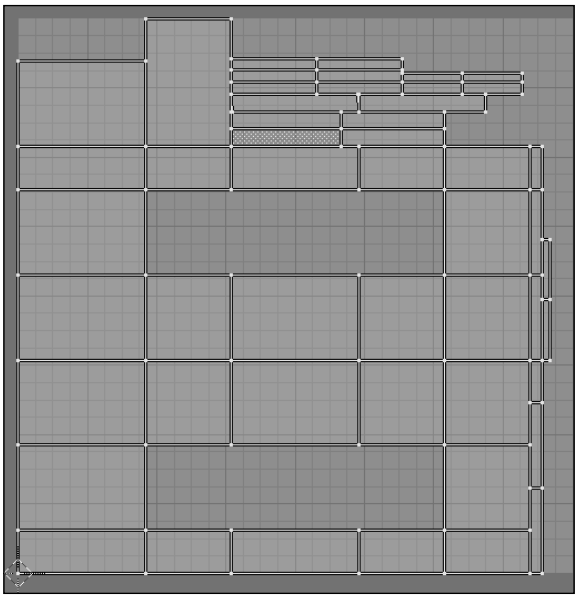
Creating the seam is not the only part of the task, but creating the seams in a way that makes your job easier is the real challenge. But what makes a good seam? Where are the best places to mark?

If you want to make your job easier at the texturing, try to leave all faces that get the same type of texture together. For instance, all faces of walls should be left together. This way it will be easier to adjust a brick texture, which should be applied to all of them.

The best way to organize these faces is with islands of faces. Before you organize the faces, try to identify which faces should be placed together as islands. Grouping them by type is a good start, such as putting faces for walls, floor, and ceiling as islands. At the end, we will have lots of groups marked with seams in orange, as shown in the following image:

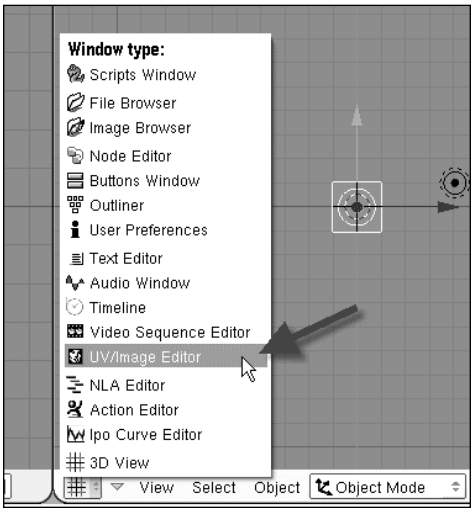


With that in mind, we can plan the best places to mark the seams that will result in those islands:



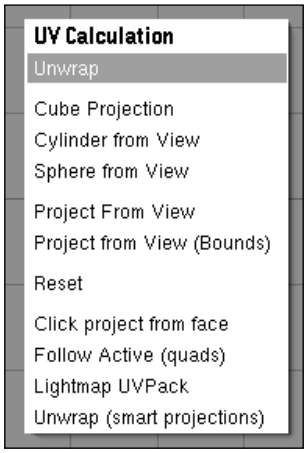
Unfold the model

When all the seams are marked, the next step is to unfold the model. Before we unfold the model, there is a special window type which will help us in the process of UV mapping. Create a new division in the Blender interface, and in the new window, choose the **UV/Image Editor** window:

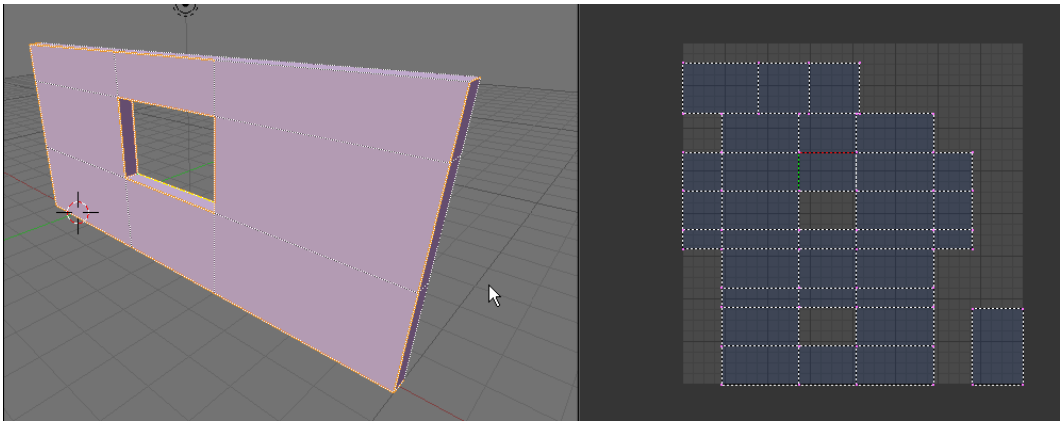


With this window, we will be able to see the unfolded model and handle all types of images for texturing. There is even a small editor and paint tool, which allows us to edit and paint the texture without the aid of an external image editor.

With this window opened and all seams marked, we press the *U* key in the 3D View to activate the UV tools:



After that, all we have to do to unfold the model is to choose the **Unwrap** option. It will then unfold the model and show the resulting image in the **UV/Image Editor** window:



Don't worry if things are not the way you want, at the first try. Make a small analysis of the faces, and try to identify areas which require more seams. Very often, 3D artists find themselves working on this kind of task over and over again, until they feel satisfied with the result.

Editing the unfolded model

Even with a lot of planning and work on the right edges to mark as seams, a few adjustments are required to make the unfolded mesh look the way we want. All we have to do here is use the basic editing commands of Blender, just as if we were modeling something.

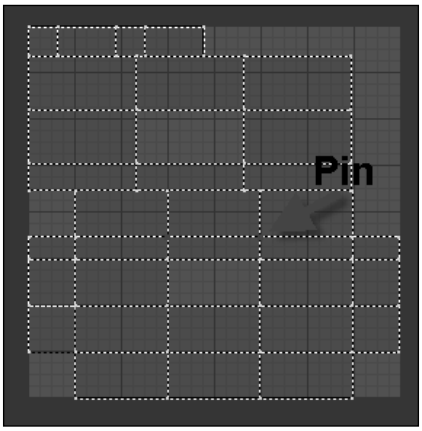
In the **UV/Image Editor** window, we can use the following shortcuts to select and adjust the unfolded mesh:

- *B* key: Box select the vertices
- *G* key: Move the selected vertices
- *S* key: Scale the selected vertices
- *R* key: Rotate the selected vertices
- *A* key: Select All/Deselect All vertices

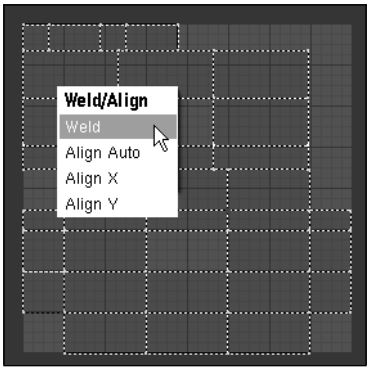
As you can see, most of the standard shortcuts for modeling work for the **UV/Image Editor** as well.

Along with this command, we can use a few other options to make our life easier. A very useful option is the Pin, which allow us to make a vertex stay fixed at its selected point. For instance, if you know that a few vertices are at their right positions, just select them, and press the *P* key to pin them. Now, if we try to move or edit them, they won't go anywhere.

Always use the pin to avoid any undesired changes in the unfolded mesh. To release them, use the *Alt* + *P* shortcut to unpin the vertices:



Besides the pin option, if we press the *W* key, a menu with a few additional options will appear. There we will find these options:

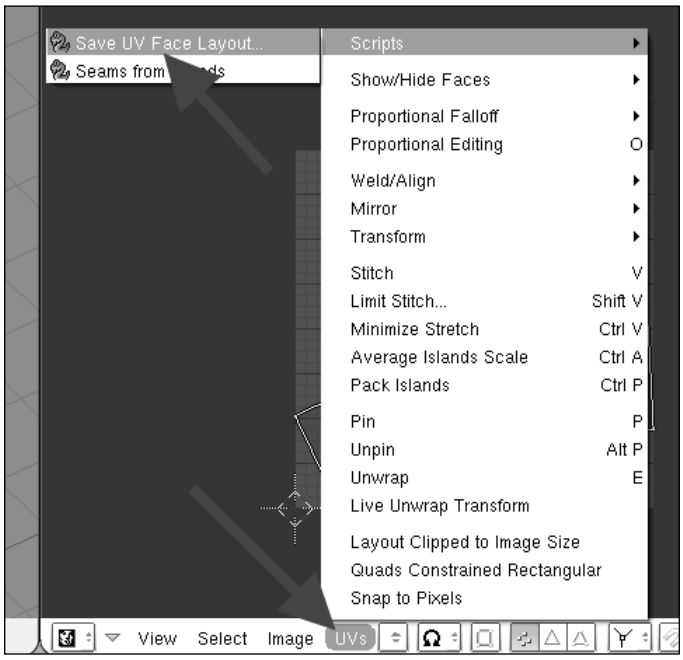


- **Weld**: With this option, if two or more vertices are selected, we can weld them to appear as only one vertex.
- **Align Auto**: Auto detects which is the best axis to align the selected objects.
- **Align X/Align Y**: Use these options to align a group of vertices on the X or Y axis.

Export the unfolded mesh

When the unfolded mesh is just the way you want, it's time to export it as an image. To do that, we use a very good script named **Save UV Face Layout...** Just access the **UV** menu in the UV/Image Editing window, and choose **Scripts | Save UV Face Layout...**

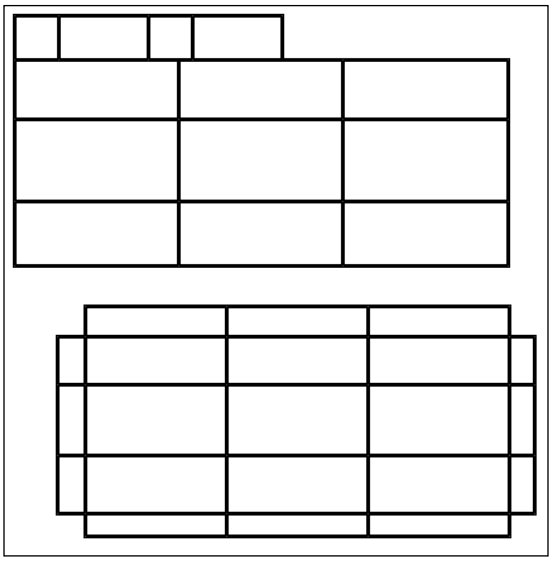
It will display a small menu, with the options for this script:

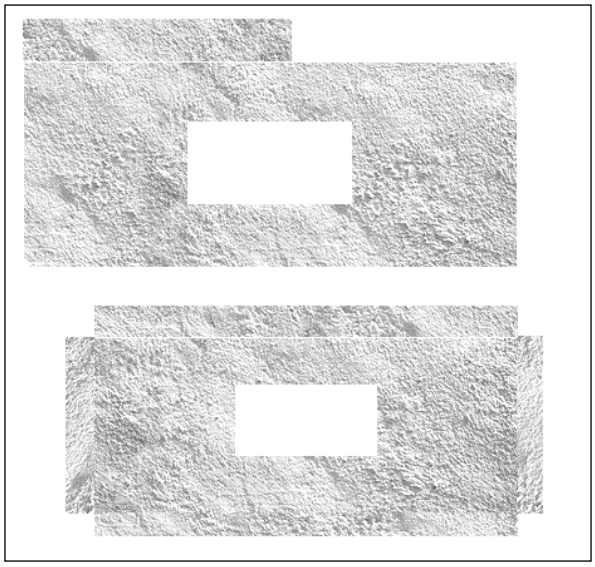


To set up the layout, we must change a few parameters, such as:

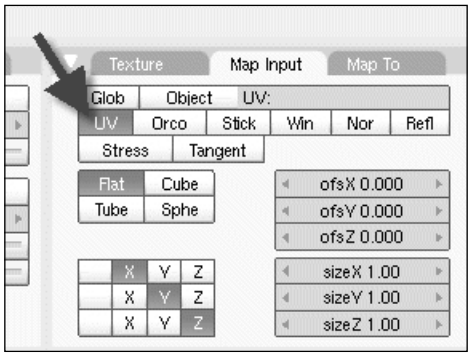
- **Size**: Here we can set up the size of the image. Always use a big value to create high-resolution textures. Here are a few sizes that are often used to export images: 512 x 512, 1024 x 1024, 2048 x 2048, and 4096 x 4096.
- **Wire**: The layout will be saved with the wires from all faces. This option will determine the width of this wire.
- **Object**: With this button turned on, the object name will be used for the file name.
- **All Faces**: This button sets up the script to export all faces, not just the selected ones.
- **Edit**: With this button turned on, right after exporting the layout, an image editing software package will be opened.

After saving the image, we have to open it in an image editor to add the textures. Try to organize everything in such a way that all the guidelines created by the wireframe of the mesh object are overlapped by the texture; otherwise, we will see these lines over the model:





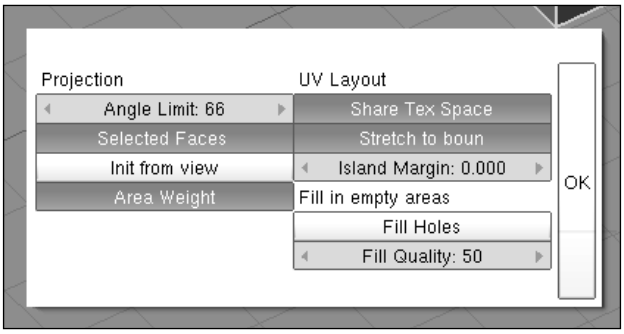
If we go back to Blender, we can apply the texture to the object and turn on the **UV** button in the **Map Input** menu:



Smart projections

The task of creating a UV mapping can be very annoying for some people because they have to imagine the unfolded model to mark the seams. To help in this task, there are a few tools to create the seams and unfold the model automatically. One of them is very useful for architectural models.

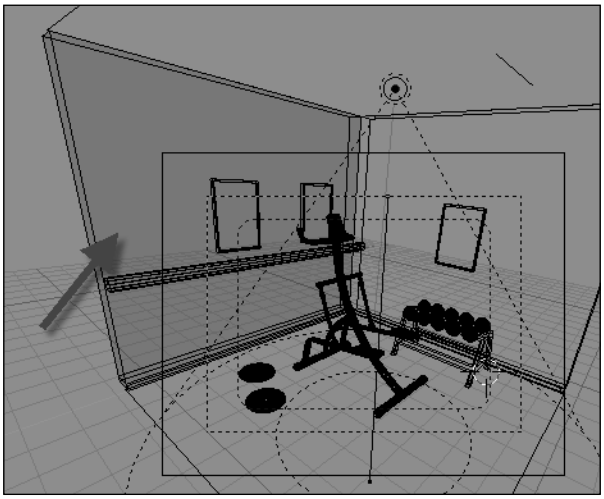
The name of this tool is Smart Projections, and it's very easy to use. Just hit the *U* key at the 3D View window, and choose the last option from the **UV Calculation** menu, which is **Unwrap (smart projections)**:



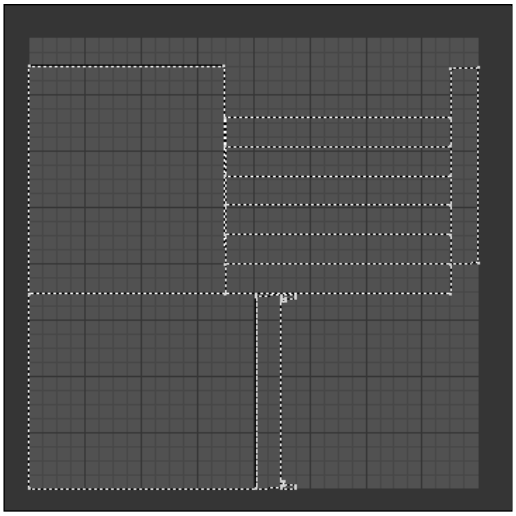
Angle Limit will determine how the faces will be unfolded. Bigger angles will generate layouts with grouped faces, and smaller angles will create layouts with more islands. The **Selected Faces** button makes the script unfold only the selected faces.

Stretch to Bound makes the layout fit the UV layout. **Island Margin** determines a limit for the UV layout to grow outside the limits of the window. And with **Fill Quality**, we can set up the overall quality of the filling for the faces, if the **Fill Holes** button is pressed.

See how it works, for the following model:



Select the model, and call the script. It will result in the following layout:



Summary

This chapter showed us a way to have more control over textures with the use of UV mapping. With it, we can create special images, which can be adjusted to fit a face of a model. With that, we can avoid the use of tiled textures and create more realistic textures.

Here is a list of what we have learned:

- What is UV mapping?
- Why we should use UV mapping?
- The tools required to create an unfolded mesh
- Exporting the unfolded mesh as an image
- Using smart projections to create unfolded meshes faster

10
Light Basics

This is the last thing about which we have to learn to make our scenes look real. Dealing with lights can really make the difference when we have a virtual scenario or architectural model. If you have ever worked with architecture, you must know that a big part of a good project is the behavior of light in various environments.

A scene with a poor light setup can be unpleasant in the real and virtual worlds. Or it can be even worse. Suppose we have a project that requires a good light, such as a living room. A setup that makes this scene look dark can jeopardize the chances of this project being sold.

Even being aware that a good light setup is the key to success with architectural visualization, why do we still see a lot of images with poor lighting? The problem with light is that every project requires an almost unique setup. This is the reason most people think that light setup is so hard.

If you have natural skills, such as painting or photography, setting up light may be a lot easier. What about the people who do not have this skill? Don't worry; even without this kind of skill, it's still possible to create incredible lighting.

To make that possible, we have to focus on three things. The first is understanding how light works with Blender or any other 3D suite. The second is to be patient! This is especially true if we work on some complex a scene, or even worse, a scene with no information at all about how the light should go in and out. This situation requires a lot of testing and prototyping for lights.

The third thing, and possibly the main point that can really save a project, is planning the light. As part of the project, you must know where all lights will be placed, and how these lights work. For instance, a daylight lamp can generate a unique kind of light. If we know that all lamps for our scene will be daylight lamps, our job will be a little easier.

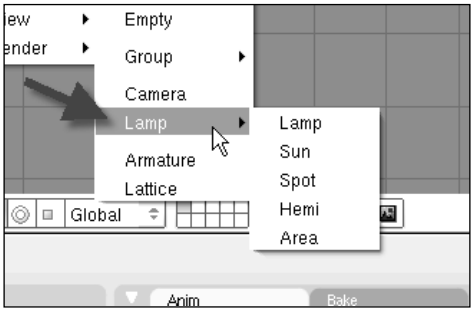
But, unfortunately, it's very hard to find a project that requires visualization and already has so many details already specified. Most of this setup part will be up to the visualization artist to decide where and how light should be set up for that particular scene.

This chapter covers how light works in Blender, and the different requirements and types of setup available.

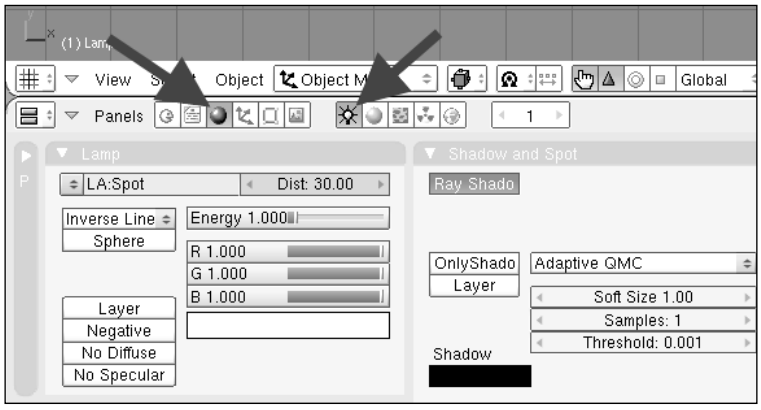
Lamps

Blender has five different types of light sources; every type has a particular effect over the scene lighting. Before anything else, we must know all those light types and understand the consequences of each one of them in our scenes.

The light sources in Blender are named lamps. To create a lamp, just press the *Space bar* and choose the **Lamp** option. Then, choose the desired lamp type in the menu:



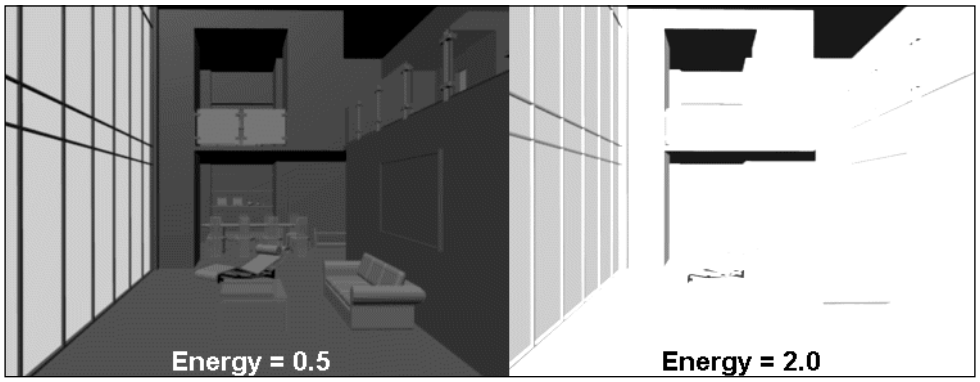
All lamps share common parameters, such as **Energy**, **Dist**, color, and others. These parameters are shown in the **Lamp** panel, which is placed just before to the **Materials** panel. If you already have a light created, just select **Lamp** to access this panel:



With the big buttons placed at the right of the preview window, we can easily switch between light types. For instance, if you have created a sun lamp, but realize that what your scene needs is a spot lamp, just select the lamp and press the **Spot** button.

Energy

Let's see how the common parameters work for each type of lamp. The first parameter is **Energy**. This slider determines how strong a light is. A good value here depends on the scene scale. For instance, for big environments, a value of five or higher may be the best fit:

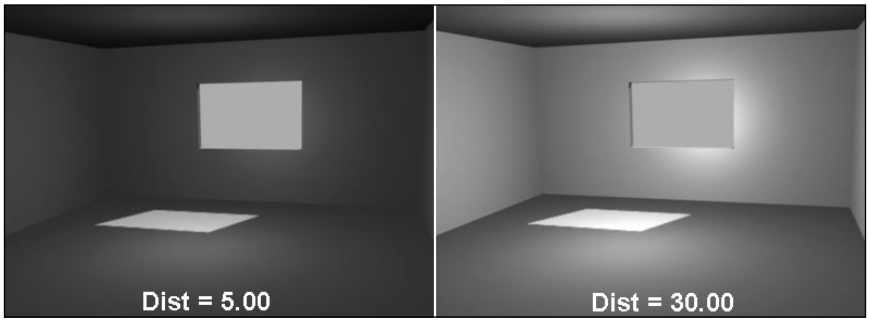


For small scenes, sometimes we have to set up the **Energy** to be under **0.5**, or close to zero! Remember, this is the value that we have to fine tune when we deal with distribution of light.

Distance

With the parameter named **Dist**, we can determine how far the light energy will travel. The use of distance can have a direct impact on the energy parameter. For instance, a lamp with a small energy and a long distance will be even weaker; the small amount of light energy will be spread throughout this distance. A long distance will spread more light energy, but with small distances, the energy will be concentrated. It means that even with a small energy setup, changing the distance to a small value will make the light look stronger.

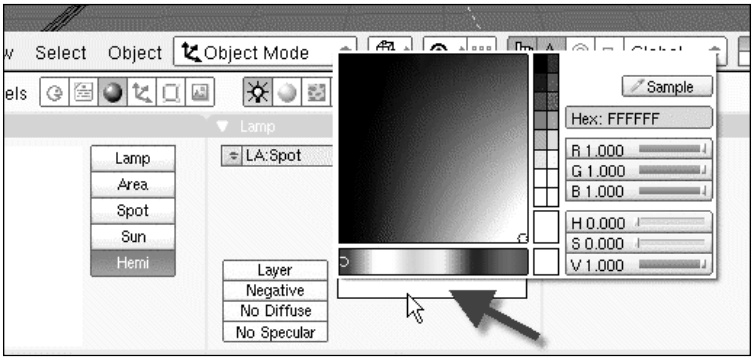
The next image shows a comparison between two light sources with different distances and the same amount of energy. In the image on the left, we have a **Dist** of **5** units and on the right a **Dist** of **30** units:



Color

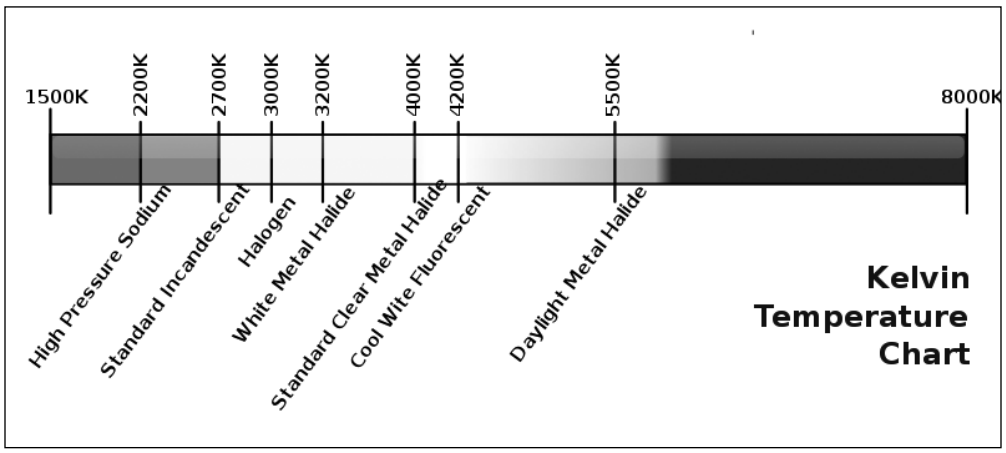
All lamps, from the simplest to the most complex, have a color. This color can be set with the color picker. Picking the right color can give the light a warm or cold sensation. If you don't know which color to pick, choose white. Most lamps emit light with a white color, so the chances of getting the right feeling will increase. Here are a few examples of light colors:

- **Warm color**: Red, Orange, Yellow
- **Cold color**: Blue, Green, Purple
- **Neutral color**: Gray, White, Black



Kelvin Chart

If you don't have any idea about the color of the lights, try to use a very good reference to pick the right colors. There is a chart named the Kelvin Chart, which presents a scale of colors with a representation for each light and environment type. For instance, to choose the color for **Cool White Fluorescent** or **Halogen** lamps is easier with this chart:

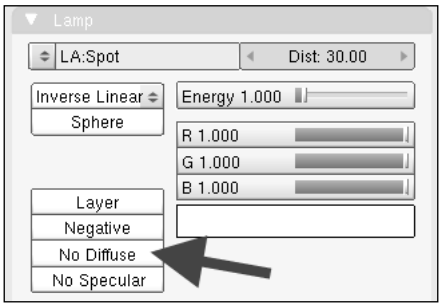


Controlling light

In addition to these options, we can choose which objects get illuminated by our lights, and also make other choices. We can do that with the buttons placed in the lower-left corner. Let's see what we can do with them:

- **Layer**: If this button is turned on, the lamp will illuminate only objects which are on the same layer as the lamp. It can be useful to create lamps that only generate light for a particular piece of furniture or a set of walls. Just place all of them on the same layer and press this button.
- **Negative**: With this option, we can make a lamp cast negative energy. This means to remove light energy from the scene. It's great to generate contact shadows or balance a very strong spot of light.
- **No Diffuse**: Turns off the diffuse option.

- **No Specular**: Turns off the specular option.



Naming lamps

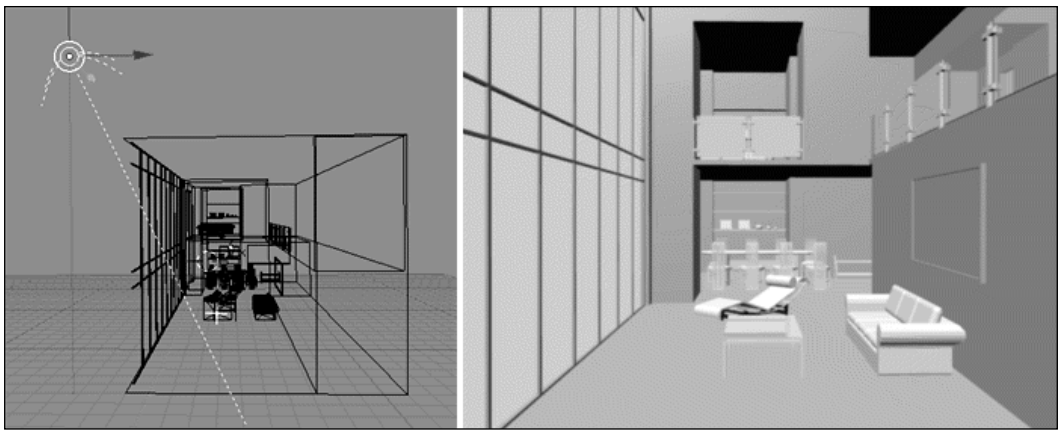
As with materials and other objects, we should give lamps unique, meaningful names. Always set a name before adjusting the lamp. It will make things easier when you need to find or search for a particular lamp.

In Blender, we have several types of lamps; each one of them can be used to generate a unique effect on the lighting of any project:

Hemi

The **Hemi** lamp is the simplest light available in Blender, because it doesn't generate any shadow. This makes the controls very simple. In fact, the only controls available are the basics, which we have already described.

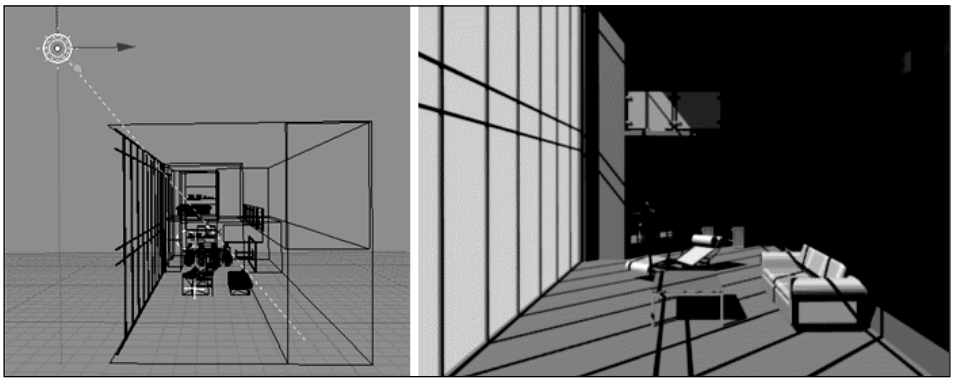
When we use this kind of lamp, it generates an illumination that simulates a cloudy sky. This is a diffuse illumination with no shadows and a well-spread illumination. If you want to add light energy to a scene with a constant grade, this is the right option:



Sun

As the name says, this light tries to simulate direct sunlight. After placing the lamp, we can control the direction of the light. Just select the lamp, and rotate the object and aim the line to the location that must receive light.

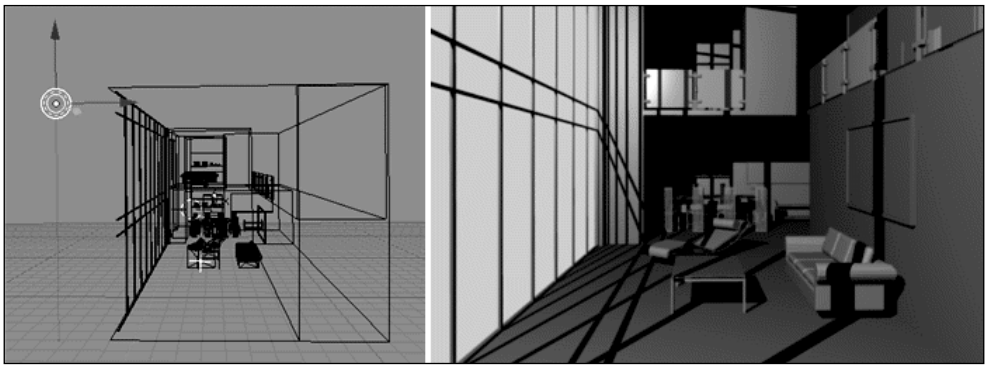
There are only two extra options for this lamp, which are the ray traced shadows button (**Ray Shadow**), and the option to turn on the creation of shadows (**Only Shadow**) only:



For external scenes, this is the best choice to simulate sunlight, even when we use an external renderer. We use it for creating sunlight for interior scenes as well. It's very easy to set up, so use it frequently.

Lamp

This is a lamp that works like a point of light, which can cast light energy from its center to all sides. There aren't any options for this lamp. The only interesting thing about it is related to its distance. We can control the distance with a radius, so it doesn't work like a directional light. Just place it at a point, and it will illuminate all around it:

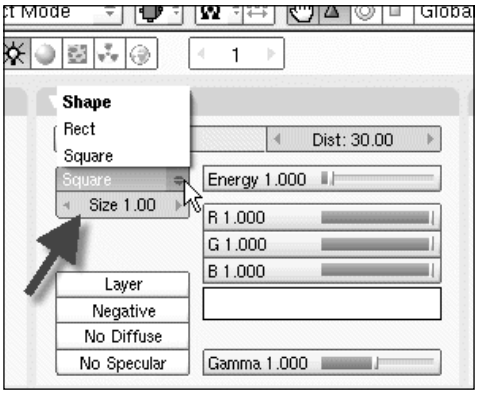


The best application for this lamp is dark corners that need extra light. Because this is the only lamp which is not directional in Blender, every time we need a light that illuminates all around its position, choose the **Lamp** type.

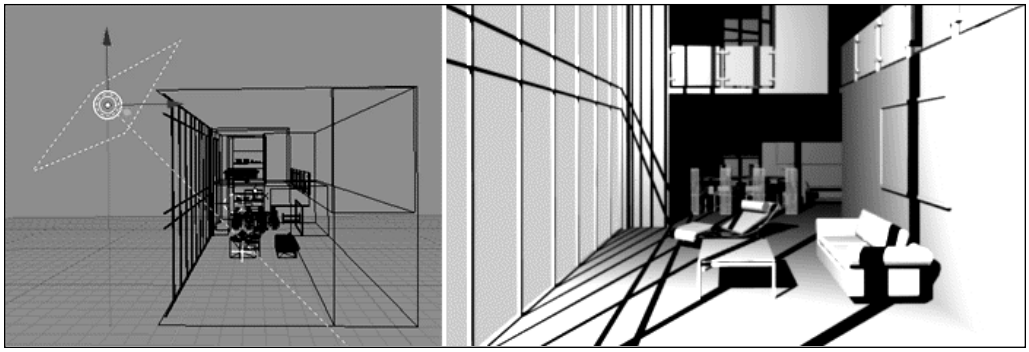
Area

This type of lamp consists of a plane that can generate light. It's perfect to fit at a window or door where light should go through. It is a directional light, so we can change the target, select the lamp, and rotate it with the *R* key. Place the line that represents the target where you want the light energy to be cast.

When we choose this type of lamp, there are some options related to **Area** lamps only. The first one can specify the **Size** for this lamp. To change the **Size**, we use the selector pointed to in the following image:



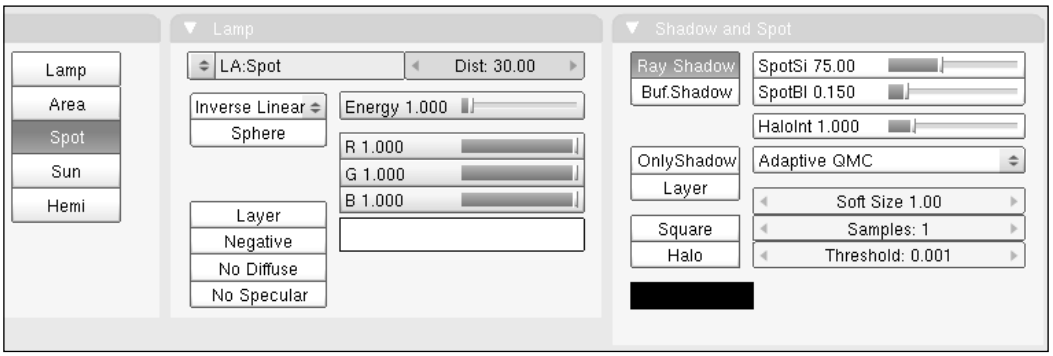
In addition to the **Size**, we can change the quality of the shadow. This is set up using the **Samples** option. With more samples, we have better shadows, but greater render time:



Spot

Just as the name says, this lamp creates a spot of light. This is perfect to represent a wide range of light sources with conic shapes. And, this is a directional light source, which allows us to point to the objects or areas that may receive light.

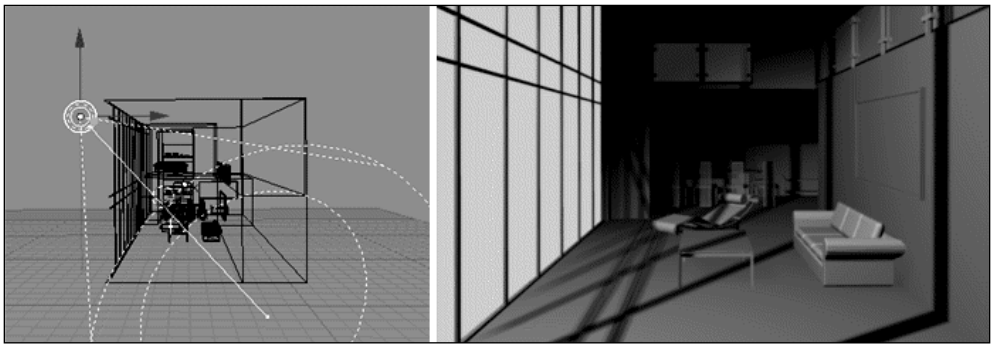
Compared with the other light types, **spot** is by far the most complex lamp type in Blender. We have a variety of options for shadows and to generate volumetric lights:



Let's see how we can use each option to interact with our scenes and produce better lights:

- **Ray Shadow**: With this type of shadow, we can create a very defined and accurate shadow. The shadow will have a well-defined border, which is perfect to simulate artificial light sources in open spaces. This happens because in open spaces we don't have surfaces to bounce the light rays and create a fuzzy shadow. To use this type of shadow, just press this button.

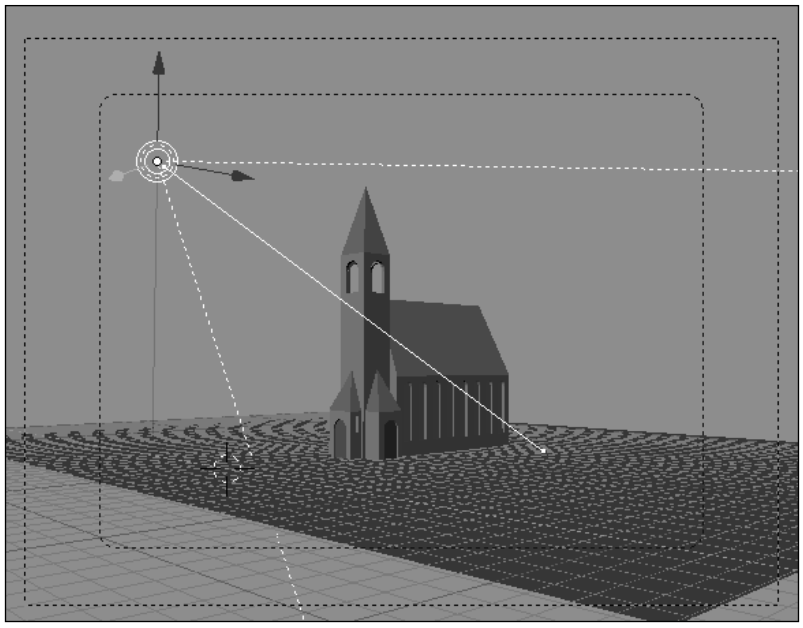
- **Buf. Shadow**: The other option to generate shadows is with a Shadow Buffer. This type of shadow is not as accurate as the **Ray Shadow**, but is much faster to render. We can set up the quality level of this shadow with the **ShadowBufferSize** option. A bigger value will result in a better shadow. Most times, we use the shadow buffer to generate shadows, because with this it is possible to create fuzzy shadows, which are generated in closed environments by bouncing light rays at walls and other surfaces:



- **Square**: The spot lamp has a circular projection shape, which can be switched to a square projection with this option.
- **Halo**: With this option, we can create volumetric light to give a more dramatic look to any environment. For instance, if we have a church model and want to create a light that comes into a big window and want to show the light rays coming through the window, this is the best option.

Volumetric shadows

As we said, the best option to generate volumetric shadows is the **Halo** button. Because it has a lot of options to set up, let's see how it works with a church model. The first thing to do is place a Spot Lamp pointing to the interior of the scene. Set up the camera view to visualize the window inside the environment:

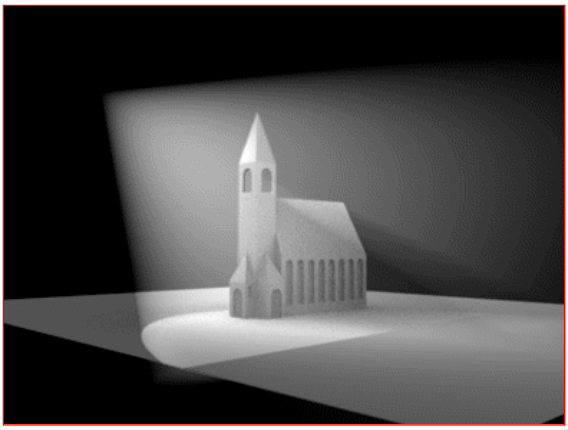


Now, we turn on the **Halo** option and set up a few of its parameters. Let's see how they work:

- **Samples**: With this option, we can adjust the number of samples used to generate the volume light. Higher values create better lights, but increase the render time.
- **Halo Step**: This option controls the sampling frequency. With values above zero, the light rays will be interrupted if they find an obstacle in their way.
- **Bias**: This option sets the shadow bias, to avoid artifacts generated by the shadow calculation.
- **Soft**: Here we can set up how soft the borders of the light projection will be. High values generate soft projections, but with a longer render time.

The bad thing about volumetric lights is that almost every parameter or option increases the render time. If your scene is complex, be careful when this type of light is required.

See how the volumetric light gives a more dramatic look to the render in the following image:



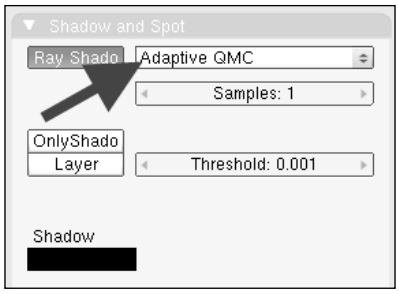
Camera view and Volumetric light

Suppose you set up a spot lamp to generate volumetric light, but nothing is happening at the render. Try to change the camera view, because it's a key point in the generation of this effect. Sometimes a simple rotation is enough to make the effect stronger for a particular view.

Soft shadows

All light types in Blender can generate soft shadows, which is a feature added in the 2.46 release. Actually, only one light type can't generate soft shadows, which is the Hemi Lamp.

These soft shadows can be set up in a small menu, available in all lamp types. Before the release of Blender 2.46, only the Area Lamp could generate this kind of shadow. Now, even a light type such as the Sun Lamp can generate these shadows. It works by simulating the existence of an Area Lamp, only just for shadow-casting purposes:



With the first combo box, we choose the method used to generate the shadows. The **Adaptive QMC** is the fastest for detailed scenes. With **Adaptive QMC**, the samples used to generate shadows are created incrementally to best fit the scene.

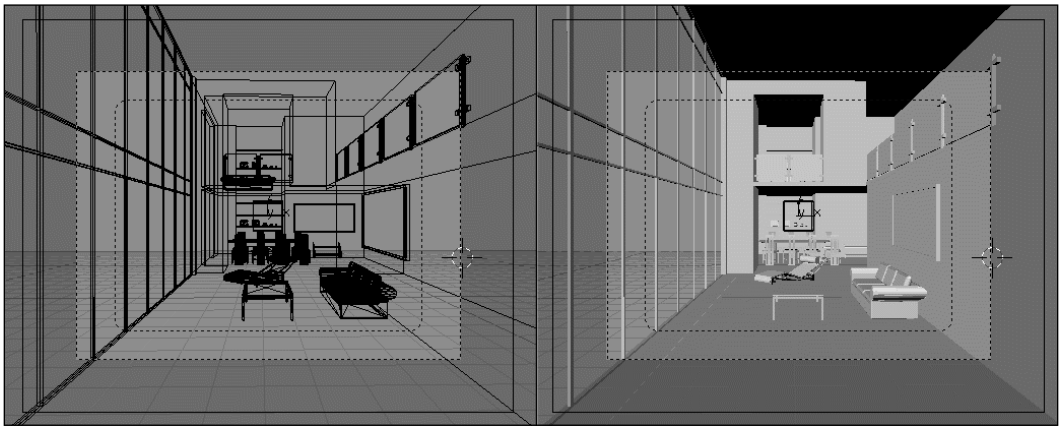
Below this option we can choose:

- **Samples**: Increase this value to make the shadows look better. But, high values can increase the render time as well.
- **Threshold**: This value controls the size of the shadow. If we increase the value of the threshold, fewer samples will be calculated for the shadow, making it noisy.

Lighting exercise

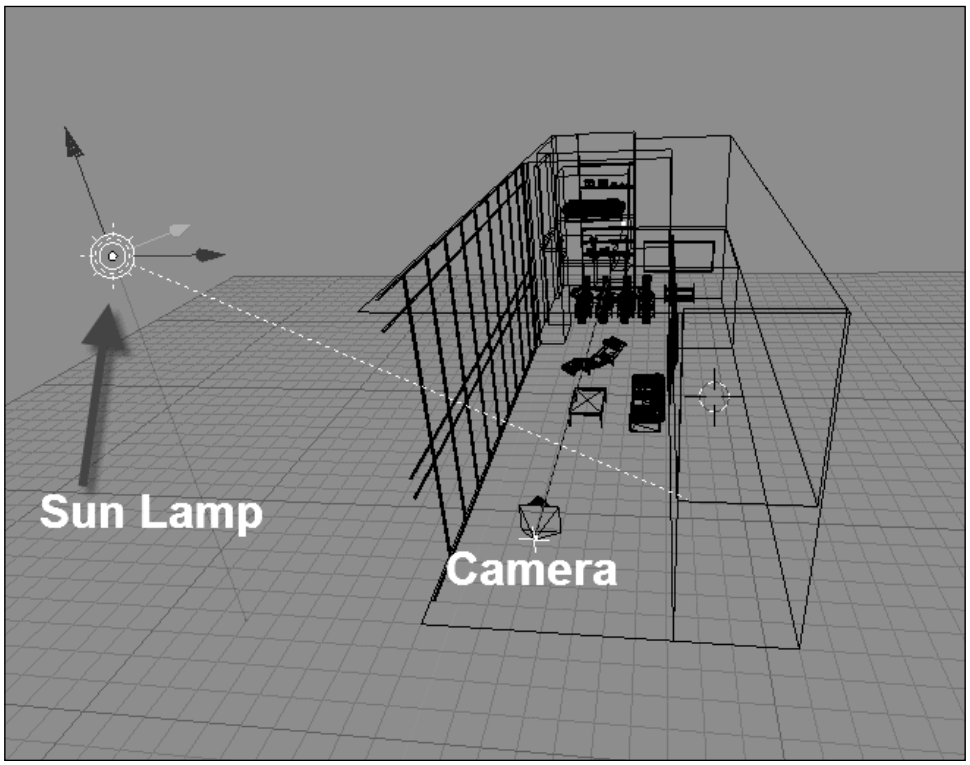
Now that we know about all light types in Blender, we will do an exercise to see how a light setup is done entirely with the standard lights of Blender. For this exercise, the scene used will be the solarium used for most of the examples of this chapter.

The model is quite simple and consists of an apartment with very big windows:



Before we do anything, let's plan how we will add the light to the scene. Because the environment has big windows, and the image will be simulating daylight, most of the light energy will enter the environment through the windows. Sunlight will be the main light source, and the only light casting a shadow.

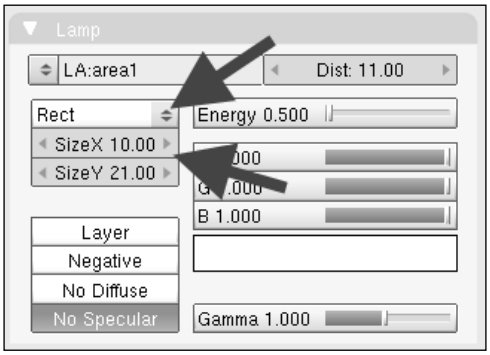
With that in mind, we can plan the light for our scene. The camera position is defined and represented in the following image. The sunlight will be created by a Sun Lamp placed behind the camera on the left side of the 3D model:



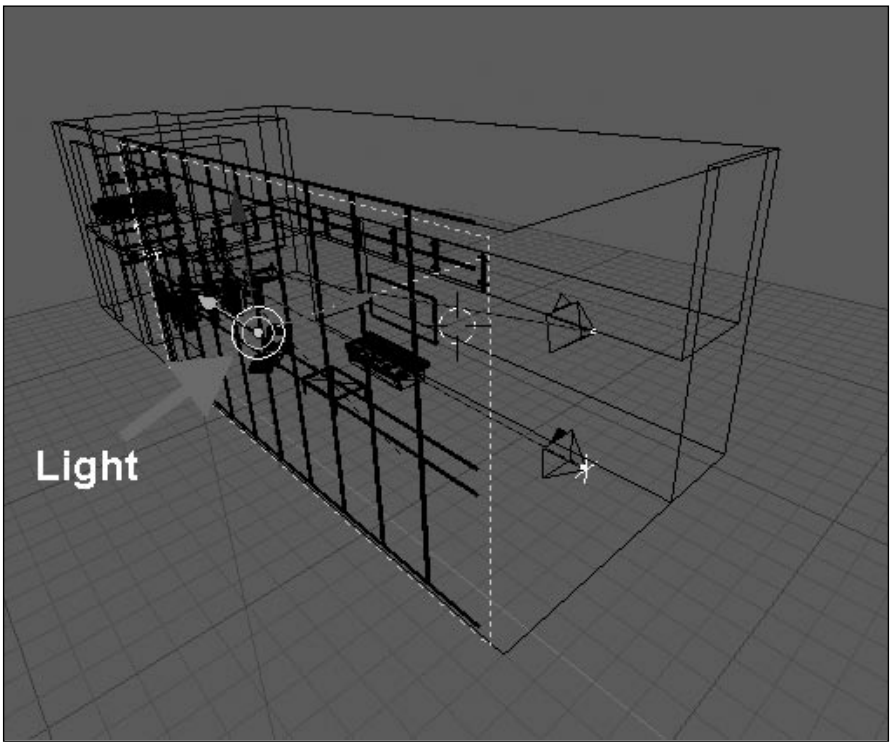
This Lamp is set up with energy of **0.30** and a slight yellow color. The **Ray Shadow** button is turned on to make this light to cast shadows.

If we render the scene now, the image will show some areas with a weak light energy and some completely dark. We have to simulate the light energy that comes in from the big windows. To do that, the Area Lamp is the best option.

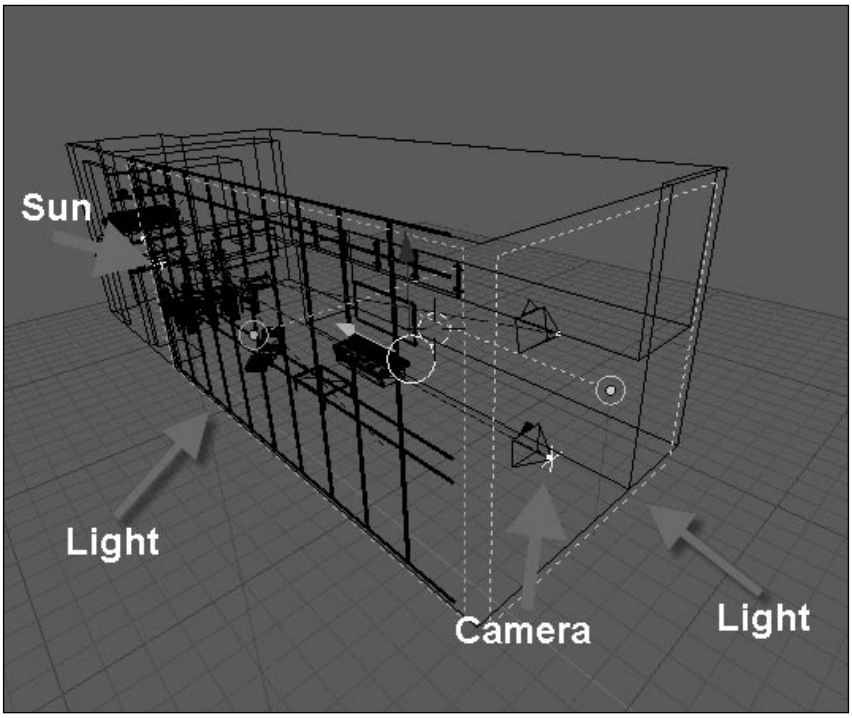
With an Area Lamp, we place the light emitter to fit the area of the window. Just create the lamp and change the area light shape to **Rect**. And then, use the **SizeX** and **SizeY** options to change the size of the lamp without changing the energy. If you don't want to face troubles with the energy of this light type, avoid the scale transformation to change the size of the Area Lamp:



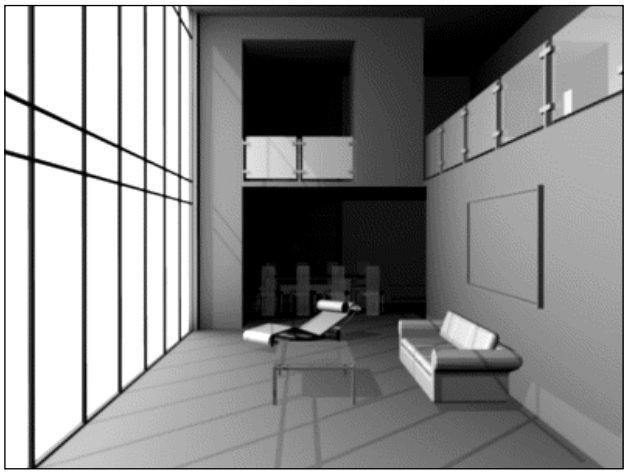
Let the shadows be turned off for this lamp. Place it in such a way as to fit the window size. Remember to point the light towards the interior:



Place another lamp on the other side. This new lamp will add light energy from both sides, where a big window will let the light come into the scene:

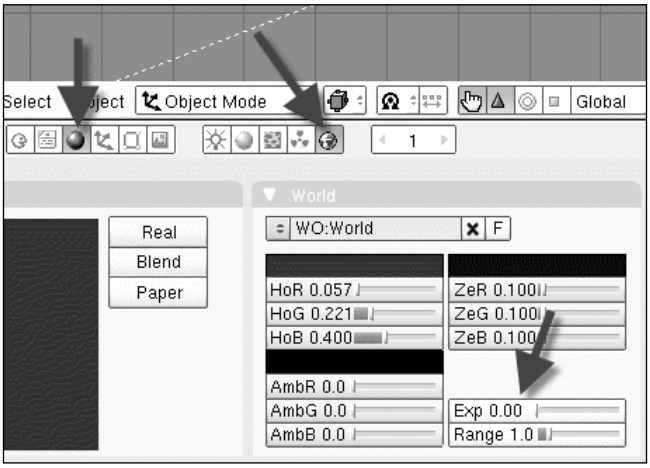


Both lights should be set up to have energy of **0.50**. If we render the scene, we will see the following image:



The light is starting to look good, but it requires more energy to get even better. We have two choices; the first one is to increase the energy for all lamps, and the second is to change the exposure for the scene.

Let's change the exposure, because it will enable us to increase the energy and make the scene brighter quickly. The controller for the exposure settings is located in the panel named **World buttons**:



There we will find the **Exp** slider. If we change this slider to **0.60**, we will make the image more sensitive to light energy and the overall rendering brighter:

Summary

We have learned how the light system of Blender works and when we must use a particular light type. In addition, we saw that all light types share common parameters, such as energy and influence area. What makes every light type unique is how it generates shadows and the way it casts light energy, which can make our life easier in some cases.

11
Radiosity and Ambient Occlusion

In the last chapter, we saw how to use lights in Blender. Now it's time to learn how the advanced lighting features of Blender work. The advanced lighting of Blender is based on two main techniques, named Radiosity and Ambient Occlusion. Both techniques give us better results for lighting environments, with less planning and lesser light sources. But, that comes with longer render times, because the major effort to generate good lighting now will be of processing time.

Which one of these methods is the best? When should we use Ambient Occlusion and Radiosity? To answer these questions, we have to first see how they work.

Global Illumination (GI)

When we use a standard method of illumination, light sources emit rays that stop when they hit a surface. And that's it! When the light ray finds a surface, it simply stops. If a surface is not touched by any light rays, it will be a black surface in the render. We have to set up the lights and their position around an object or scene to distribute light sources to make sure all surfaces get the right amount of light rays. This setup is the key to building good lighting. But, it requires a lot of work, and sometimes a bit of artistic touch to set up a scene. Most people don't have this kind of artistic feeling or even the patience to stay focussed on a single scene for a long time just adjusting lights.

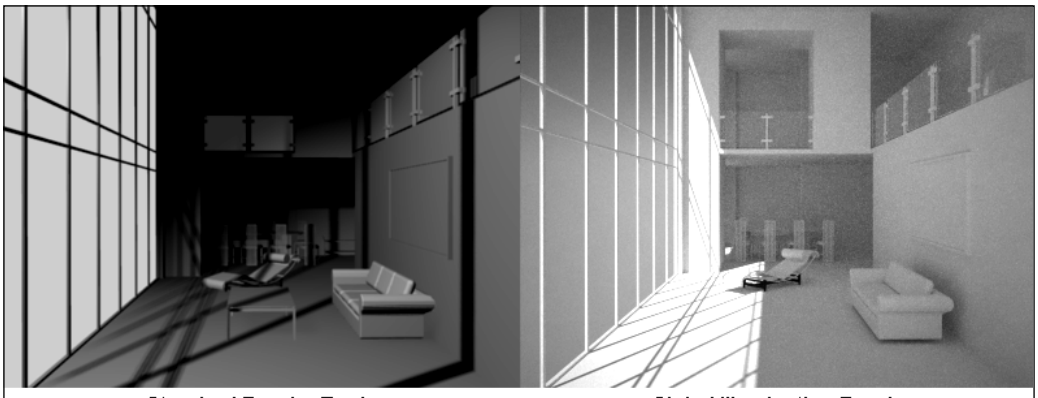
To aid such people, there is a method to distribute light energy through an environment named global illumination. This method is based on the bouncing of light energy on the surfaces of an environment, which is what happens in the real world. The light rays don't die when they hit a surface, they bounce and reflect, illuminating large areas just with this bouncing action. Depending on the amount of bouncing, a single light ray can illuminate a large surface, just with the bouncing energy.

In addition to the simple bouncing, there are some physical aspects of light and materials that can influence the result, such as the light or surface color. The lighting produced by a white light source is, of course, white. But when this light hits a red wall, the bouncing light will be red. With a standard light setup, we will have to add a red light near this wall to simulate the emission of red light energy. But a global illumination system can do that automatically, avoiding a complex setup in big scenes.

It looks as though a global illumination system is the best thing for us, right? Well, before using this kind of light system, there are a few points that must be clarified. The first thing about global illumination that we have to know (even if it looks easier to use), is don't think that a scene will be created without any kind of setup. It requires some setup, but it will depend on the method and renderer used.

Sometimes, it can be even more complex to set up a scene with a global illumination method, depending on the number of parameters and area that must be illuminated. Another important aspect is the amount of processing power required, which can make the process an impediment on some old computers. To build good lighting for a scene, the process is heavily based on mathematical calculations to determine the direction, amount, strength, and other aspects of light bouncing.

This means that a global illumination method requires fast hardware with a lot of RAM and a good processor to reduce render times. Otherwise, it will result in a long render:



Standard Render Engine **Global Illumination Render**

The bottom line is, with a global illumination system, we have a simpler setup of lights to achieve good illumination. But it will require a longer render time.

With this in mind, we can say that both Radiosity and Ambient Occlusion are methods of global illumination. We can't call the Ambient Occlusion a full GI method, but it works with the same concept, with a longer render time for a better illumination.

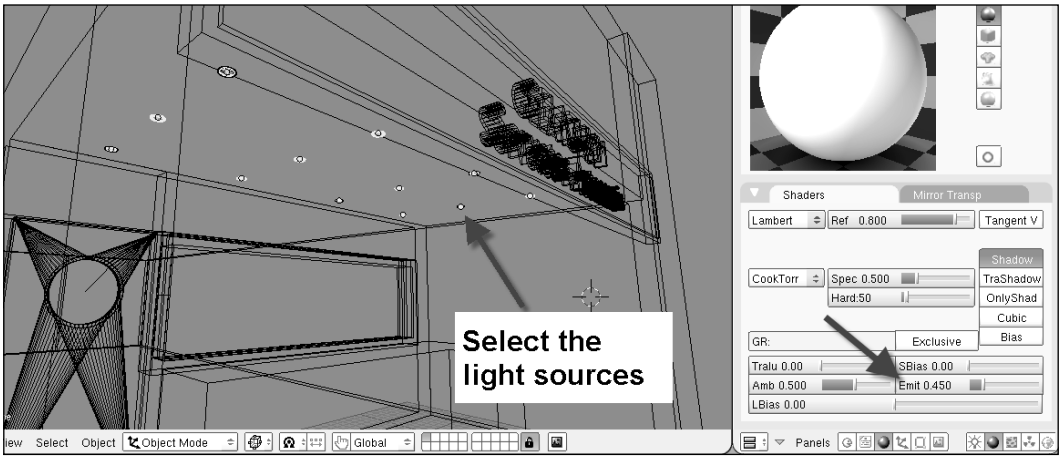
In addition to the use of Radiosity and Ambient Occlusion, we can use a GI method with an external renderer, such as YafaRay. We will learn how to use YafaRay in the next chapter.

Radiosity

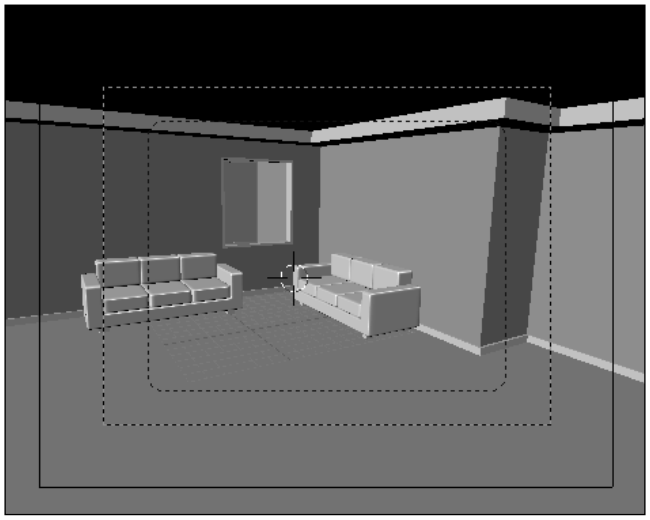
With Radiosity, we can generate good light for animations and games, because most of the illumination is processed just once, and stored as some kind of paint on the surfaces. Where can we use Radiosity? In Blender, this global illumination is primarily used for interactive presentations created with the Blender Game Engine.

Another interesting thing about Radiosity is that we don't use lamps to generate light for our environments. With this system, we use an object named "emitter" to generate light rays.

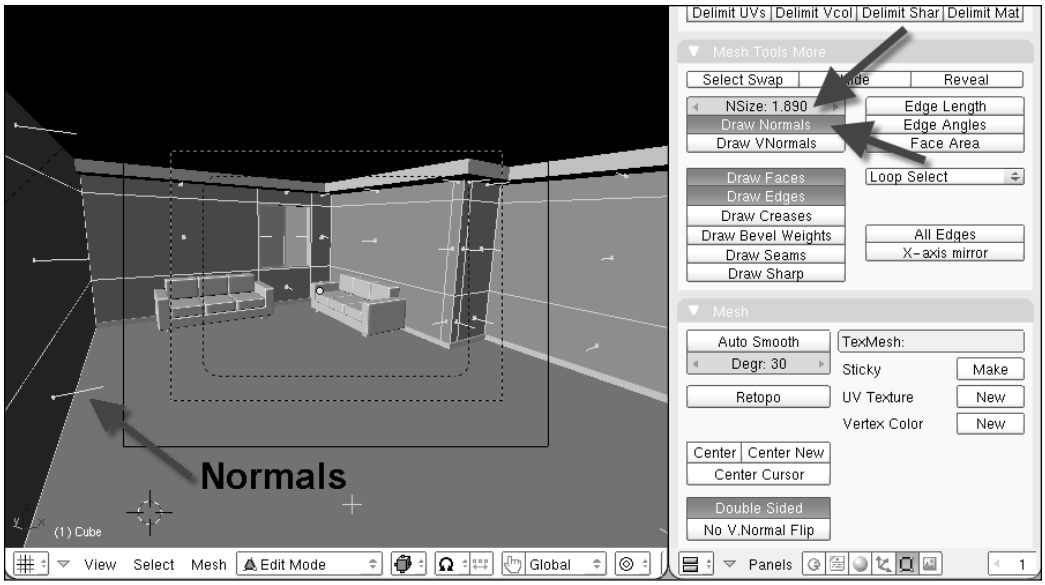
The way to create these so-called emitters is to change the **Emit** properties in the **Shaders** panel. For instance, if we want to create a plane, that generates light energy for Radiosity, we must apply a material to the plane and then change the color of the material to match the light color, and then increase the **Emit** property. The **Emit** option will work just like the Energy option for lamps, with high values generating a brighter illumination:



Let's see an example with a simple room. Just place the emitting object where you want the light to be placed. If the project requires more than one light, duplicate the object until you get the right number. When they are placed at their right positions, apply or create the material, with the **Emit** property adjusted to something greater than zero:

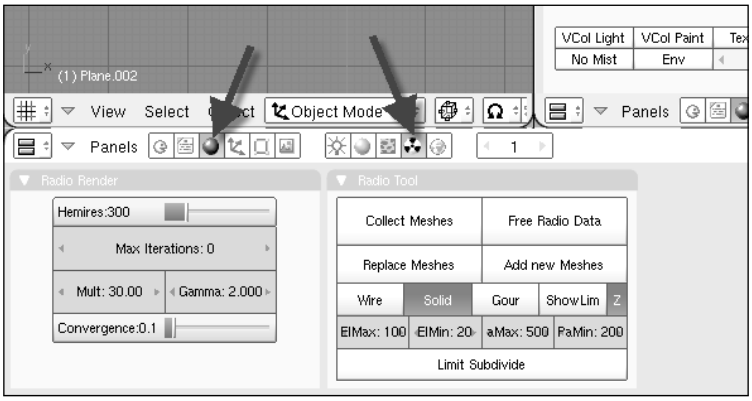


Before we start to actually do the Radiosity illumination, there is something we have to do. Double-check all the objects involved with the scene. The normals of all faces must be pointed towards the camera. If any normal is pointing away from the camera, the surface will look black. To change and ensure that all normals are pointing in the right direction, select all the objects in Edit mode, and in the **Editing**, panel press the **Draw Normals** button. It will make the normals appear as a small blue line, coming from the center of all faces:



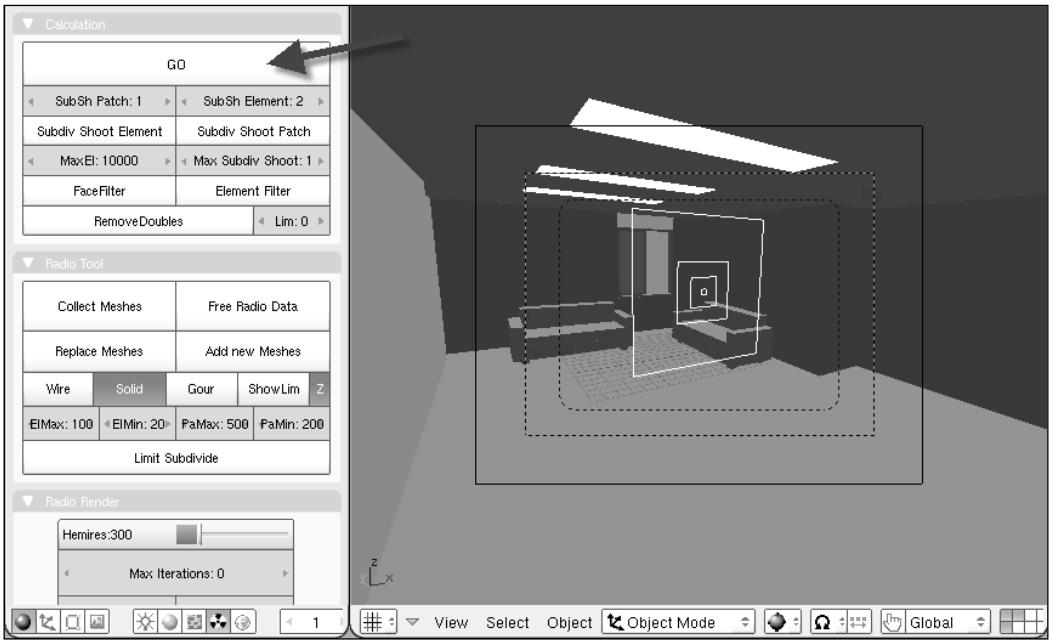
To change their direction, use the **Flip Normals** button in the **Mesh Tools** menu. Or, select all faces, and press *Ctrl* + *N* to recalculate all normals and aim them to the outside direction of the 3D model.

When all normals are pointing in the right direction, we can start to simulate the Radiosity solution. To do that, open the **Radiosity buttons** option. There we will find all buttons and options related to the Radiosity:

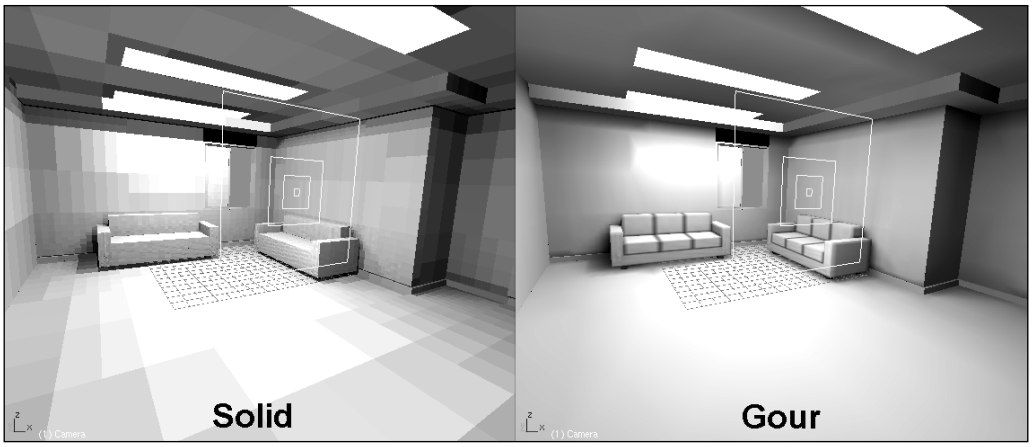


The process to generate illumination is simple; just select all objects in Edit mode and press the **Collect Meshes** button. When we press this button, all meshes will be temporarily converted to Patches, which are the objects manipulated by the Radiosity engine. These patches are made of triangles or square faces that can emit light energy.

Make sure that you have the View mode configured as **Shaded** (*Shift* + *Z*) to view all the details for the illumination. To start the calculation, press the **Go** Button. It will start the process and it will continue the calculation until the algorithm produces enough convergence. But, we don't have to wait that long. The calculation will update the light in the 3D View. By the time the light solution meets our requirements for the project, if it looks visually solved, stop the calculation. To stop it at any time, press the *Esc* Key:



See that we now have an image with light information stamped on all surfaces. Because the light is stamped with the default settings, we can see a lot of pixels on all surfaces. To make the lights smoother, press the **Gour** button. With the **Gour** option, the big squares in the left image will be smoothed and will mostly disappear, as the right image shows. The effect is very similar to what we get with the **Set Smooth** and **Set Solid** options in the **Editing** panel:



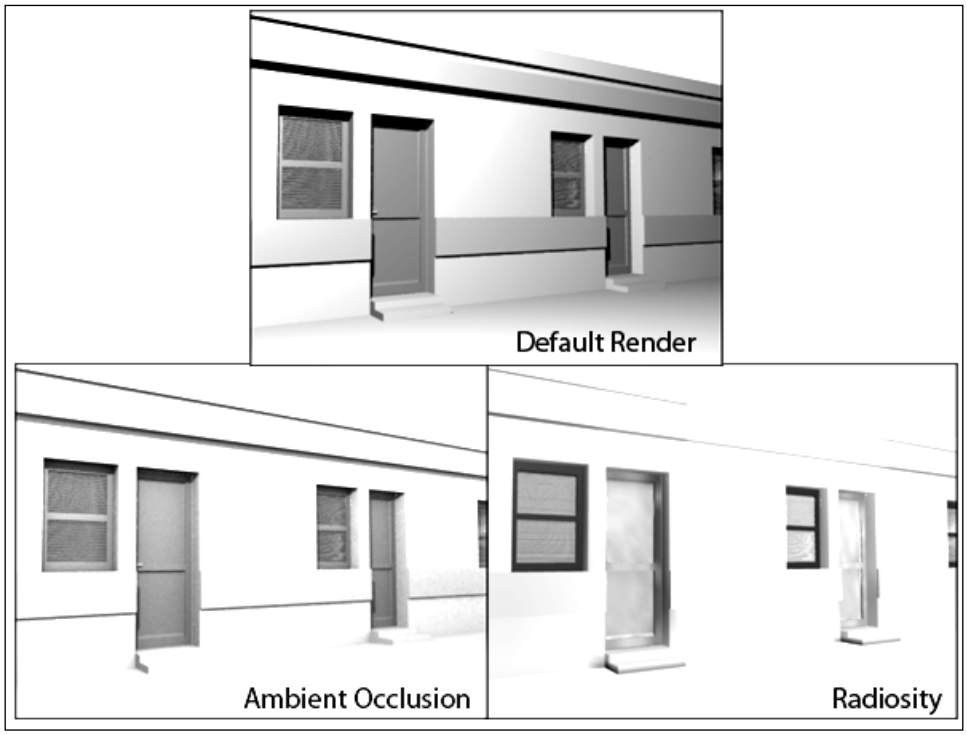
After we create the light distribution with Radiosity, it's not possible to change anything in the scene. If you need to change something, press the **Free Radio Data** button to make all objects editable again.

If we really want to have control over Radiosity, we must learn a few commands and what their controls do:

- **Max iterations**: With this option, we can set how many iterations the Radiosity solution should run. The default value is zero, but we can set it to a lower value. If it's set to zero, we will have to manually stop the calculation by pressing the *Esc* key.
- **Hemires**: This is a value that determines the quality of the Radiosity. If you need a better light distribution or smoother shadows, increase this value.
- **Convergence**: When we start the Radiosity solution, there will always be some portion of unshot light energy at the scene. This parameter can set the minimum value before the solution stops for this unshot energy.
- **Mult**: This option sets the multiplier factor for the light energy. Higher values will generate a brighter scene.
- **Add new Meshes**: Press this button to generate a new Mesh object with the color information from the Radiosity solution. The new mesh will have a material with the light information ready to be rendered. We will also be able to change the colors with the Vertex Paint tool.
- **Replace Meshes**: This option does exactly the same thing as **Add New Meshes**, but with the difference of replacing the original objects with the new, and illuminated meshes generated by the Radiosity solution.
- **Go**: With this button, we can start the Radiosity calculation.

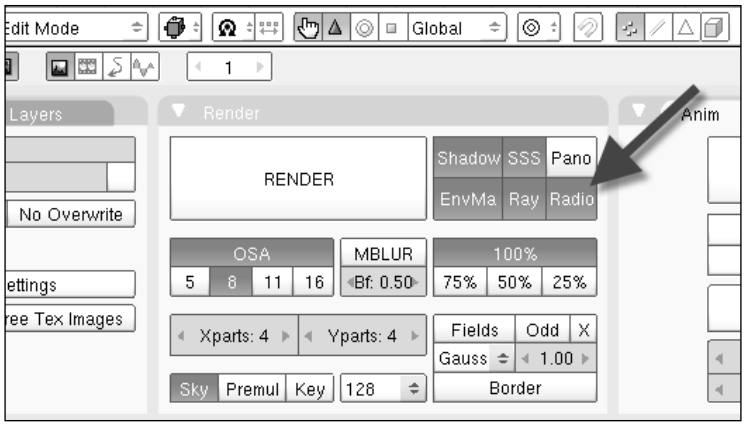
As we can see, the Radiosity option is best-suited for interactive visualizations, but not for printed presentations or video. For that, the best choice is to use Ambient Occlusion or another global illumination tool. The main reason is the better quality of shadows and the overall surface smoothing, which will result in a pixelation effect with Radiosity.

In the next image, we can see a comparison between the light generated by the Blender standard light system, Ambient Occlusion, and Radiosity:



With Radiosity, one thing with which you have to be careful is—always try to set up and adjust the color of your materials. Their color is what will define almost all reflections and soft shadows generated by light bouncing off surfaces.

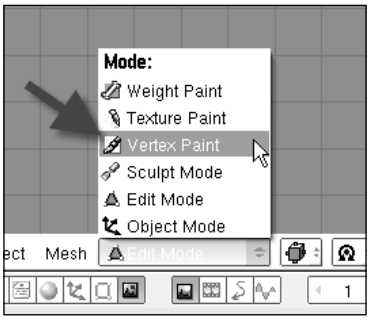
To use the Radiosity for rendering, just press the **Radio** button in the **Scene** panel. It will enable the Radiosity information to be considered for the final render. If you leave this button turned on, you won't have to calculate the Radiosity again. For instance, to render an animation, you won't have to change anything. The Radiosity will be calculated automatically for each new render:



Vertex Paint

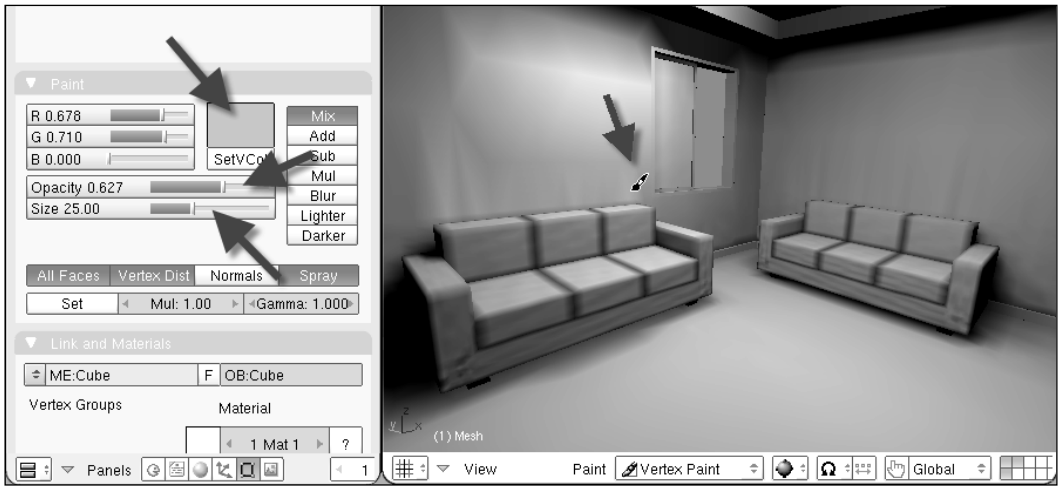
If you are not happy with the Radiosity solution generated by Blender, there is an interesting way to improve the lights and shadows generated by the calculation. Because the information for the Radiosity is stamped on all surfaces as vertex colors, we can literally paint these colors if want to improve the solution.

To do that, just select the model and enter the **Vertex Paint** mode:



In the **Editing** panel, choose from all the tools available to paint the vertex of the model. For instance, if the shadows in a scene are not so great or need a bit more color, just select the desired color from the selector and adjust the **Opacity** and **Size** for the brush.

Hold and drag the mouse cursor over the faces that you wish to paint. It will add the color information to the faces:

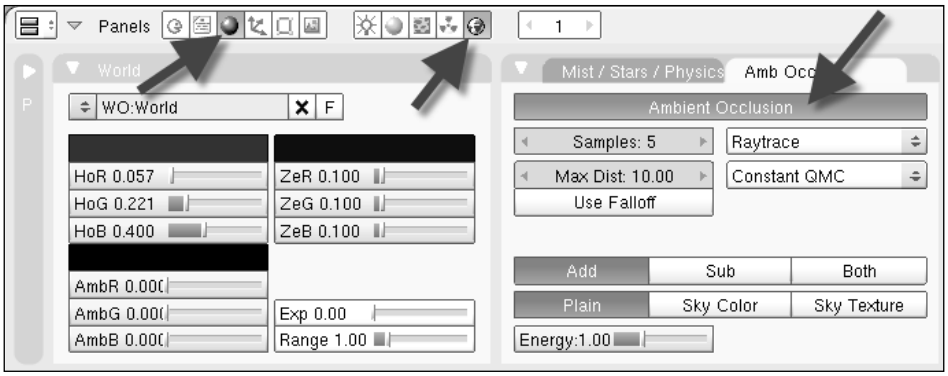


Use it as much as necessary to improve the lighting. By the end, we will have a scene prepared for interactive visualization. If you want to use Radiosity to light a scene for print, be aware that your scene may need a lot of work and adjustments to present a good light.

Ambient Occlusion

The Ambient Occlusion is the best way to simulate a global illumination environment for architectural visualization without the use of any external renderer in Blender. It works in this manner—when we turn on the Ambient Occlusion option, the background of the scene will randomly cast light rays to the center of the scene. When a light ray hits a surface, it stops. It will make objects near each other block rays that would otherwise hit each other, causing shadows to be cast. The surfaces will be stained with the shadows, which will create the effect of a global illumination render.

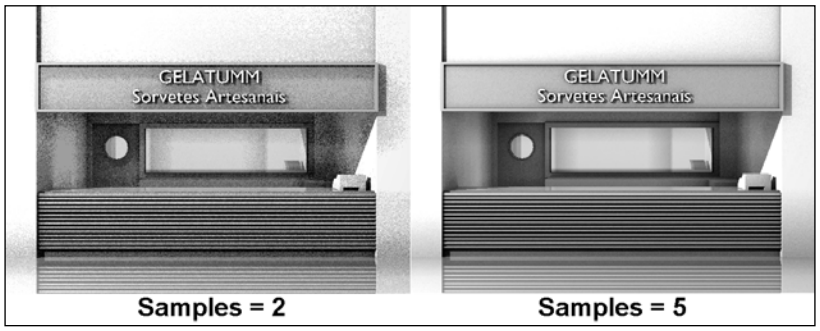
It will give us a better lighting, but, of course, a longer render time for any scene. To use the process is very simple. We just have to set up the **Ambient Occlusion** menu, located in the **World** panel:



The menu is very simple, with only a few parameters. There is a button named **Ambient Occlusion**, which turns on the Occlusion effect.

When the button is turned on, we have to set a few parameters. There are two main occlusion methods, which are the **Raytrace** and the **Approximate**. All images generated by the **Raytrace** Ambient Occlusion have a common characteristic, which is the noisy aspect of the images. This aspect can turn them into higher or lower quality images and increase the render time. With less noise in the image, we have higher quality and render time. To control the amount of noise in the image, we use the number of **Samples**. With a high number of **Samples**, the longer the render will take, but our renders will get less noisy. The occlusion method named **Approximate** uses a technique that can create noise-free renders, which are suitable for animation. The following image shows a comparison between scenes rendered with both methods:

For architectural visualization stills, the best option is to use **Raytrace** occlusion:



The main parameter of **Raytrace** Ambient Occlusion, which controls the level of noise for the render, is **Samples**. The value of Samples always starts with **5**, but we can increase this number up to **16**, which is the best quality. It's very rare for a project to require a number above **10** the **Samples**. This setup already results in a very good render quality.

For testing purposes, we use a **Samples** value of **2** or **3**. This will result in a very grainy image, but with a very fast render time.

Let's see how the other parameters for the Ambient Occlusion menu work:

- **Max Dist**: With this parameter, we can set up how far the light rays are cast. If we set up the **Dist** with a high value, it will make the light go even further, resulting in a brighter scene.
- **Use Falloff**: Setting up a distance for the light rays is not enough to make them work. To use the distances, we have to turn on this button. When the button is turned on, surfaces located far from the light source will be less illuminated.
- **Add**: With this option, we will create light energy to be added to the scene.
- **Sub**: Here, we can set up the Occlusion to remove light energy from the scene. If this button is turned on, remember to place a lamp at the scene, or the render will result in a black image. This option is good for night scenes and environments that require a lot of shadows.
- **Both**: Here, we can use the Occlusion to Add and Remove light energy from the scene.
- **Plain**: The three buttons placed at the bottom of the menu determine the color of the light. With this option, we can choose the light rays to be white. In most cases, this is the best choice for the Ambient Occlusion, which reproduces most light sources, such as the sun or artificial lamps. Unless the project requires a different light source, choose this option.

- **Sky Color**: If a white light source is not enough for your project, we can set up the Ambient Occlusion to use the **Sky Color**. This is the color used as the background. If you want to simulate a clear sky, try to set up a light blue as the background color. For alien environments, such as Mars, use a redder background.
- **Sky Texture**: To use something more complex as the light color, such as a texture, we can use this option.
- **Energy**: Here, we can set up how strongly the light rays will hit the environment. It works exactly the same as the energy parameter in the lamps. High values will result at a brighter scene.

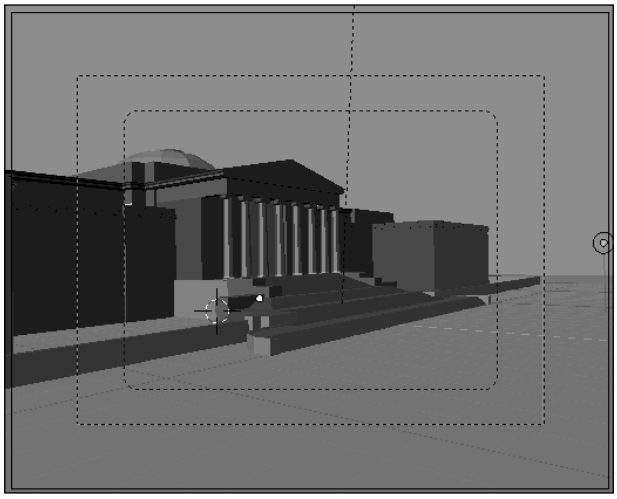
It's all about finding the right setup for a scene, so let's see how we can set up both indoor and outdoor scenes.

Approximate Ambient Occlusion

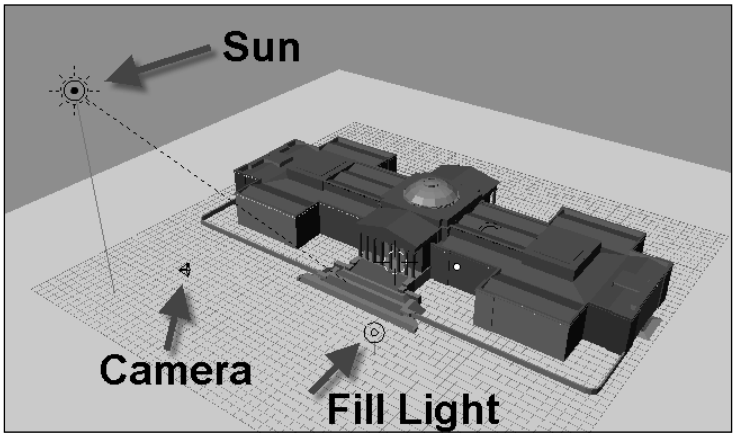
Use this occlusion method only for animations, because it's not as good for lighting as **Raytrace** occlusion. But, if you want to use the method, adjust the **Error** value to control the quality of the render. With low values, we have better occlusion effect but slower renders.

Outdoor scene

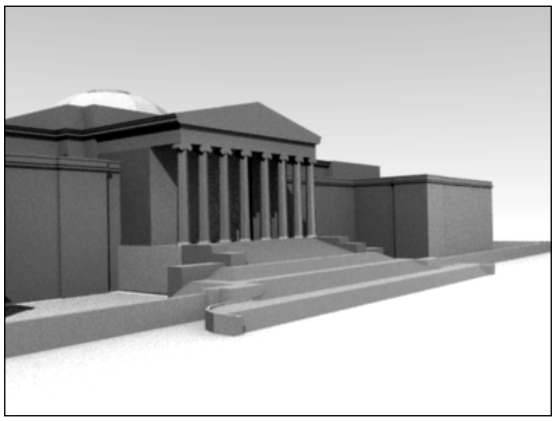
When we have to light an outdoor scene, the setup will be extremely simple, with only a few lights. In most cases, we will need only two main light sources, to simulate the sunlight and a small fill light. The fill light will make the darker areas of surfaces a bit brighter, because the sunlight will hit the surfaces only on one side:



Take a look at the scene given as an example. It has a small model with a museum. To light this scene, place a Sun lamp right at the position of the Sun. And then, place a lamp or Hemi at another side, to make it work as does a fill light:



Remember that the position of the light will depend greatly on the camera view. That's why we always have to decide where the camera will be before placing the light points. When the lights are placed, just run a small setup through the Ambient Occlusion options:



Ambient Occlusion for Outdoor scenes

This is the scenario where the setup for Ambient Occlusion will be easier. The only precaution we have to take is to choose the camera position before doing anything. If your scene has any special requirements, such as a specific sun position or time of the year, it may require some changing of the position of the lights.

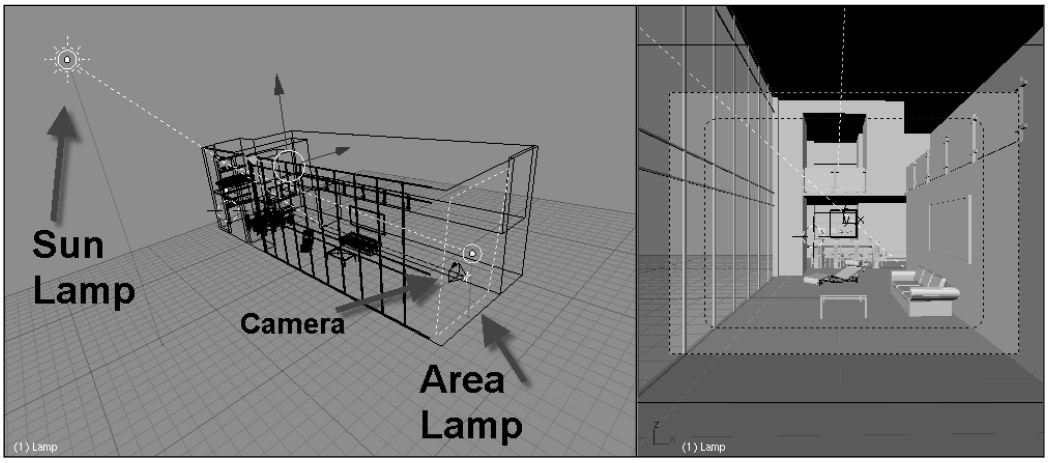
Indoor Scene

An indoor scene is a bit more complex to set up, because the lights can react to a lot more surfaces and other factors. Similar to the outdoor scene, we have to determine where the camera will be placed before starting to work on the light. When the camera is placed, we must answer a few questions to set up all light sources for the scene:

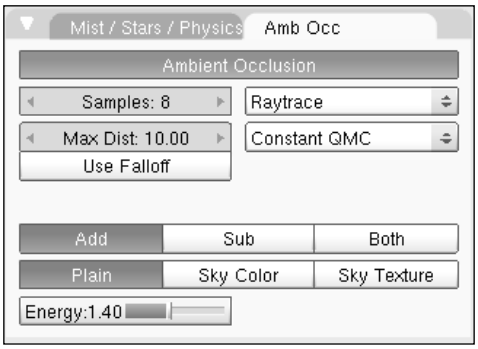
- How will the colors of the surfaces interact with the light?
- Is the main light source for the scene a natural or artificial light?
- From where is the sun coming into the scene?
- How strong are the artificial light sources? How many of them are placed at the scene?
- How will the windows or doors interact with the light sources?
- Is there any piece of furniture with material that can change the way a light source works?
- How big is the scene?

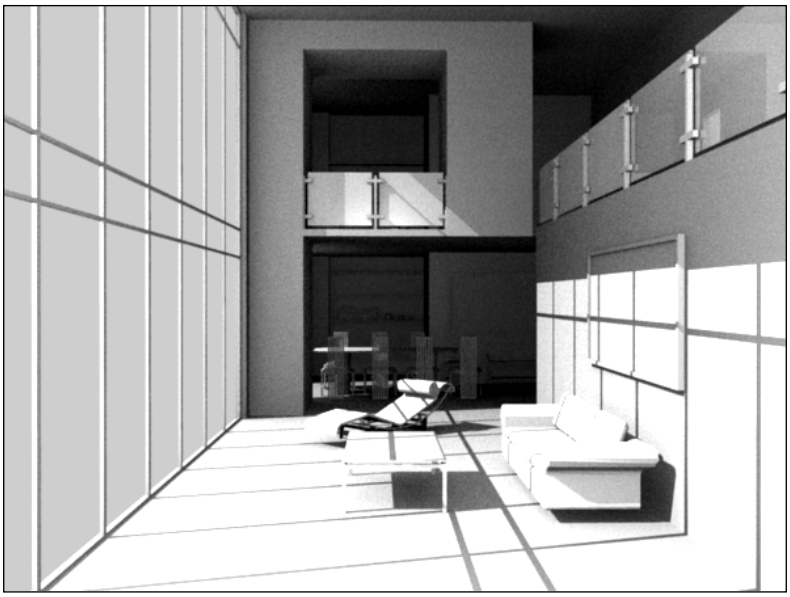
If we can answer those questions, the setup of all lights will be a lot easier. But what if we can't? Well, the answers for those questions should be in your project if it is already completed. What I meant is—look at the answers before you start to make tests for light. You should use a 3D environment such as Blender to test concepts. But these concepts require a minimum level of planning; otherwise, it will turn out much like a guessing process. And, believe me when I say that this kind of process can and will consume a long time.

Take a look at the following scene. The lights are placed in such a way as to simulate the sunlight coming in from a window:



When we take a look at the setup for the **Ambient Occlusion** and the lamp, we can see that a lot of adjustments are required to fit the scale of the scene:





Measures and light

Always use the same measurements for all scenes. It will make the light setup process a lot simpler. If you always use the same units, when the lights needed to be placed, you will already know the right values for energy and the Ambient Occlusion settings.

Working with interior light is a very complex subject in architectural visualization. To successfully reproduce the light for an interior scene, you should start by observing light conditions in closed rooms. With these observations, you will be able to reproduce the light by adding lamps with a small amount of energy.

A good exercise would be to find some good tutorials or images of interior scenes, even if it's created with another 3D software package, and recreate it with Blender. Because it's a general subject in architectural visualization, you will be able to reproduce these tutorials in Blender.

If you want to try the results of a global illumination solution, just go to the next chapter, where we will learn to use YafaRay.

Summary

In this chapter, we have learned how to use the Radiosity and Ambient Occlusion options to create a better illumination for our scenes.

With the Radiosity option, we could generate a lightweight solution to create shadows and interactions between elements. The light is distributed at the scene using a realistic energy distribution, with the light rays bouncing off the objects' faces.

Even being a great solution, the illumination generated with the tool is used mainly for interactive animations.

For more sophisticated illuminations, there is the Ambient Occlusion option. With this tool, we can simulate a global illumination environment, creating a smooth light solution.

12
Global Illumination with YafaRay

What is YafaRay? It is an external renderer, which works well with Blender. Actually, it's the external software that presents the best integration with Blender. We can even call YafaRay directly from the Blender interface using a script. With YafaRay, we can use tools and features that aren't available with the default Blender renderer, such as global illumination and a very fast ray tracing. There are some other effects that we can do with YafaRay that we can't do easily with Blender, such as caustics and creating self-illuminated objects.

In the last two chapters, we have learned the tools and techniques to illuminate our scenes along with a wide range of options. If you tried them, and felt that they were too difficult with which to create a good illumination, use YafaRay and its global illumination render system. When we use it, the light rays emitted by lamps will bounce off the surfaces of our models.

But, there is a price to use YafaRay, which is that the consumption of resources from the computer is higher. So, if you base all renderings on a global illumination solution, be ready for long render times.

In this chapter, we will take a look at examples of renderings created with YafaRay, such as the ones shown in the following images:

These are good examples of what we can do with Blender and YafaRay, and we will see in detail how to set up each scene to create both images.

Because it's an external renderer, the first thing we have to do to use YafaRay is to visit the official web site and download a copy of YafaRay.

Installing YafaRay

When we get to the YafaRay website (`http://www.YafaRay.org`), there will be a download area where we can choose to download a copy of YafaRay compatible with our operation system. Don't worry; they have versions for almost all major operating systems, including Microsoft Windows, Linux, and Mac OS X.

Before the installation of YafaRay, we have to download and install Python in our system. In some cases, like with the Mac OS X, the system already has a Python version installed. But for most Windows systems, the download will be required. Because the integration between Blender and YafaRay is made by a script written in Python, if we try to use YafaRay without Python for rendering, nothing will happen. For Blender 2.49, we must download Python 2.6.2 from `http://www.python.org`.

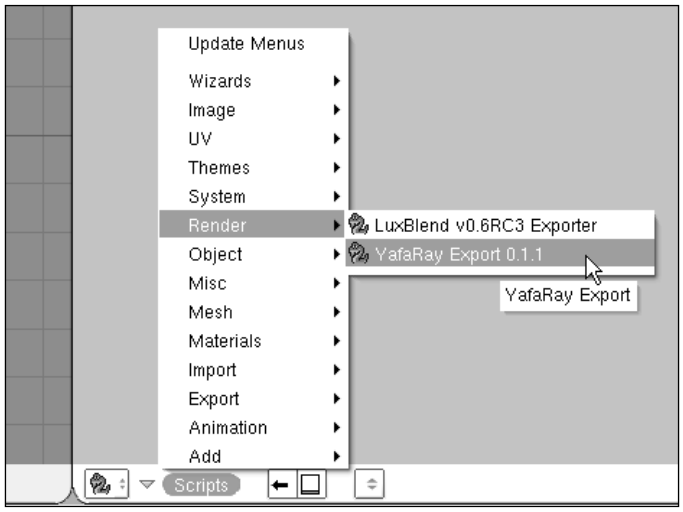
Only after installing Python can we install YafaRay. After installing the YafaRay renderer, nothing will happen. YafaRay is a command-line renderer, which can only be used from the command line. To make things a lot easier, there is a way to call YafaRay from the Blender interface with the script.

Troubleshooting YafaRay installation

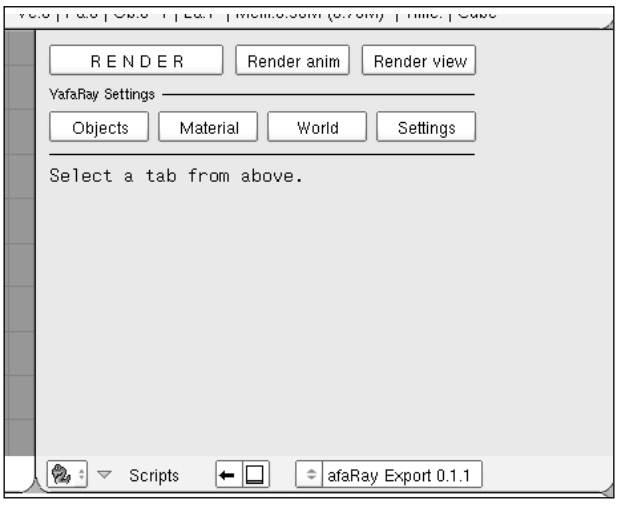
The installation of YafaRay can be painful for some people, because, in some cases, the files won't be placed in the right folders. If you don't see a YafaRay exporter in the Blender scripts window, look in the Blender `scripts` folder and for a find a file named `yafaray_ui.py`. If it's not in the `scripts` folder, find the file and copy it to the `scripts` folder. The scripts folder is usually at `/appdata/Blender Foundation/Blender/.blender/scripts/` for Windows users. For Linux users it is at `/usr/share/blender/scripts/`. For Mac OS X, it is at `/usr/local/share/yafaray/blender/`.

Blender and YafaRay

To use Blender and YafaRay, we only have to open a new window in the Blender user interface, and change the window type to **Scripts**. In the **Scripts** window, we will find the YafaRay exporter in the **Render** menu item. Another way to call the YafaRay exporter is by using the **Render** menu, located at the upper part of the interface. Both options will only appear if YafaRay is installed on your system. If they don't appear, run the installer again:



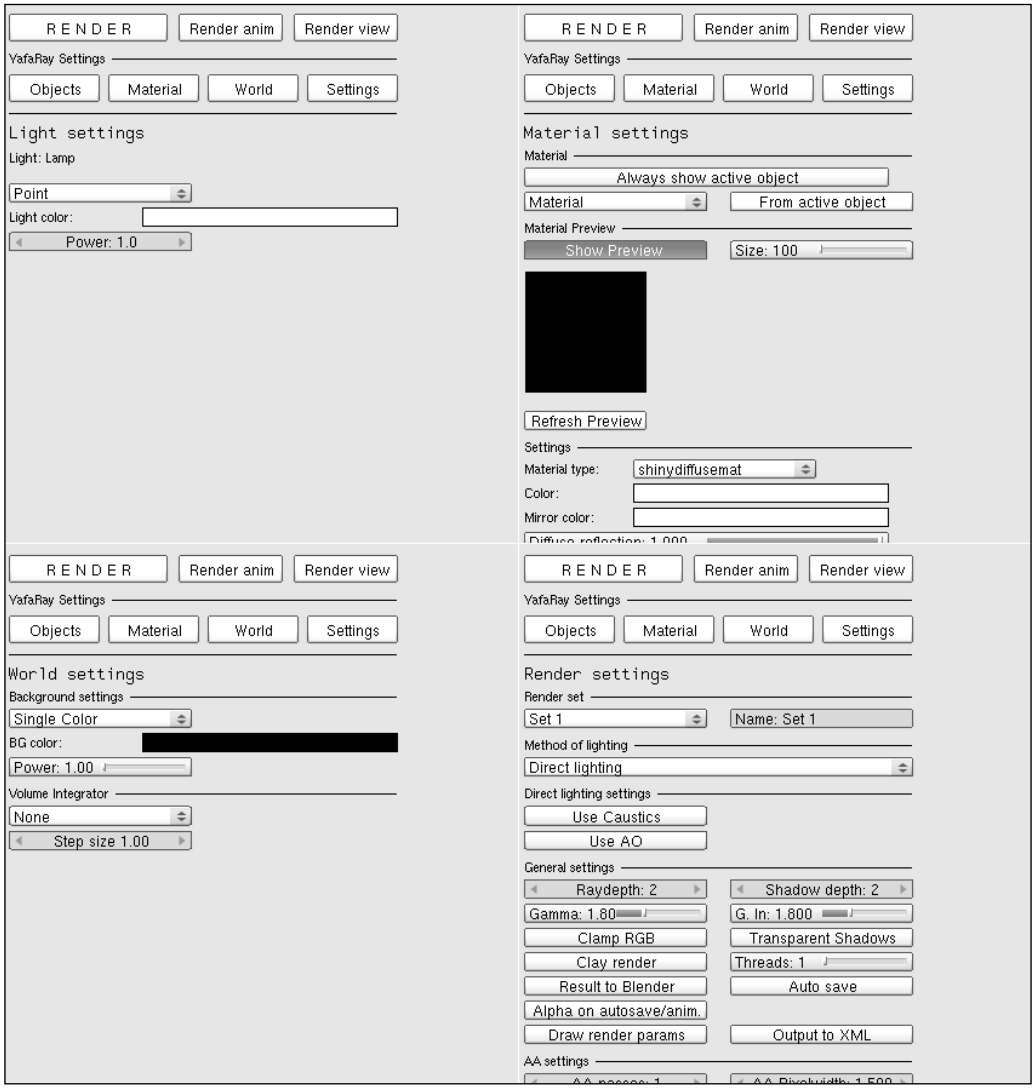
When we call the renderer, a panel with options to set up YafaRay will appear:



To simply start a rendering with YafaRay, press the **Render** button. But, in order to create a better illumination and use advanced materials, we have to change a few settings in two key panels, which are the **Settings** and **Material** panels.

YafaRay setup

The setup of a render in YafaRay is made in four different panels, which controls individual aspects of our project. We can access each panel using the second row of buttons in the YafaRay exporter interface:



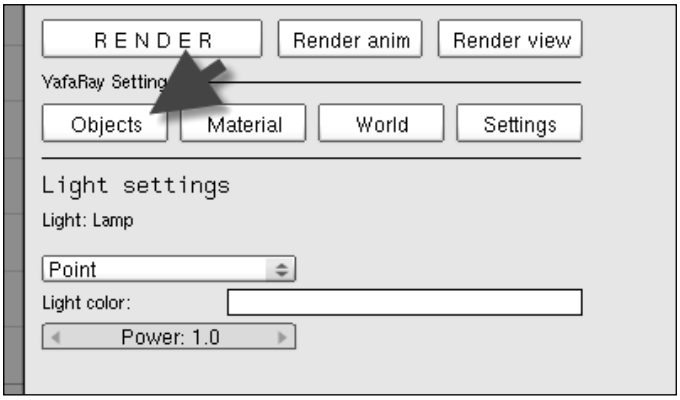
Here is what we can do with some of their parameters:

- **Objects**: This panel can change the properties for individual objects such as cameras, lights, and meshes. For instance, we can make an object emit light energy or change the behavior of a camera.
- **Material**: Most of the parameters for materials in YafaRay come directly from the Blender standard materials. But, we can change a few settings especially for YafaRay, such as the material used for rendering, with a few presets.
- **World**: Here, we can change the way our environment will look, and even set up a sun or moon to be present at the scene.
- **Settings**: From all of the panels, this is the most important for global illumination rendering, because it is here that we change the settings for rendering with photorealistic quality.

Let's take a look at how to work with each of the panels in greater detail.

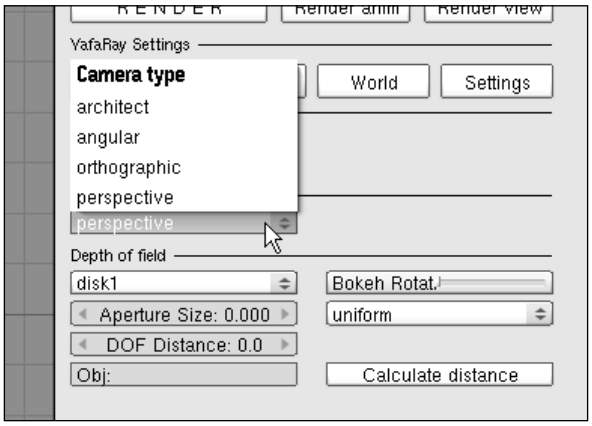
YafaRay objects

With the **Objects** panel of YafaRay, we can change the way objects will behave during the rendering. The panel changes depending on the type of object that is selected, showing options related to the nature of each selected object. For instance, the options related to mesh objects will be different from light sources. We see an image showing the options for the **Object** panel when a light is selected in the following screenshot:



Cameras in YafaRay

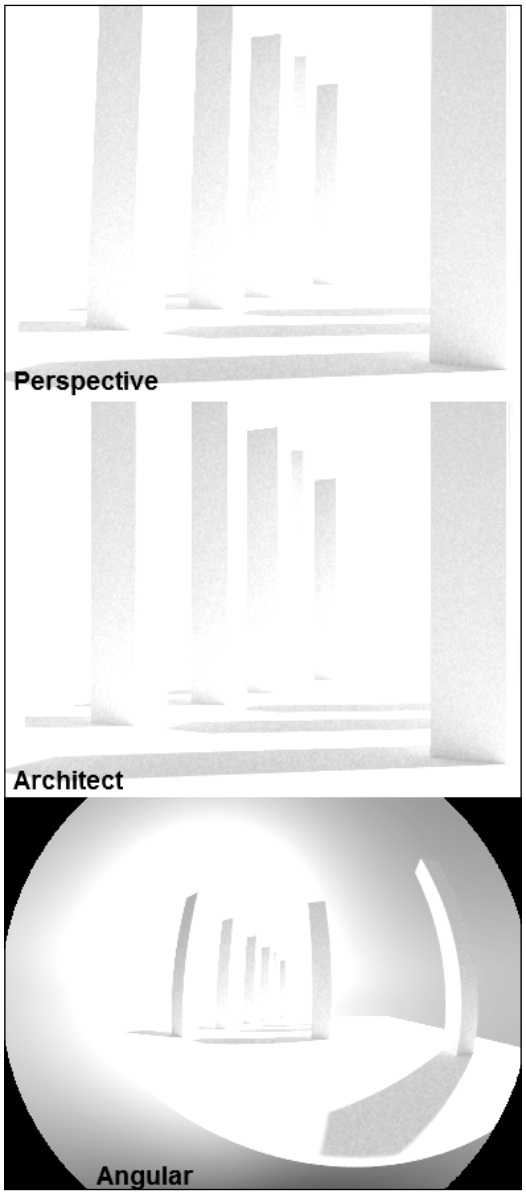
The use of cameras in YafaRay will give us a few more options for architectural visualization than the standard Blender camera. Right after we select a camera in the 3D View and open the **Object** panel, we will find several projection options for the camera. The default camera for YafaRay is the **perspective**, which will give us the projection of lines with vanishing points. Following is an example of the **Camera type** selector in YafaRay:



In addition to the **perspective** camera, we also have another three camera types:

- **orthographic**: With this camera, the view of the scene will be similar to the one we get in one of the orthographic views of Blender such as the front, right, or top views. All lines and projections from the image will be rendered without the **perspective** effect.
- **angular**: Here we have a type of camera that simulates a spherical projection. Our image will be deformed to fit into a sphere.
- **architect**: When this type of projection is used in the active camera, the result will be a rendering with the vertical lines perfectly parallel to each other.

The following image shows an example of the **architect** camera projection compared with the **perspective** and **angular** types:



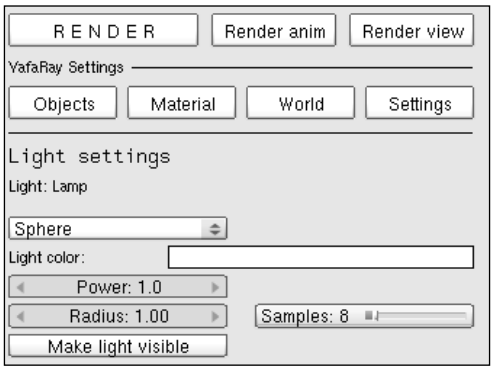
From the four types of cameras in YafaRay, we will primarily use the **architect** and **perspective** types for architecture, because they produce the most common type of projection used in visualization projects.

Lights in YafaRay

To use the YafaRay lights, we can use the same lights with which we are used to working in Blender, with the difference being that they will be exported to YafaRay with new properties. Each time we select a light source in the 3D View, and open the **Object** panel in the YafaRay exporter, the options related to this light for YafaRay will appear. Depending on the light type created from standard light types in Blender, we will see different light types in YafaRay. The only light type in Blender that doesn't have any related light in YafaRay is the Hemi.

The options available to set up the light types may change a bit, depending on the light selected, but some of those options remain the same. For instance, all light sources have a **Power** slider and a **Light color** picker. Other light sources also offer:

- **Make light visible**: With this button turned on, the light source will be visible in the rendered image
- **Radius**: A few light sources, such as the Sphere type, can be sized to any value with this option
- **Samples**: This is a value that controls the quality of the light. High values produce soft light



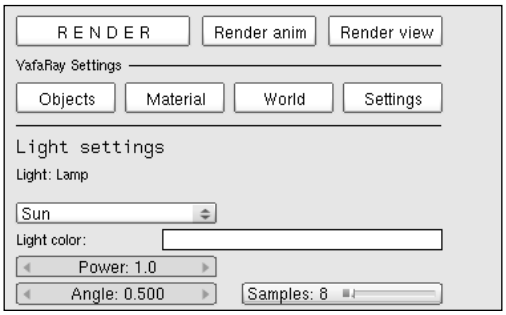
Selecting a Lamp

When we select the Lamp light in Blender we will have two types of lights available in YafaRay, which are the **Point** and the **Sphere** types. The first one is a very simple light that functions as a small point that casts light to all sides. It has the basic light setup options.

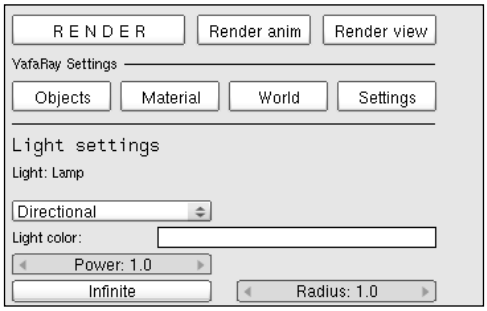
In the **Sphere** type, we have a light source that looks like a sphere and can be used to create globes of light. It has, along with the basic light options, a button to make it visible, and the **Radius** slider to choose how big the sphere will be.

Selecting a Sun

The Sun lamp in Blender is the best option to simulate direct sunlight in architectural visualization projects. And in YafaRay it's no different. In fact, with YafaRay, we have two options to simulate direct sunlight. The first one is the **Sun** that has primarily two options, which are the **Power** and **Angle**. With this last option, we will set the angle of the cone projection that will generate the sunlight. It controls the softness of the shadows produced by the sun:



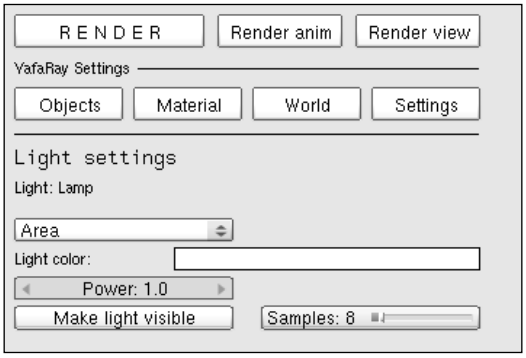
The second option that we use to work with Sun is **Directional**. Both lights try to simulate sunlight, but with **Directional**, we have light projection using a cylinder instead of a cone. The cylinder can be set up as an **infinite** cylinder or a semi-infinite cylinder:



Which one is the best for architectural visualization? It will depend on the model that we want for the project. Both models basically differ in the way they generate shadows, because the Directional is based on the parallel projection of rays. This will generate hard-edged shadows on the scene. With the Sun model, we have a cone projection, which will generate soft shadows on the scene.

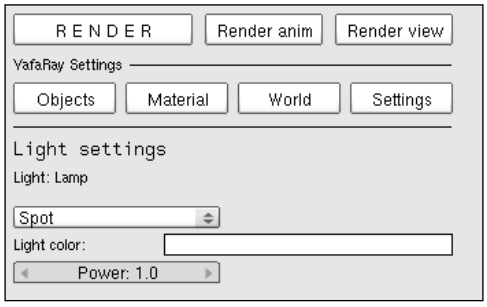
Selecting an Area

By selecting an Area Light in Blender, we will only have one option in YafaRay, also named **Area**. The light principles in both lights are the same, and they work as a plane that can cast shadows. This light type can produce soft-edged shadows, and appear on reflective surfaces such as glass and mirrors. If you want the light to appear on the render as well, make sure the **Make light visible** button is turned on:



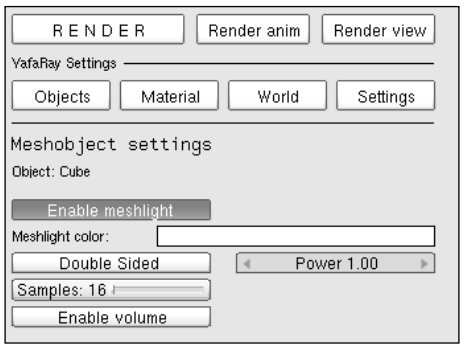
Selecting a Spot

The selection of a Spot lamp in Blender will result in a **Spot** light in YafaRay, and they share several properties. All settings to control the Spot size and parameters remain in the Blender interface, with only the options to control the **Power** and **color** of the light in YafaRay. This light behaves exactly the same way as the Point light from the lamp options, with the difference being that we have directional controls:



Mesh light in YafaRay

If none of light types in YafaRay will fit in a project, we can always use a Mesh object as a light source. Select an object in the 3D View, and in the **Objects** panel of the YafaRay exporter script, choose the **Enable meshlight** button. This will make the object emit light rays from all faces:



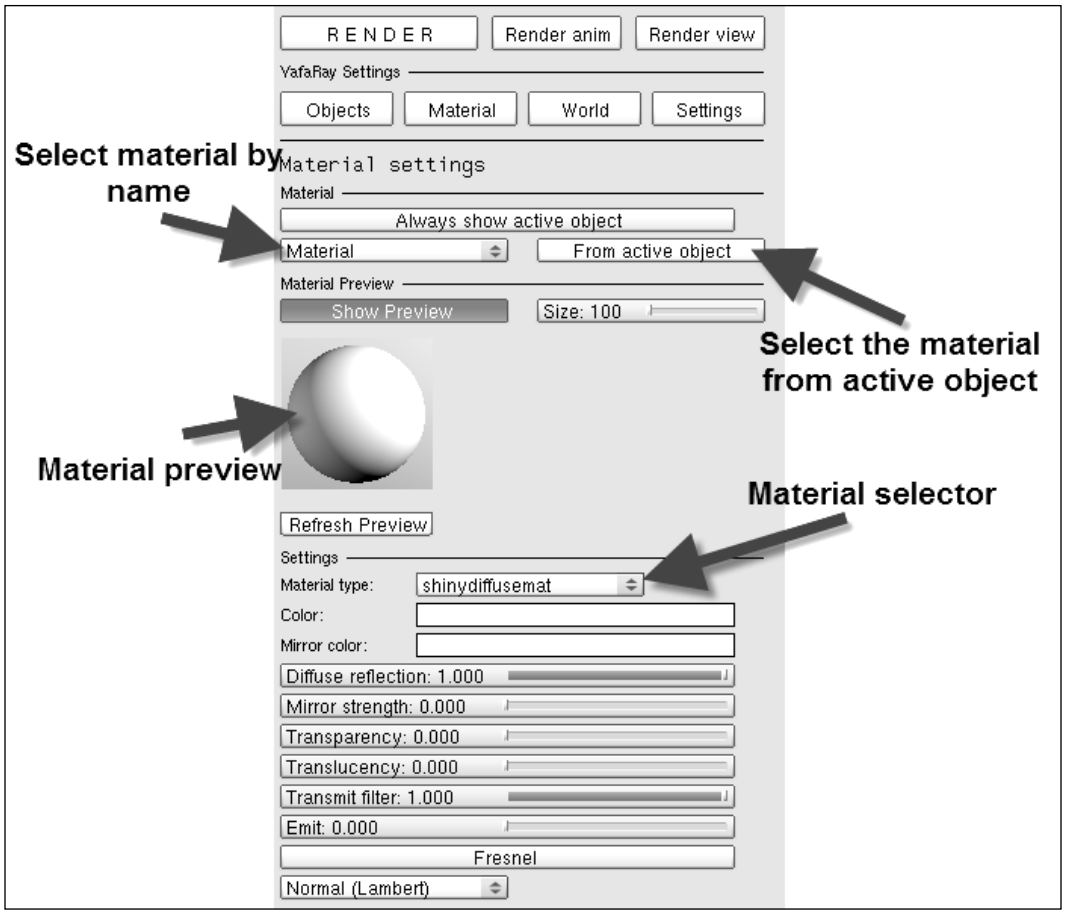
This option can be used in several architectural visualization projects to simulate light bulbs or neon lights. Every time we have to use lights with odd shapes, the Meshlight is a great option to model any type of light source.

If the light is made out of planes, make sure that the **Double Sided** button is turned on to make the light emit energy from both sides of the face.

YafaRay materials

The setup and use of YafaRay materials works in a very similar way to the lights in Blender, where we have to create the object in Blender and change the settings in the exporter script. In the **Material** panel in YafaRay, we won't create any materials, but take materials created in Blender and add the parameters related to YafaRay to add advanced properties, such as transparency, to the surfaces.

The first step to set up materials in YafaRay is to select an object that already has a material added in Blender. If we open the exporter and go to the **Material** tab, we will see the **Material** options, such as the ones shown in the following image:



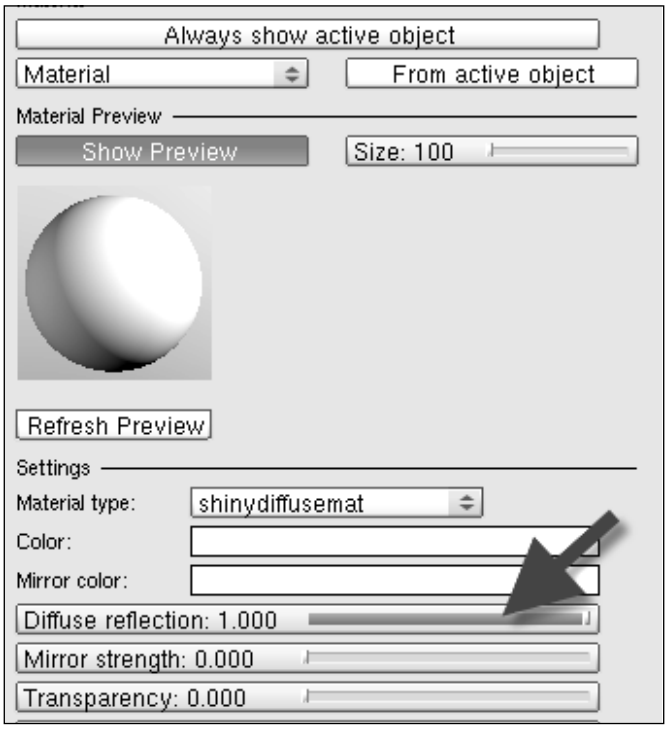
In the upper part of the panel, we will find the **Material** selector and the preview window. By default, only the materials from the selected object will be displayed on the panel. When the material is selected, we can start to work with it by choosing the material that will be used in our object. There are five types of materials available:

- **shinydiffusemat**: This is the most useful material of YafaRay, which can be used for almost all surfaces. We can create mirrors, fabric, and many other materials with this option.
- **glossy**: With this material, we can create materials that have blurred reflections, including some plastic surfaces and some types of wood or waxed floors.

- **coated_glossy**: This material is the same as the **glossy** material, but with an extra layer of a reflective surface above the blurred material. It's a great option to simulate metals such as car paint.
- **glass**: As the name suggests, we can create all sorts of transparent materials with this material.
- **blend**: This material can mix two different materials to create another one.

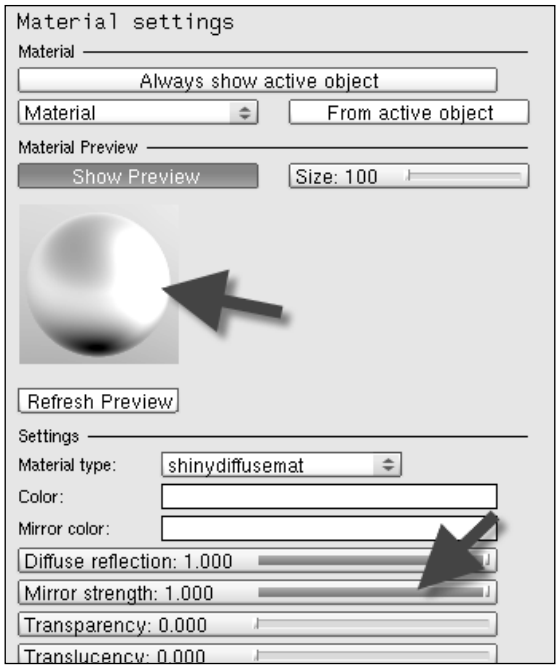
Creating diffuse material for a wall

To create a simple diffuse material for a wall, we have to only set up a **shinydiffusemat** material with the **Diffuse reflection** set to **1**. This slider controls the amount of light that is reflected by the surface:



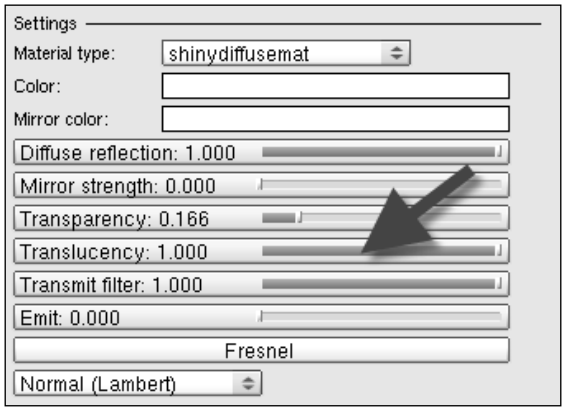
Creating a mirror

Using the **shinydiffusemat** material, we can create a mirror surface with the **Mirror strength** slider. If we set the slider to the maximum value, the surface with the material will reflect all light and behave like a mirror:



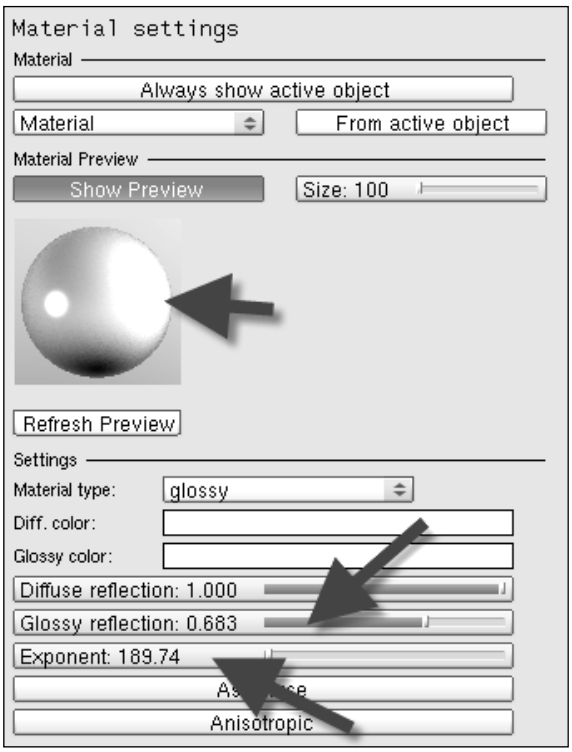
Creating semi-transparent fabric

If the material to be used is similar to transparent fabric in curtains, we could achieve this effect by using both the **Translucency** and the **Transparency** sliders:



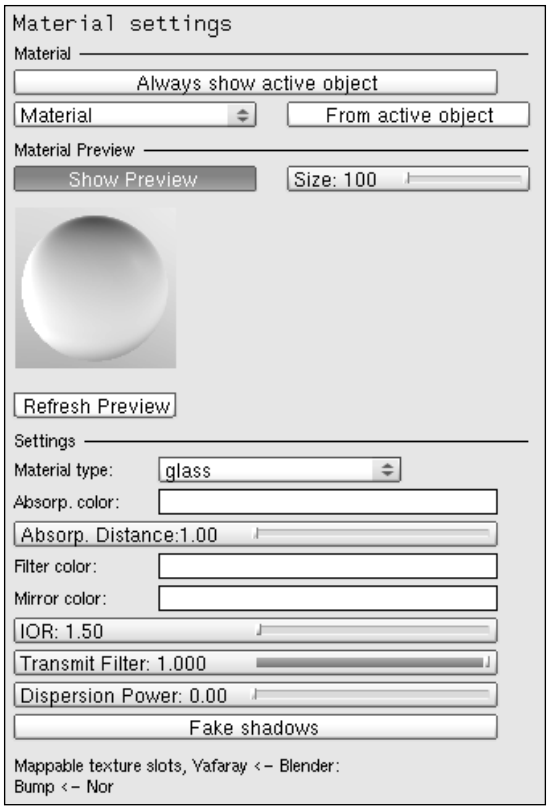
Creating blurred reflections

To create blurred reflections and materials, we should use the **glossy** material and sliders to control the amount of the reflection. The **Glossy reflection** parameter will set up the strength of the reflection, and the **Exponent**, the strength of the blurred effect:



Creating glass

One of the most common material types in architectural visualization projects is **glass**. To create **glass** in YafaRay, we choose the **glass** material and set up the type of **glass** that we want in the project. One of the most important parameters of the **glass** material is the **IOR**. This is the index of refraction, and will set up how light will be used to change in velocity and direction once inside the transparent material:

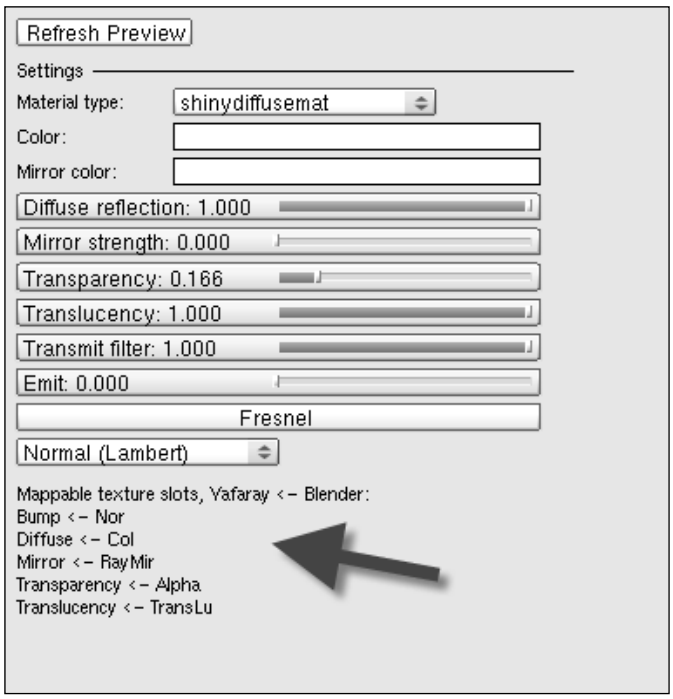


Here are a few common parameters for **IOR** for transparent materials:

- Water: 1.33
- Glass: 1.5
- Plastic: 1.4

Using textures with YafaRay

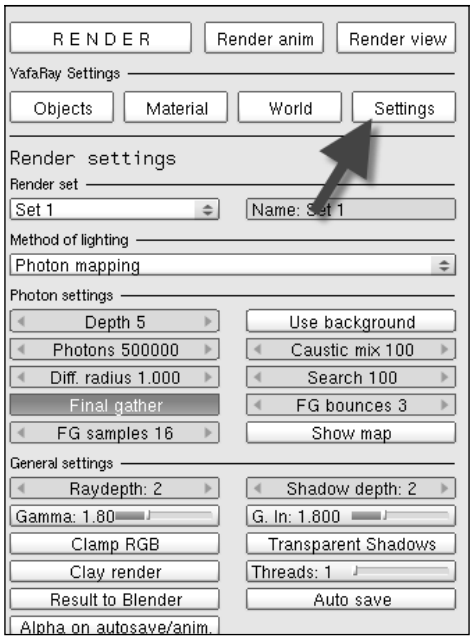
The textures setup in YafaRay comes from Blender parameters and tools. In fact, for each material in YafaRay we have a small list at the bottom of the panel showing how to map the material with Blender:



For textures based on bitmaps, the secret to making them appear in YafaRay is to add them using a UV Map. If the texture is not applied to the surface using a UV Map, it won't appear in the YafaRay render.

YafaRay render methods

The actual rendering process that uses global illumination techniques is set up in the last panel of YafaRay named **Settings**. This is where we can choose from four different methods to start rendering our architectural visualization projects. Each method has weak and strong points that make them more useful in different situations. For architectural visualization projects, we often use only two of the four methods available in YafaRay, because they will render scenes with the best quality in both of the most common types of visualization for architecture, which are the interior and external views:



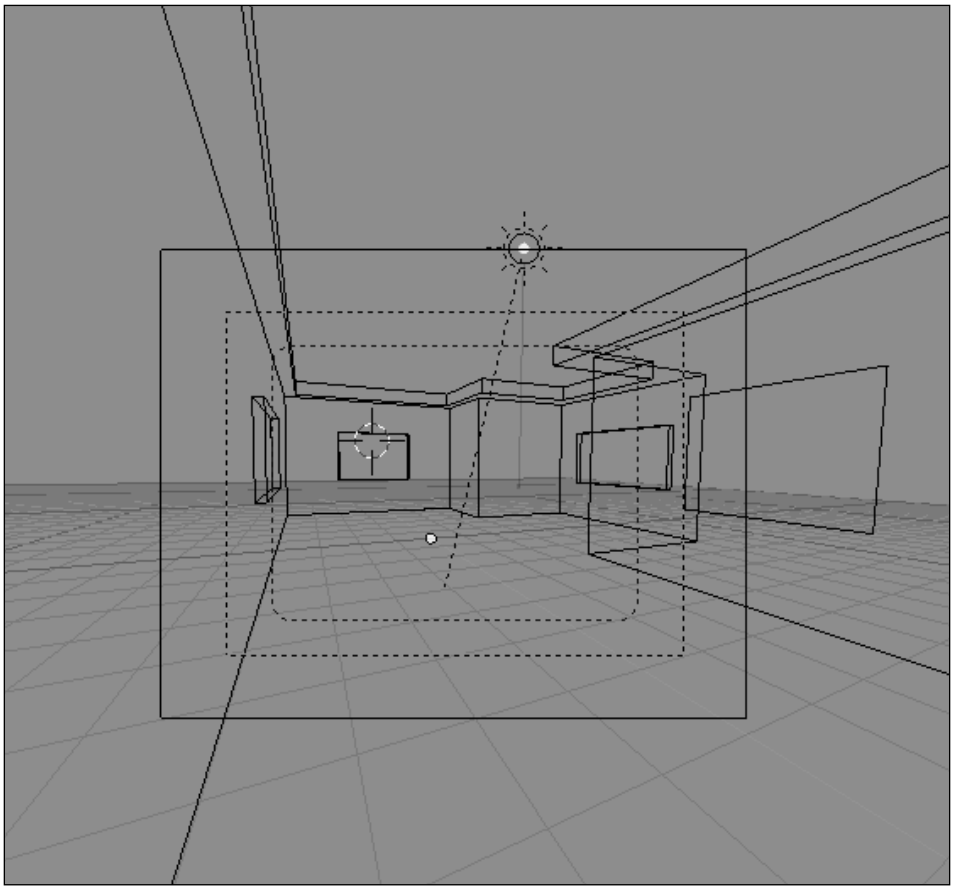
The two methods used in architectural visualization projects are **Pathtracing** and **Photon mapping**. The other two methods aren't used in architectural visualization, because one of them has only the basic features of raytracing already available with the Blender internal renderer, which is **Directional**. The other method, which is Bidirectional is still experimental and can produce strange results in complex scenes, which is the **Bidirectional**.

In the rest of the chapter, we will take a look at two scenes and learn how **Pathtracing** and **Photon mapping** can help us achieve a good light setup for architecture.

Rendering an interior scene

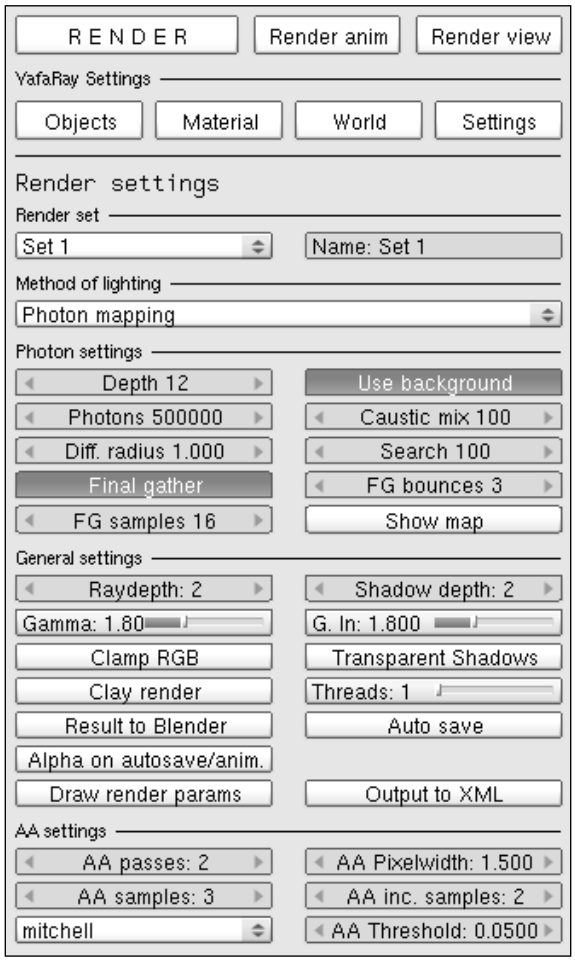
For interior scenes, the best option is the **Photon mapping** technique that generates light by casting rays from all light sources on the scene. When a lot of rays hit the surfaces of the scene, they leave a **Photon mapping** that is used to generate the light. Because it needs the surfaces to create the mapping of photons, indoor scenes can generate the best results with fast mapping generation.

For instance, let's examine a scene like the one shown in the following image. Notice that we have a light source made by a sun lamp:



This scene is perfect for the **Photon mapping** method because it is an indoor scene, and made with lots of walls and surfaces. A map of the photons cast by the light sources in this scene would be created really fast.

Here are the settings used to render the scene:



The settings used to create the render will set up how the rays will be cast from the light sources, and how many times they will bounce over each surface:

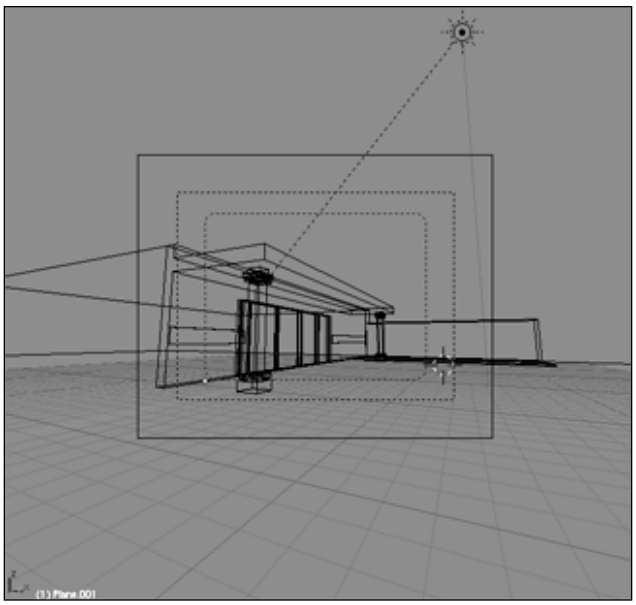
- **Depth**: How many times the light rays will bounce.
- **Photons**: Number of photons that will be used to create the **Photon mapping**. If the lighting of a scene is too noisy, a way to improve the light is increasing this value.
- **Final gather**: This is a technique used to improve the **Photon mapping**.

The final result for this scene can be seen in the next image:

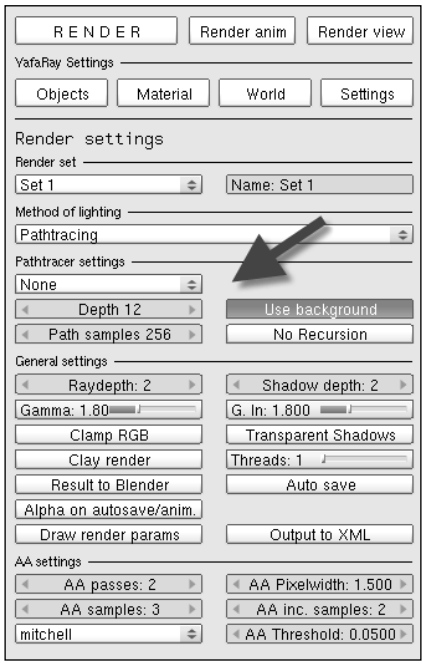
Rendering an external scene

For external scenes, the best option in YafaRay is the **Pathtracing** method, because of the way it works to generate the scene. The light rays are cast from the camera and will bounce until they reach a light source on the scene. For indoor scenes, this could be a very long time, especially with small light sources. But, for outdoor scenes with the background acting like a light source, it can be a really fast process.

Let's take a look at a render created with **Pathtracing**. For that, we will be using the scene shown in the following image:



Following are the settings used to render the scene:



Compared with **Photon mapping**, the settings available in **Pathtracing** are quite simple to choose. We have an option to choose the amount of times that each light ray will bounce until it finds a light source in the **Depth** parameter and the sampler that determines the render quality. A high number of samples will generate a less noisy image.

The final result of this render is showed in the following image:

Summary

In this chapter, we have learned how to improve our images with the advanced features offered by YafaRay. With these features, we can produce images on surfaces using global illumination options, such as photons and light reflection.

Here is a list of subjects learned in this chapter:

- Choosing YafaRay as render engine for Blender
- Setting up the basic parameters of YafaRay
- Using the YafaRay global illumination settings
- Setting up some of the materials available in YafaRay
- Choosing the best method for indoor and outdoor scenes in YafaRay

13
Animation for Architectural Visualization

From the beginning of the book until now, we have only worked with static images. But Blender can offer us a lot more than static images, if we start to work with its animation features. Showing a project with an animation makes a very strong impression on any person, especially for selling purposes. And Blender can offer us an extra feature, which no other 3D package can—interactive animations. Yes! With the integrated Blender Game Engine, we can create interactive animations. This enables the people who have to validate your project walk into all environments, and have a full virtual experience of your project.

This is a great feature and tool for anyone working with architectural projects. Additionally, we have the standard features in Blender to work with linear animations. So, let's get started with the animation basics and go further into how to make small movies of your projects.

Animation

When we have a project with an animation, and want to show it to someone, the best solution to produce this animation is a walk-through. This is a very simple type of animation, where we have to work only with the camera position. In this animation, the camera goes through the entire project, simulating a walk into the building or environment.

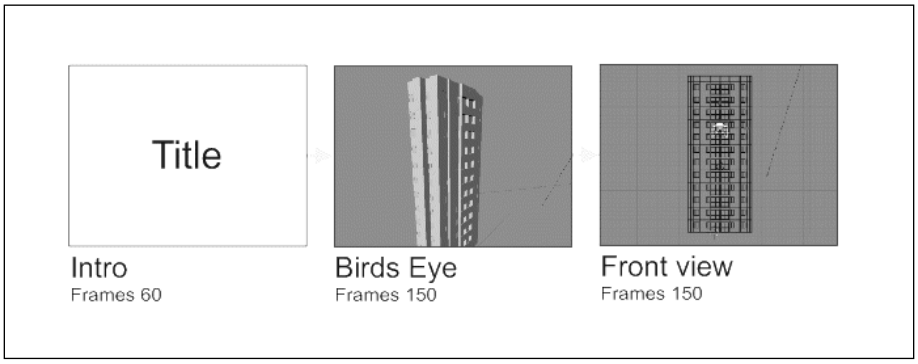
Because we work only with the camera, there won't be any requirement to learn or work with more advanced animation features, such as armatures or bones, which are used to make character animation.

Even though it's simpler, we have to take some time before the production of any animation, to prepare and plan all steps to avoid any troubles or rework. Planning an animation can really save you a lot of time and if well executed, can even show you how to improve or even not to go ahead with the animation.

Planning the animation

The first thing to plan for the animation is to make a small storyboard, or to trace the path of the walk-through, where the camera should pass. It's important to know exactly what you want to show in the animation, because, as you will see, the time required to build an animation and the long render times required, make an animation a very expensive type of media for presenting a project.

After choosing the path where the camera should be moving, the next step is to identify the places where the camera should stop or turn around, to show a particular area of the project. This can be done on a paper or any type of media where you can make some notes. A storyboard can help too, but it's not very useful to make a storyboard for an animation aimed at an architectural visualization:



Animatic

The basic plan of an animation is great to lead you to exactly what you need to show, for your client or the development team at the office. But there is something that can't be predicted with the planning done with a storyboard or 2D project. It is the timing for the animation.

In addition to the technical challenge, setting up the right timing for the animation can be a very difficult task. A very common mistake for artists who don't have experience with animation is to try to guess the right timing in the 3D View and not in the rendered animation. Sometimes, the artist sets up the animation and waits for some hours to finish the render, only to discover that the timing is not right. The animation is too slow or too fast.

Don't be afraid to spend a lot of time fine-tuning the animation. It's perfectly normal, and it probably will happen with you. Finding the right timing for the animation can take more time, even more than light setup. Just as with light setup, with the proper time and practice, you will find the solution for the timing in a shorter time.

A good solution to avoid this kind of mistake is to make an animatic! This is a very simple type of animation. It's exactly the same animation that you will create for the main animation, but with simpler shapes, lights, and textures. Because the objective here is to validate the timing only, no other visual aids are required.

Here are a few tips on how to build an animatic:

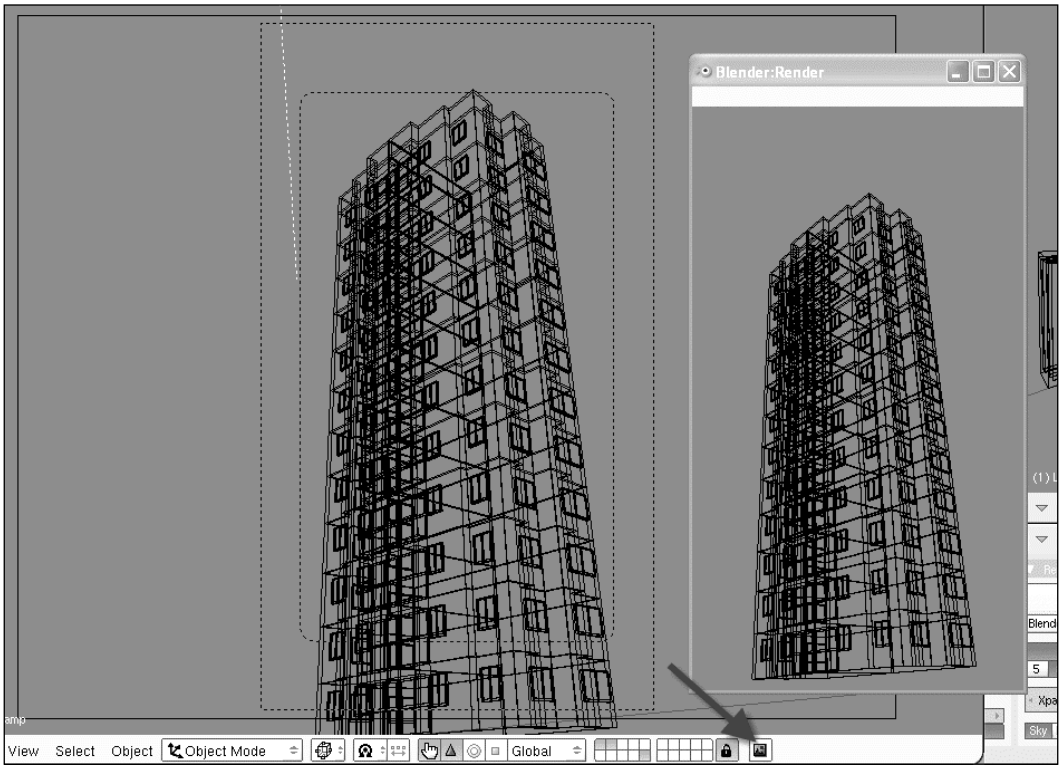
- Replace all models on the scene with primitive shapes such as cubes
- Remove or decrease the light quality
- Don't use global illumination
- The animation for the camera should be exactly the same as the final render
- Remove all textures or advanced materials from the objects, such as ray traced reflections

With these guidelines, a very simple animation can be created. In addition to the simplicity, there is another important thing about the animatic—the render time required to make one is very short.

Because the render time is short, if the animation requires any type of adjustment, you will be able to make the camera move faster or slower, and validate it again with the same speed. And, for the final render, there is big chance that everything will work exactly the way you want.

A great way to create an animatic is with the **Render this window** button. It's located on the right side of the 3D View header. If we hold down the *Ctrl* key and press this button, a very simple render will be generated. The *Ctrl* makes Blender render an animation.

The animation will be generated with the same view type as the object is shown in the 3D View. For instance, if we set up the object to be displayed as wireframes, the render window will be generated like this:



Save time with the planning

Some people think that creating these types of pre-production animations or plans are a waste of time. They go straight to the animation production. The point of doing this preparation is to save time, and not be surprised after a long render with a bad camera move or too slow a movement. Always pass through this preparation, and there is a strong likelihood that you won't have to rework the animation ever again.

Animation and frames

How does an animation work? Before we start to create animations with Blender, let's understand how the process of creating a linear animation works. The result of an animation created with Blender is a video file, such as an `AVI` or `MOV` file. For interactive animations, we will create an executable file.

To make a video file, a lot of still images are rendered and played in order. That's the reason which makes an animation so time-consuming. If we think that one single frame can take something around thirty minutes to render, an animation with fifty frames can take about twenty-five hours to render! Now, do you understand why it is so important to plan the animation?

How many frames are required to make an animation? It depends. In most cases, for a smooth animation, we use a rate of twenty-four frames for each second. That's the frame rate.

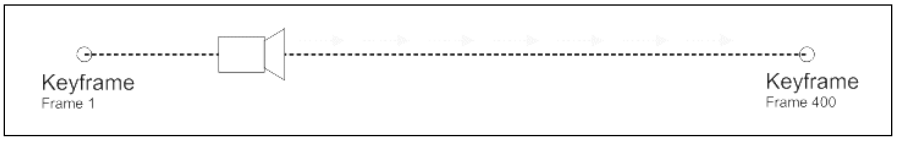
Here is a list of common frame rates used for animation:

Animation type	Frame rate (FPS)
CG Animation	30 FPS
Film	24 FPS
Animatic	10 FPS

Keyframes

If we want to create any kind of animation, we must add and manipulate keyframes in Blender. To understand what a keyframe is, we have to think about how the animation is formed. All animation is made of a series of frames, rendered in order to create a motion image. When a series of images are played at a rate of twenty-four frames per second, we will have the appearance of motion.

With keyframes, we can set up the properties of an object at a specific point in the animation. For instance, we can set up a camera to be at the entrance of a house in **Frame 1** and then outside of the house, a hundred meters away, at **Frame 400**. What will happen when we hit play? Because the camera is placed in different positions and frames, the animation will automatically move the camera between the two positions:



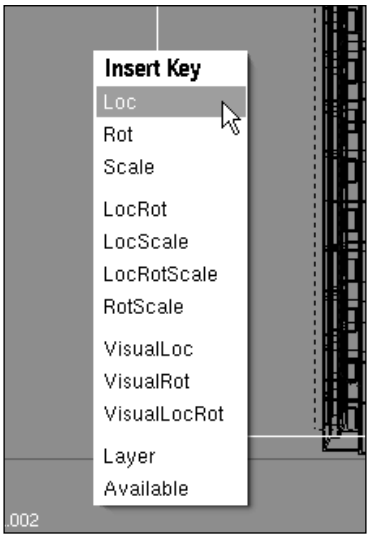
We won't have to do anything else; it will all be handled by Blender. All we have to do is to set up where the keyframes should be, and the properties of the object. For us, the main properties that we have to set up are the position, rotation, and the scale of objects.

Interactive Animation

For interactive animation, it won't be necessary to set up keyframes, because all animation will be interactive. Because it is interactive, the user will decide where the animation should go and what the camera should be seeing.

Creating keyframes

To create keyframes in Blender, we have to select the object which will receive the keyframe, and press the *I* Key. When we do that in the 3D View, a menu will appear for us to choose the proper keyframe type:

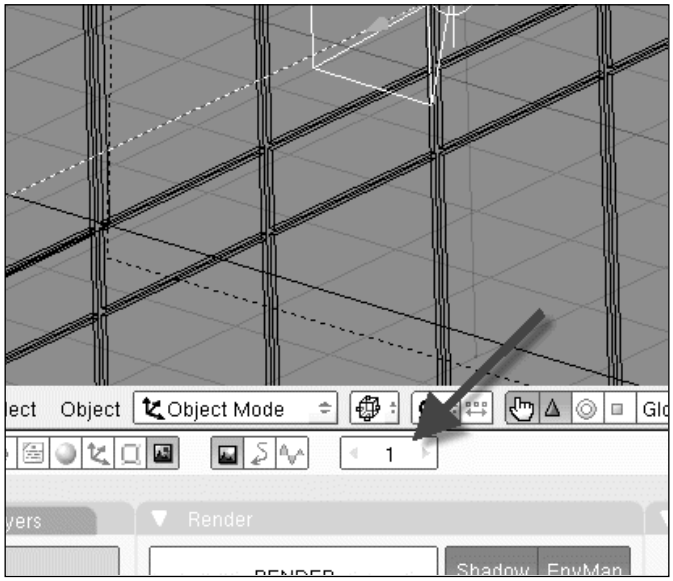


For each type of transformation or even attributes of an object, there is a specific keyframe. Let's see what some of them mean:

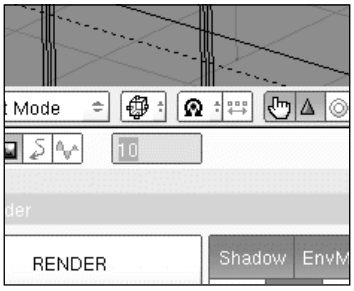
- **Loc**: Keyframe for position
- **Rot**: Keyframe for rotation
- **Scale**: Keyframe for scale
- **Layer**: Keyframe for layers

There are some options that blend different keyframes as shortcuts to mark properties for **Loc**, **Rot**, and **Scale** at the same time. For instance, if we choose the **LocRot** option, a keyframe for the position and rotation will be added.

Before we add any keyframe for an object, we have to place the playback head at the desired frame where the keyframe should be placed. To do that, just use the navigator pointed to in the following image:



For instance, if we want to add a keyframe for an object in frame 10, then we have to move the playback head to this position. To move it, just click and drag the number, or click once and type the number of the frame:



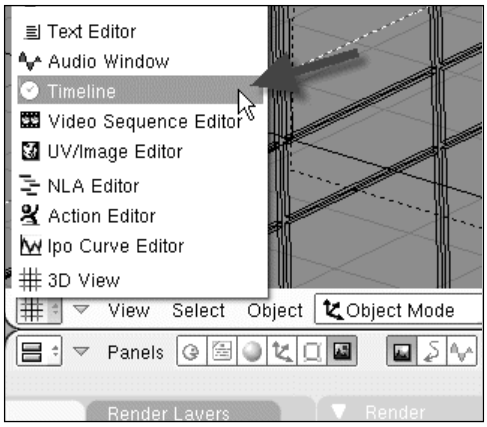
When the playback head is in the right position, press the *I* key, and choose the keyframe type. Always remember to place the frame before adding the keyframe, because it can alter the animation or the way all properties of your object change. A very common mistake is to add keyframes without changing the position of the animation. It will cause the keyframes to be added at the same position, which will not make an animation.

For a camera animation, always use the following sequence to add keyframes:

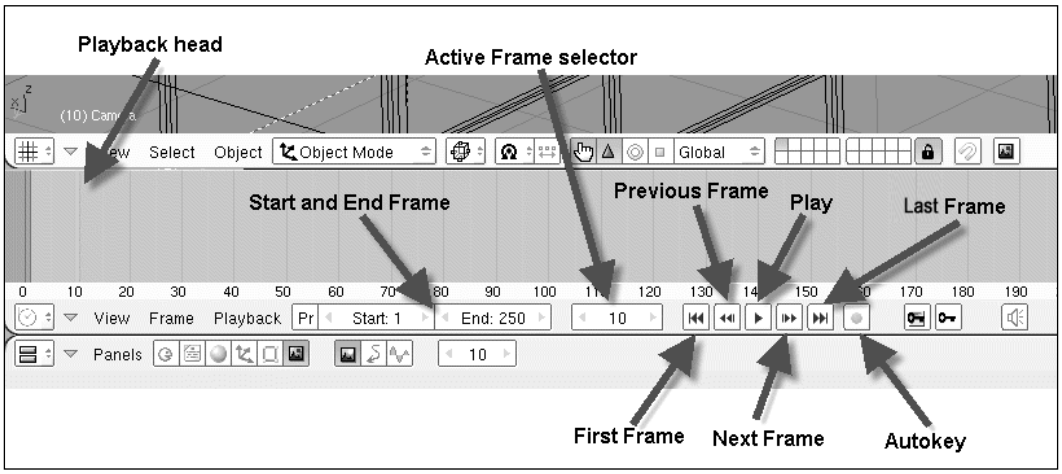
- Place the playback head at the right position.
- The second step is to edit and adjust the object. For instance, alter the size, position, or orientation of the object.
- After the previous two steps, add the keyframe to the object to build the animation.

Timeline

There is a window named **Timeline**, which was created to help with the animation process. With this window, we can easily add, move, and change the timing of animation. To make the process even easier, we can open and adjust the window to make it appear on the interface horizontally:



The organization of the **Timeline** is very simple, as the next image shows:



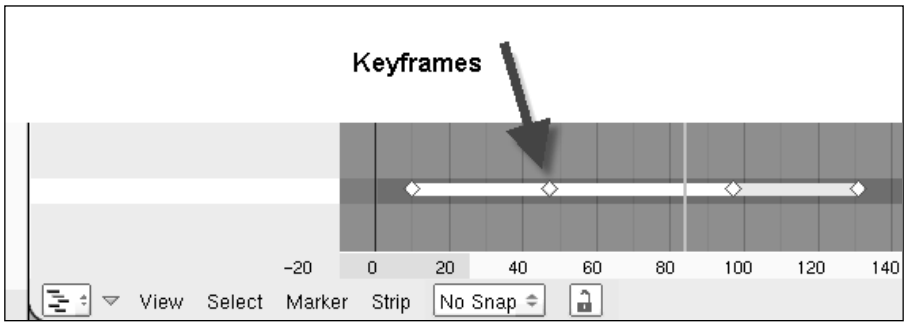
- **Start**: With this option, we can set up the frame where the animation will start.
- **End**: Here we can set up the frame where the animation will end.
- **Play**: Press this button to start the animation.
- **Next frame / Previous frame**: These buttons allow us to navigate through the animation, frame-by-frame.
- **First frame / Last frame**: If we have to jump to the beginning or end of the animation, we can do that with these two buttons.
- **Automatically insert keyframes**: Here we have one of the most useful options for the timeline. When this button is turned on, the keyframes will be added automatically when we change a property of an object.

Use the **Autokey** option to make animations faster. Just turn it on and make the changes required to an object. For instance, place the playback head at the required frame, and then change the property of the object. The keyframe will be added automatically. For the next keyframe, change the frame to the next position and then change the property of the object again.

With this sequence, you will create a complex animation in a short period of time. Always use this window to help in the process. But even with this window, we will have to make some adjustments later, because the motion or sequences won't always be created the way we want.

Managing keyframes

Sometimes we may want to manage the keyframes used to build an animation, such as moving, erasing, or duplicating these keyframes. To do that, we can use a window named NLA Editor. There we can select each keyframe and do almost anything that we can do in the 3D view with 3D objects. For instance, we can move, rotate, and erase a keyframe:

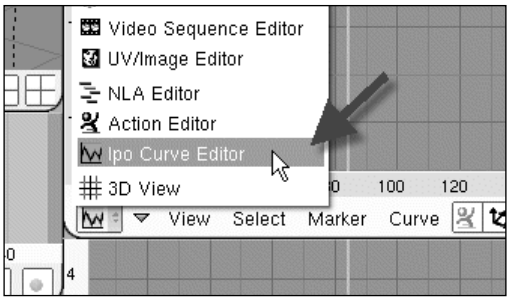


The process of doing that is very simple, and uses the same shortcut keys used for the 3D view. Of course, the only thing that we can't do in the NLA Editor is a rotation:

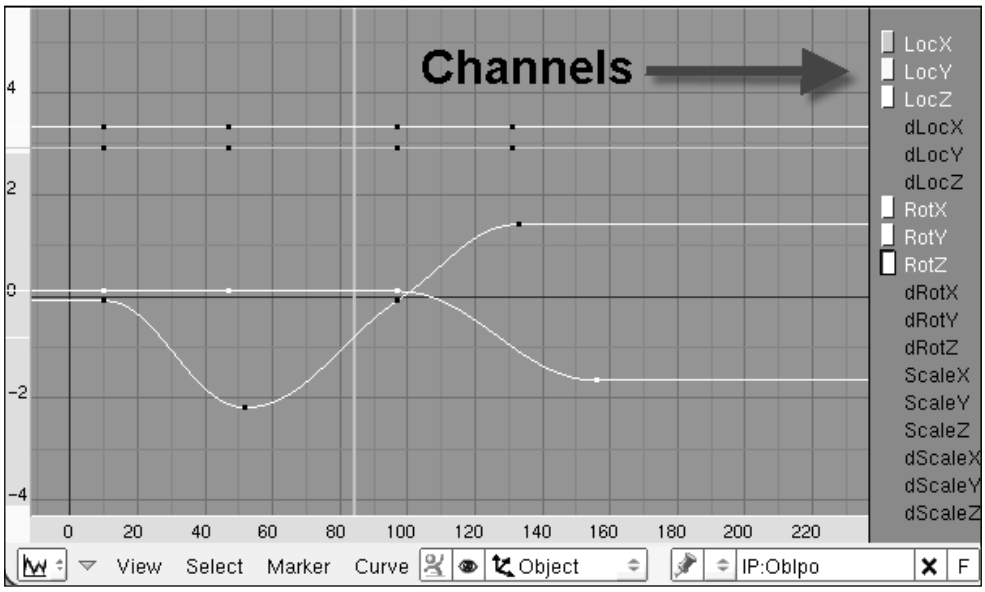
- Right-click: Select a keyframe. If you want to select multiple keyframes, just hold the *SHIFT* key while you click on the keyframes
- *G*: With this key we can move the keyframes
- *S*: If we select one or more keys, we can scale them with this shortcut
- *X*: Just select the keys that you don't want anymore, and press *X* or *Del* to erase them

IPO curves

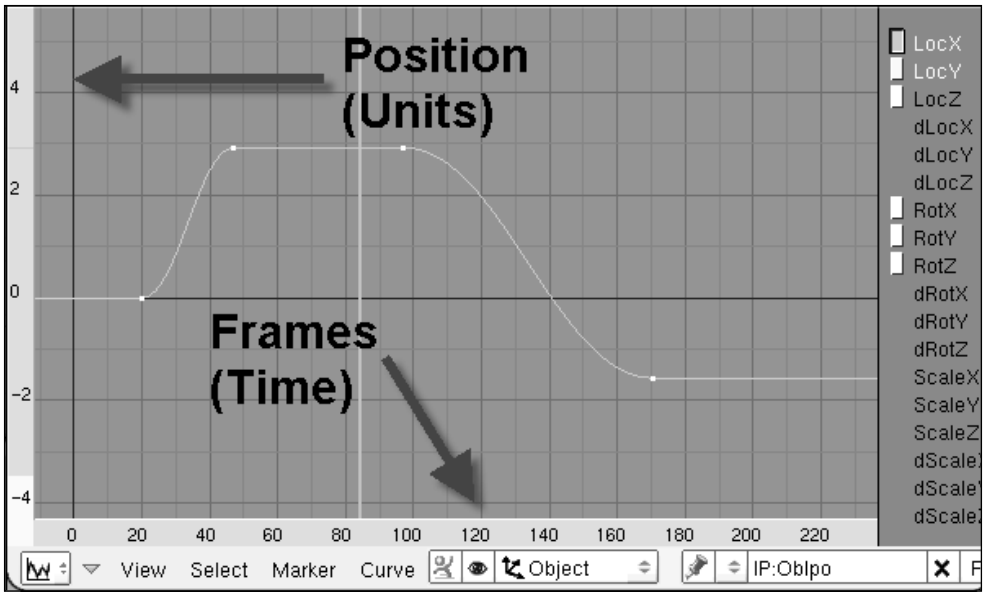
When we edit our animations, sometimes we will have to fine-tune the motion or adjust the rotation speed of the camera. The nature of the motion generated by keyframes is based on curves, and not on straight lines. If we want to edit this kind of animation, we have to use a special kind of window named **Ipo Curve Editor**. With this window, we will have total control over keyframes and the curves over time, because we see the animation in a graphical environment:



This graphic allows us to choose several animation channels, which contain their own curve types. All those channels are placed on the right side, and we can add curves and keyframes to almost any property of an object. To select a channel, just click with the right mouse button over the name of the channel, and the curves for this property will be showed in the curve editor:



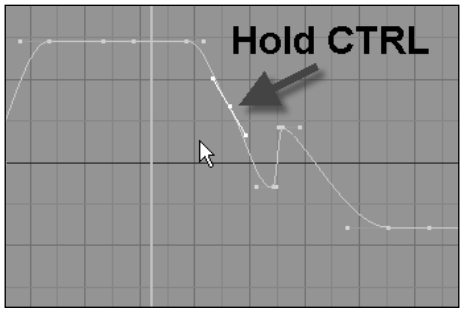
The curve is shown with the display of time on the horizontal axis, and the property value on the vertical grade. For instance, when we select the **LocX** channel, the curve for the motion of an object will be shown with the frames on the horizontal axis and the location on the x axis on the vertical grade:



Showing multiple curves

If you want to show more than one curve, just hold the *Shift* key while you click on the channel's names.

If we want to add new points to the curve, hold the *Ctrl* key and left-click on the place where you want the new point to be added. We can do that as much as we want, and the new points will be added:

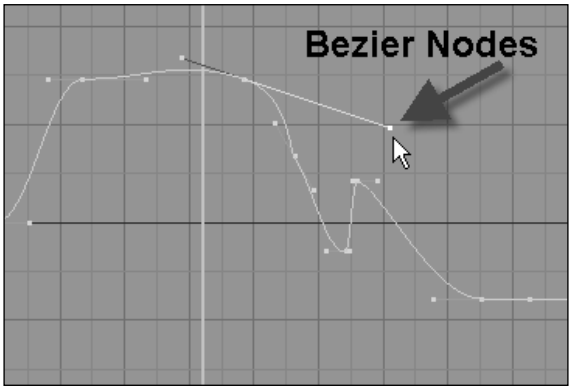


Editing the curves

To edit a curve, we enter an Edit mode for the selected curves. If we select one channel and press the *Tab* key, the nodes of the curve will be shown as Bezier nodes, and we will be able to move and edit the curves. Only one curve at a time can be edited. If you have multiple channels selected, only the active curve will be edited, which will be the last one selected.

The curve can be edited with the same shortcuts used to edit objects. With the *G*, *R*, and the *S* keys, we can move, rotate, and scale the curve points. Hold the *Ctrl* key to add more points.

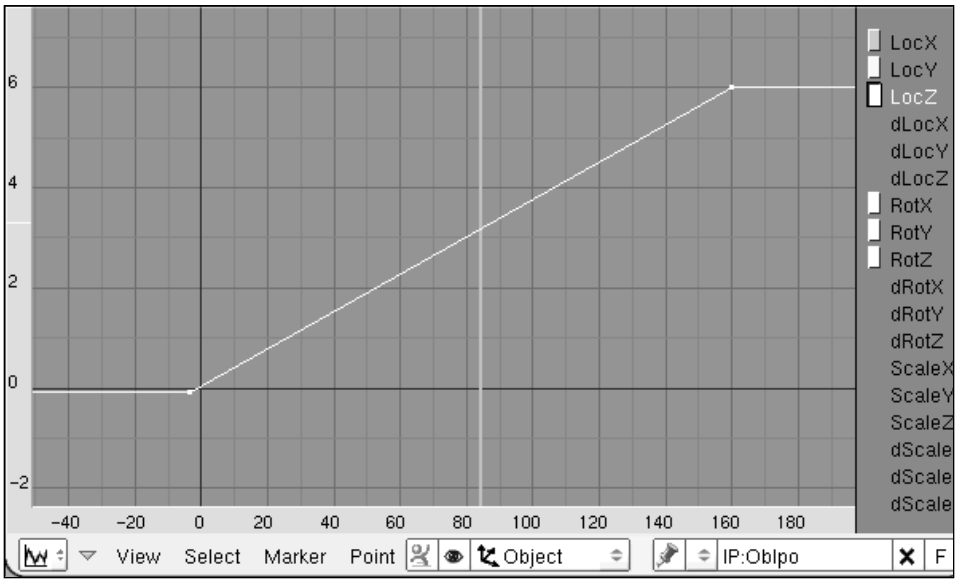
If you need more precision, such as putting a node at a specific position, press the *N* key to open a property menu. It will allow us to place a node at a specific position on the curve editor:



Using curves

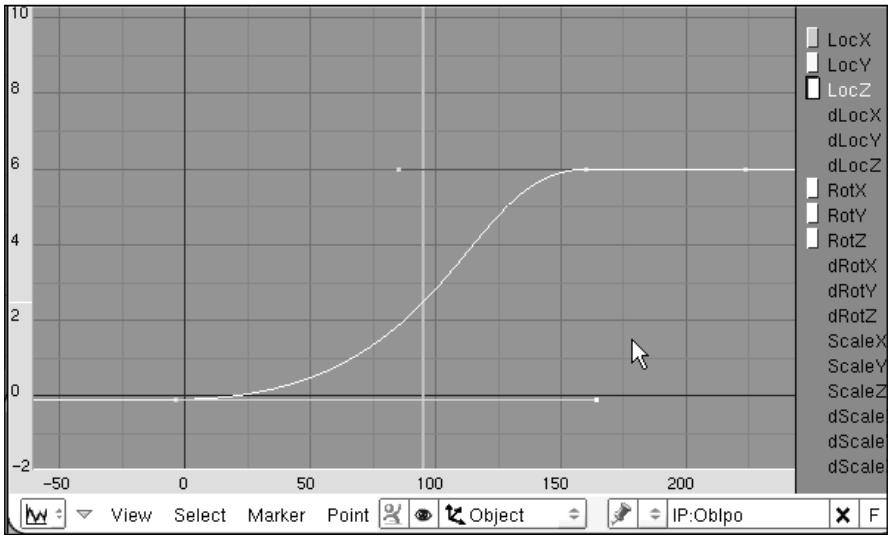
You may be asking yourself, why should I use these curves? Well, there are a lot of uses for these curves. An animation may require a more sophisticated movement than a simple linear translation. If we want to add ease in or out on the movement, we have to use curves.

Let's see an example of that. Take a look at the following curve for a linear motion:



This curve will generate a linear motion with no acceleration or changes in the motion. If you ever have seen this type of animation, you will know that this kind of motion is very artificial and doesn't give a natural feel for the animation. We have to add some ease of the movement. It can be done with the curves.

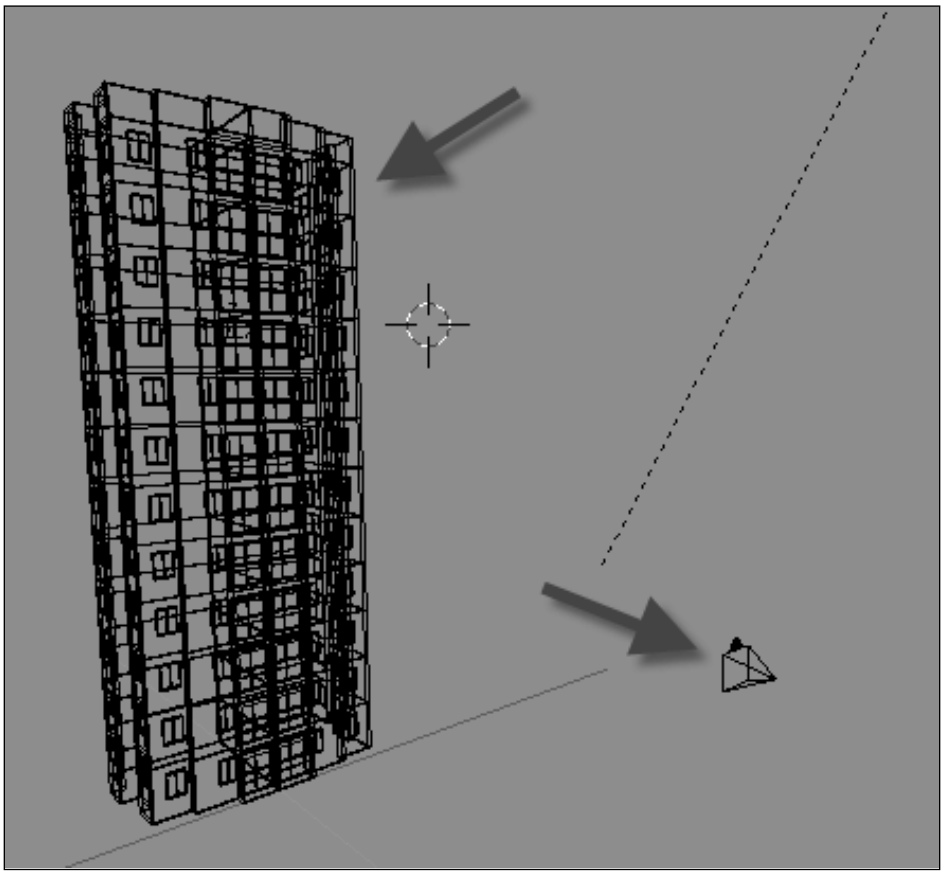
Let's change the curve, so that it looks like the following one:



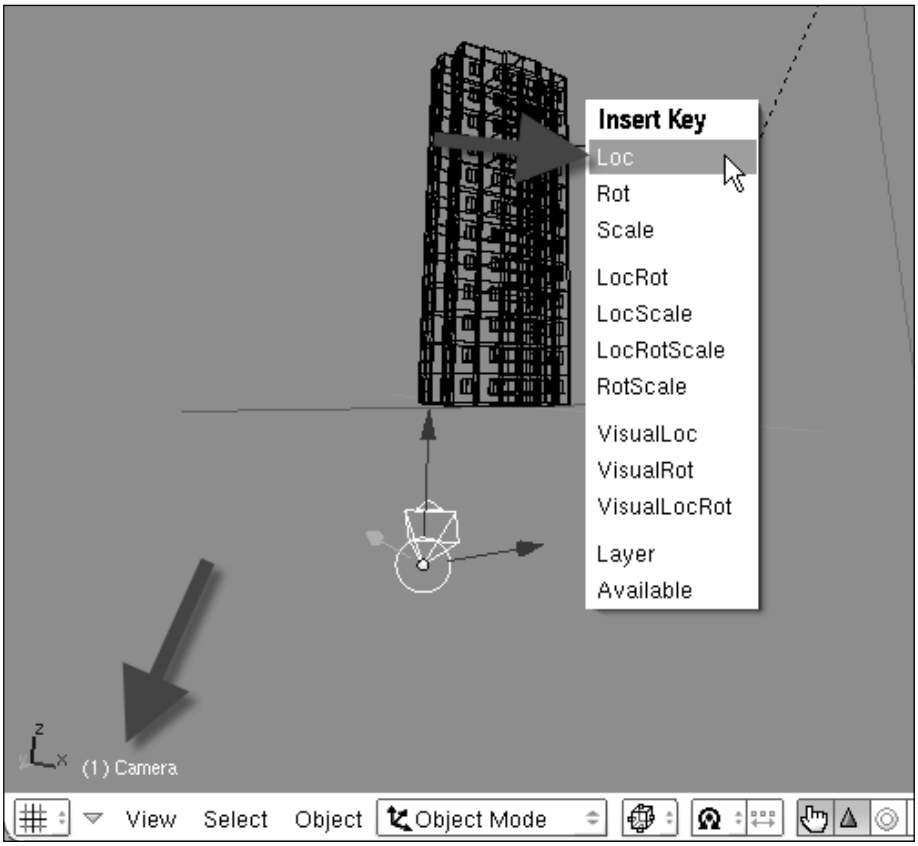
The motion of the object or camera will present a small decrease in the speed of the motion at the end of the animation. It's an effect named ease out, which will create the sensation that the camera is slowing down the motion as it gets closer to the destination. It's a more natural feeling than just linear motion.

Animating a camera

The most common animation that we will create for architecture is camera animation, which consists of using the movement of a camera to show a building. To create an animation such as this, we must have at least one building and one camera to start:



Place the camera where you want the animation to start, and add a keyframe. If the camera will only translate, add a **Loc** keyframe:



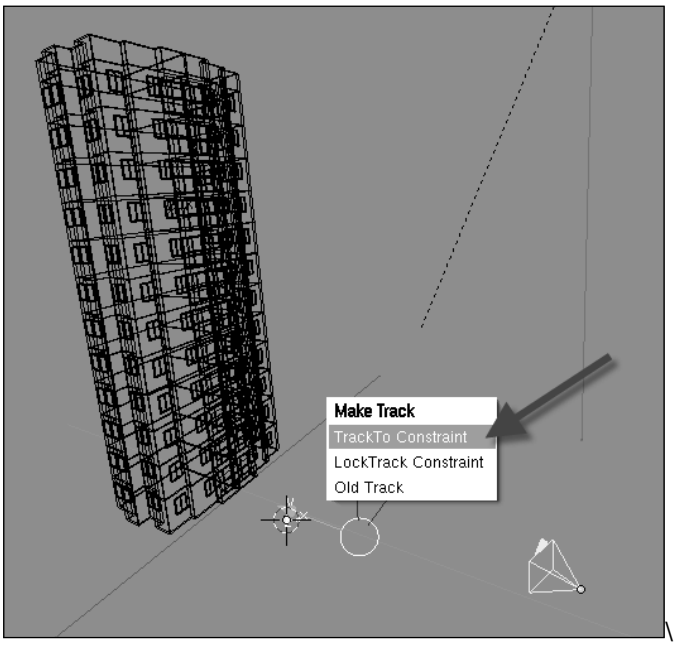
Change the current frame to the next keyframe position. If the camera translation requires 2 seconds, use 60 frames to get to keyframe 2. So, change the current frame to **60**, and place the camera where you want, and add another **Loc** keyframe.

And it's done! Just place the mouse over the 3D View and press *Ctrl* + *Alt* + *A* to play the animation.

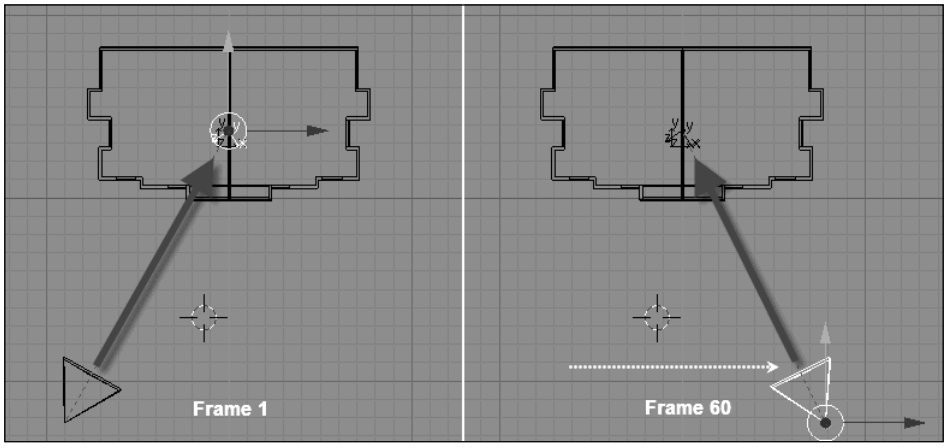
Adding a target

Animating the camera may be a difficult task without the use of a target object, especially if the camera has to be aimed at a specific part of the project. In Blender, we can add an **Empty** object as the target of the camera, and make the camera always look for this object.

Add an object named **Empty**, from the toolbox. Then, select the camera first and **Empty** last. When the two of them are selected, press the *Ctrl* + *T* to add a **TrackTo Constraint** to the camera:



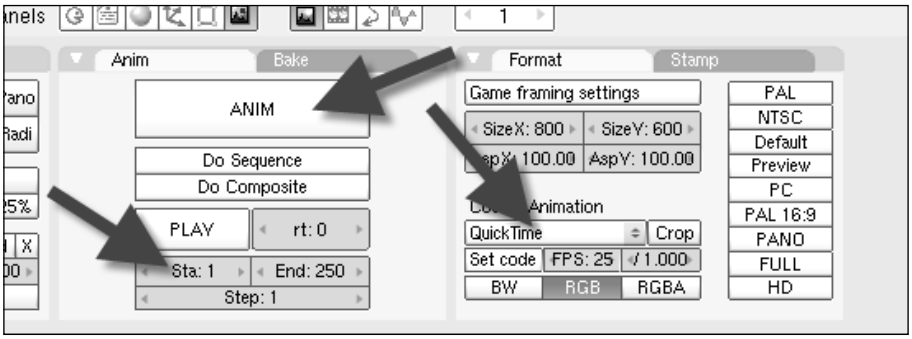
It will make the camera to always look for the **Empty** object. Now, we can animate and manipulate the camera in all directions, and it will keep focusing on **Empty**:



This tool works for all objects. Use it every time you need the camera to be aimed somewhere.

Rendering animation

The main purpose of making an animation is to create a video file, which requires a different approach to render the file. Before anything else, we have to know how many frames will be rendered to create the video. It's important, because Blender can only render animations with a predefined number of frames. To set up the start and end frames of an animation, we can use either the **Timeline** or the **Anim** menu in the **Scene** panel:

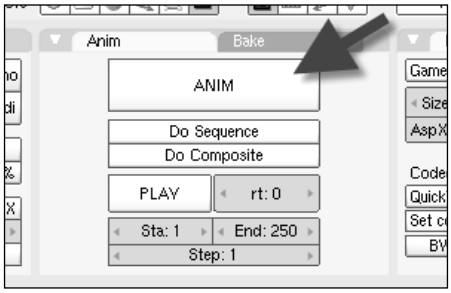


After we set up the start and end frames, the next step is to choose the file type. For most animations, we choose the AVI or MOV file formats. If you want to output the animation for a video format with a high quality, choose the MOV format. In addition to the format, we have to choose the compression format for the video as well.

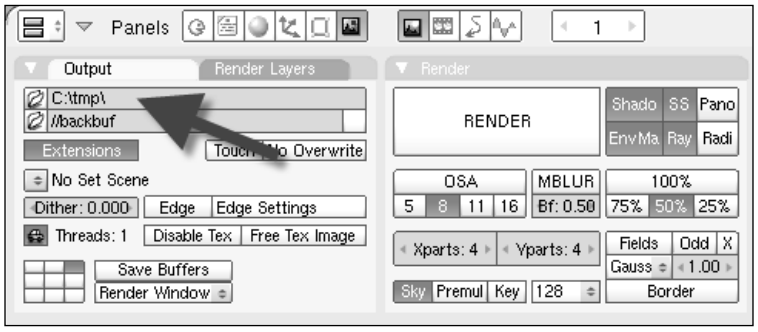
What's the best compression type? The focus of this book is not to describe how to compress video, but we can suggest some popular compressors:

- **MOV files**: For this kind of file, choose the H.264 or H.263 compression types. If you want to create a video file with the highest quality for post-production, just choose **none** to create a video with no compression. It will generate a huge file, but with no loss of quality.
- **AVI files**: With this file type, we can choose three different options for the AVI. The first one is named AVI Codec, which allows us to choose compression for the video. It's very useful if you want to publish the video over the internet. Just choose Xvid or Divx for a good compression rate. The next type is AVI Jpeg, which stores the files as a sequence of jpeg images. The compression is very good and it maintains the file quality. This type of file is primarily used to play animations inside Blender in the Video Sequence Editor. The last one is AVI Raw, designed for post-production, because it doesn't compress the video file.

After the file type is selected, the last step is to press the **ANIM** button. When this button is pressed, Blender will render the animation with all the frames that we set up:



If you don't like where the animation file is being saved, we can alter the default folder where Blender saves this file. Just select the location where you want to save the file in the place pointed to in the following image:

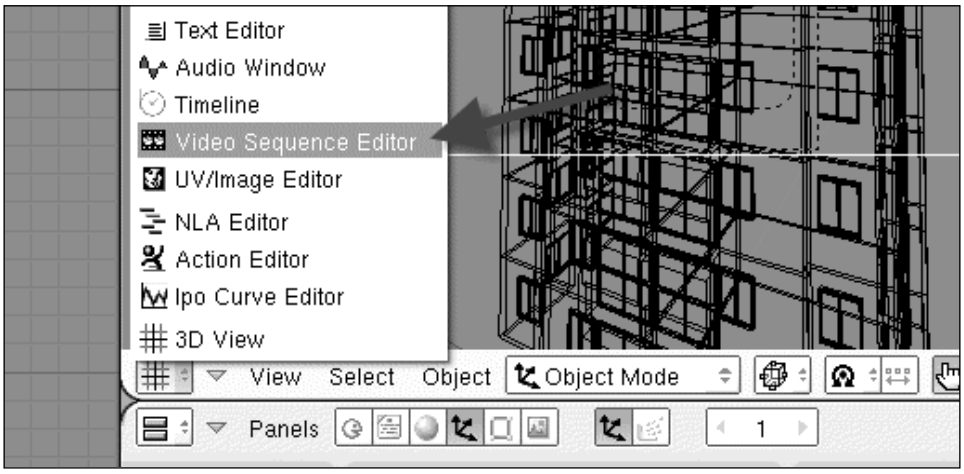


Video Sequence Editor

When we finish a project for an animation, the result sometimes is a set of video files. For each scene or shot, we render a new video file. Unless you want to select those files and play them individually, the overall project should be presented as a single piece. For that, we have to merge all video files into a single presentation.

For this task, Blender has a tremendous advantage, because it has a built-in video editor, which allows us to edit and merge these files without the need of video editing software.

To open the **Video Sequence Editor**, we select it with the Window Type selector:

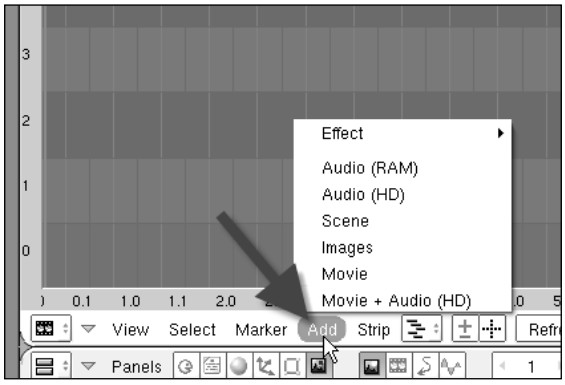


Editing video

Now that we know how to open this window, let's add and edit video files to finish our animation. For instance, suppose we have an animation made up by three video files, and we want to merge all those files into a single sequence.

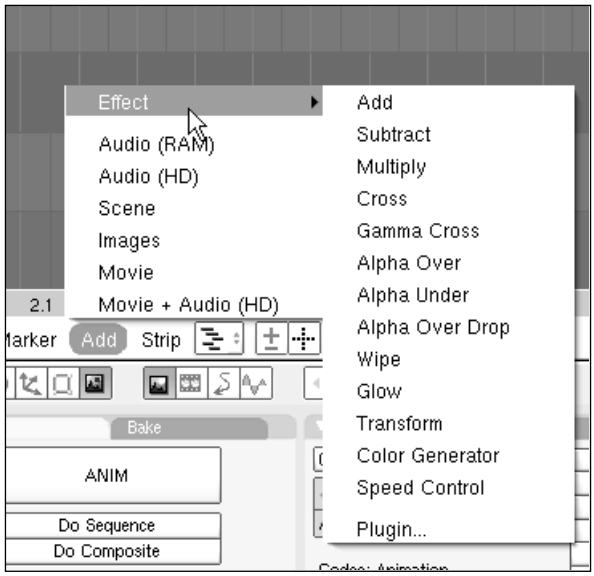
In this window, we can mix other types of media with the video files; the most common being audio files for soundtracks and images. With image files, we can create slideshows of renderings, adding a soundtrack and making a high quality presentation for our projects.

Let's take a look at how it works. Right after we open the editor, the next step is to add the media to edit. This is done with the **Add** option in the menu:



Here are descriptions of the media types we can use:

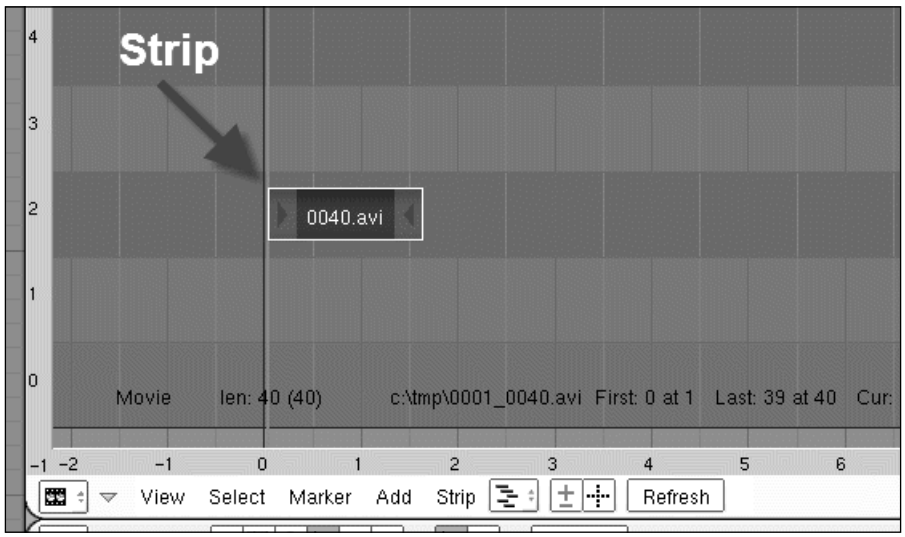
- **Movie**: With this option, we can select movie files such as AVI and MOV.
- **Audio**: Here we can choose audio files. For instance, we can choose a WAV file to add as a soundtrack.
- **Scene**: If you have separated your animation into scenes in the same Blender file, it will be possible to join them in the Sequencer. Choose this option, and then pick the scene by name. At the end, we will be able to render all scenes as a single animation.
- **Effect**: Along with the media types, we can add a few effects to make our animation look more interesting. For instance, we can add a PNG image over an animation, and make the background of the image transparent:



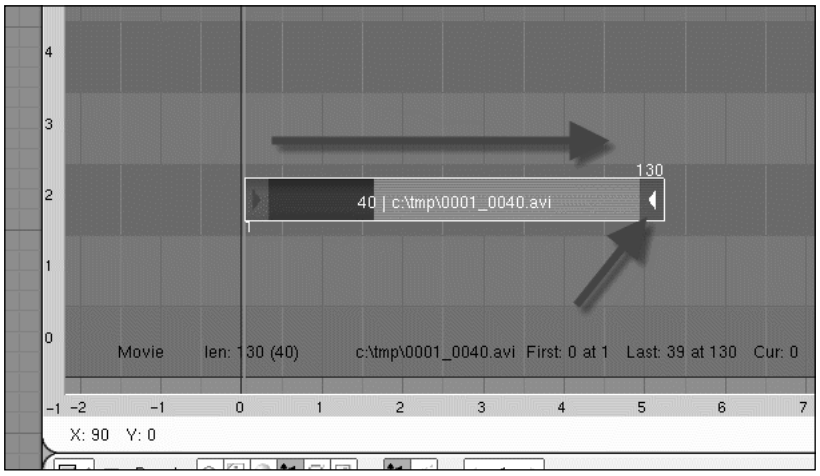
When we add a media type, such as a movie, into the sequencer, it's then named a strip. Right after choosing the file or scene, we will have to place the strip in the sequencer. To transform a strip, we use the same shortcuts we used in the 3D View. For instance, we can move a strip with the *G* Key. All the other shortcuts are the same ones used in the 3D View for:

- Select
- Transform

- Erase
- Edit

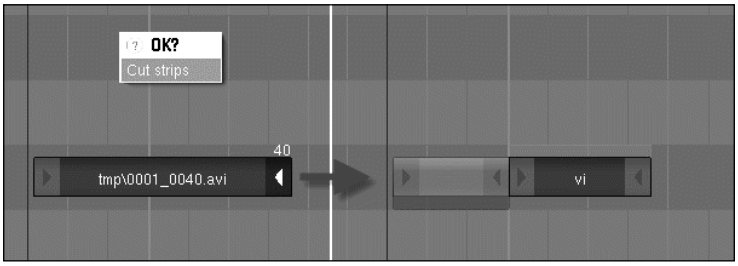


Just as in a 3D model, we can edit a strip. This is used to set up the timing for an image, because a movie already has a defined timing. It works this way; right-click on a strip to select it, and then right-click again on the left or right arrows of the strip. When one of the arrows is selected, press the *G* key, and move the mouse to change the timing of strip:



With the left mouse button, we can change the location of the playback head. This playback head is used to place the animation at a certain frame. If you want to split a movie into two or more parts, the playback head will mark the place used as reference to cut the strip.

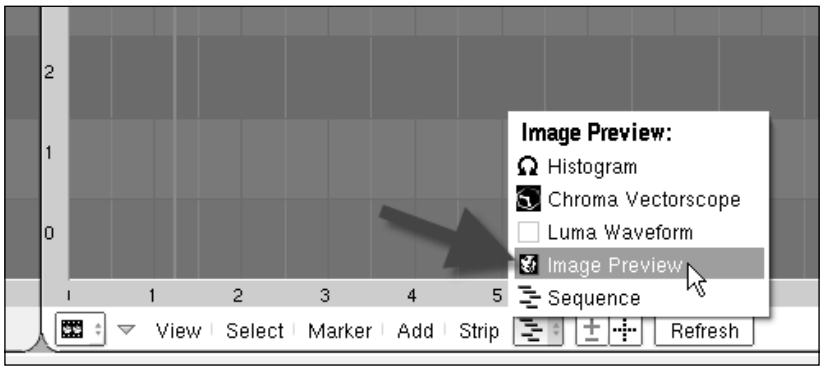
Select the strip, place the playback head where you want to cut the movie, and press the *K* key. It will split the movie into two separate parts:



On the left side of the sequencer, we can see a few numbers, which are related to the channels of the editor. They function as layers, where we can place a movie or image in front or behind another media file.

Preview the video

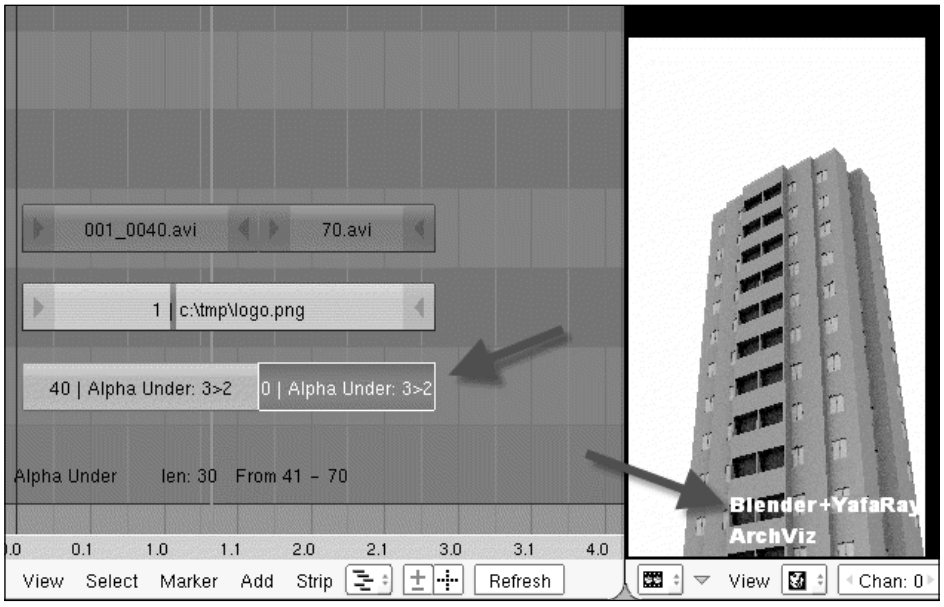
We can see a preview of the video while we edit the strips. For that, open a new division in the interface and choose the **Video Sequence Editor**. Then, we can choose the **Image Preview** option in the header. Now, we won't see the layers of the sequencer, but a preview of the current edition:



Effects

Now that we know how the sequencer works, we can edit our previous animation and make it into a single file. And to make things even better, let's add a watermark at the top of the video.

The watermark image has to be in an image format that supports alpha channels. For instance, we can use PNG or TGA files. For this example, we use a PNG file, which is added in a channel above the video file. When it's placed, select both strips and add an **Alpha Under** effect:

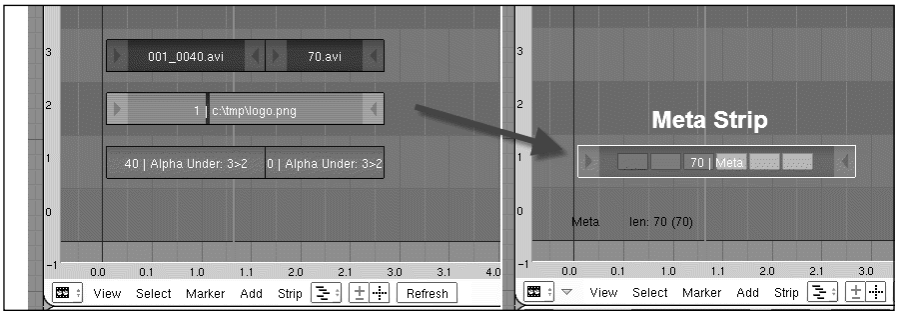


Other great effects for the sequencer are the **Glow** and **Wipe**. The first one makes the image brighter and the last one creates a wipe effect for slideshows and transitions. The Video Sequence Editor is really simple to use, but gives Blender a unique power to edit and manipulate video files that almost no other package provides.

Meta Strip

When we work with several strips in a project, a good practice to organize our workspace is to group all strips together. This way we won't get lost in a maze of video files. The group of strips is named a Meta Strip in Blender, and to create one, just select more than one strip, and press the *M* key.

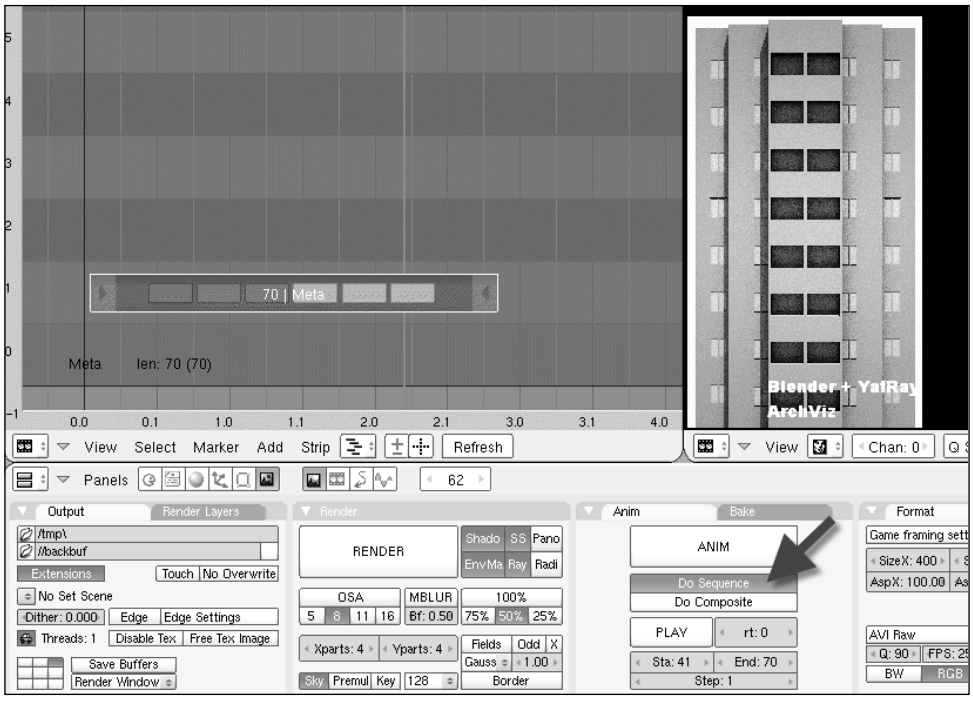
The difference between a Meta Strip and a normal Strip is that we won't visualize all strips of the videos, just a single block:



But, if we need to edit a strip from a Meta Strip, press the *Tab* key. It works the same as the switch between Object and Edit mode.

Exporting the video from the Sequencer

To export the sequence to a video file, we only have to turn on the **Do Sequence** button, right below the **ANIM** button. If this option is turned on, when we press the **ANIM** Button the Sequence will be rendered:



Interactive animation

The difference between a regular animation, created for a video file, and an interactive animation is that here we won't have to set up keyframes. The goal is to create a simple game, where someone will walk and even interact with a virtual environment.

Most 3D suites can't create this kind of simulation by themselves, but Blender can! It's because Blender has a Game Engine, which allows us to use physics simulations and of course, create games.

For architectural visualization, it's very useful to create an interactive walk-through into a project. It's extremely simple to build this simulation, because it will involve only the animation of the camera, and not of characters or complex interactions. So all we have to do is prepare the environment, textures, and lights. Then, we set up the camera movements to respond to keyboard commands.

And guess what? We can do all that without the need of a programming language. Everything is managed by something named a logic brick.

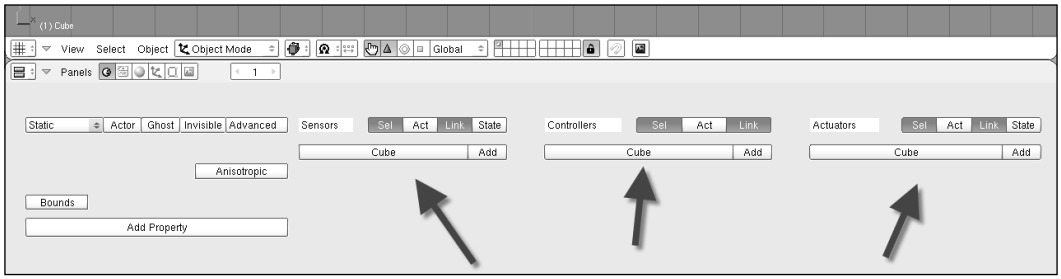
Test drive the Game Engine

To call the game engine, just press the *P* key. If you do that without the proper setup of the scene, everything will look simpler, but the animation won't start. Just press **Esc** to exit the game engine.

Logic bricks

These so-named logic bricks are small menus located in the **Logic** panel. To open this panel, press *F4*, or use the button with a small "Pac-man" icon. This panel displays all options related to the game engine manipulation in Blender. The use of these bricks is simple; we just have to choose options and parameters, and then connect the bricks with a simple drag-and-drop action.

Before we start to work with the bricks, let's understand the different types of bricks and how they behave. There are three types of bricks named **Sensors**, **Controllers**, and **Actuators**:



Sensors

This type of logic brick is used to gather information from the keyboard, mouse, or the scene. For instance, if a user presses a key on the keyboard, or two different objects collide on the scene, an event may happen. Or something may happen to an object constantly, without intervention from the user or anything else.

With this brick, we can send messages to the Actuators every time a specific event happens.

Controllers

Here we have a brick that controls how the action required from the sensor must be applied. We can use an expression, python script, or simply logic controllers, such as OR and AND. For our use, we will primarily use the OR and AND controllers.

Actuators

With this logic brick, we will actually specify what the object should do, or what action must be executed when the sensor brick sends the "signal". There are a lot of different options available, such as move an object, apply torque, change visibility, call a sound file, and much more.

For us there are some particularly useful actions. What we have to know about the Actuators is that they do the action for the interactive animation. The sensor identifies a specific event, and with the Actuator, we can set up the object to do an action, such as move, jump, or play a sound file.

Walk-through

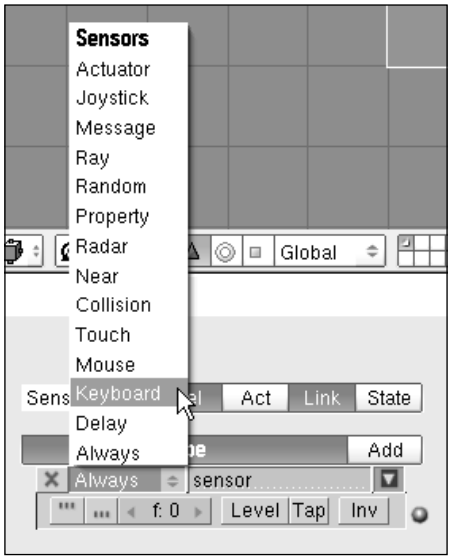
Now that we know what every type of logic brick does, let's start to build our own interactive animation. Before we start, we must prepare the environment. These two steps will guarantee the quality and accurate display of our scene:

- To set up the light for the scene, we use Radiosity to make shadows and simulate some kind of global illumination. If you don't remember how to do that, just check Chapter 11.
- For the textures, all of them have to use UV Mapping to be shown in the Game Engine. If you don't remember how to do that, just check Chapters 8 and 9.

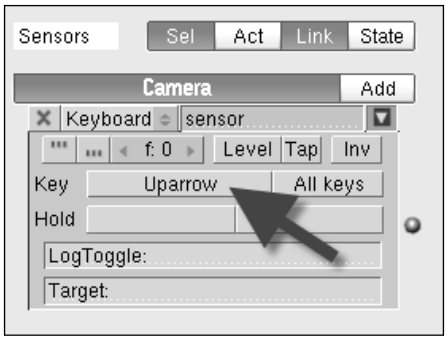
If you have textured your scene for rendering, you will probably have to work on the textures again. It's a completely different production environment, and it's very hard to reuse textures and lighting for the game engine. If you want to test the textures, just try to visualize them with the textured mode in the Blender 3D View. If a texture doesn't show up there, it won't appear in the game engine either.

When the light and textures are all set up, we can go work on the interaction. The first thing to do is to place the camera at the starting point. If you don't have one, create the camera, and right-click on the camera to select it.

When the camera is selected, go to the **Logic** panel, and add a **Sensor** brick. Choose the **Keyboard** type of sensor for this object:



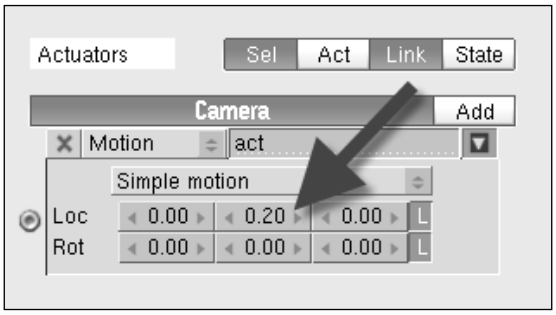
Right after choosing this brick type, we have to set up what key will act as a trigger for this sensor. To do that, press the button placed right next to the **Key** option. When you press this button, a message saying **Press a key** will appear. Then, press a key on the keyboard:



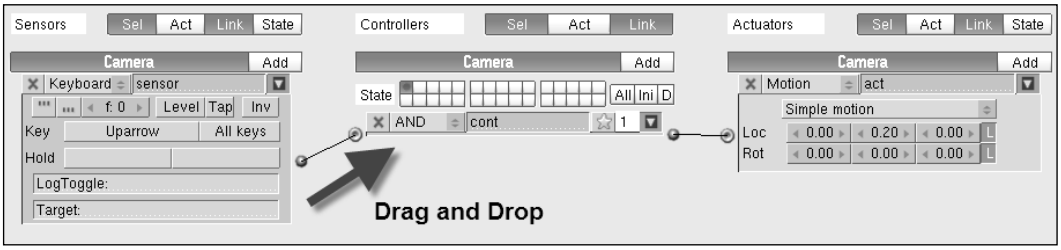
For instance, if we press the **Uparrow** key, a message is displayed showing that this key was selected. After that, add a **Controller** brick and choose the **AND** type. It will connect the trigger, which we set up with the sensor brick, to the actuator. When the user presses the up arrow, we want the camera to move forward. So, we have to add an **Actuator** brick, and choose the **Motion** option.

This actuator presents six types of forces available to us. We will use the **Loc** and the **Rot**, which will change the position and the rotation of the object, respectively.

There are three columns to alter the values for the x, y, or z axis, respectively. In our scene, to move the camera forward, a positive force in the y axis is required. So we alter the value in the **Loc** line, and in the second column. Increase the value until it reaches **0.20**:



How do we know if it works? Press the *0* key to see the active camera, and then press *P*. It will start the game engine. If everything was done correctly, when we press the up arrow key, the camera will move forward. Repeat the same steps to create the actions to move the camera backwards and sideways. If you want, set up the camera to rotate when the user presses the right or left keys. This way, his/her view will always be pointed to the front:



If your scene is big, it may be a good idea to set up the camera to look up or down. This way the user will be able to move and explore your scene more naturally. Just add some forces to the **Rot** line.

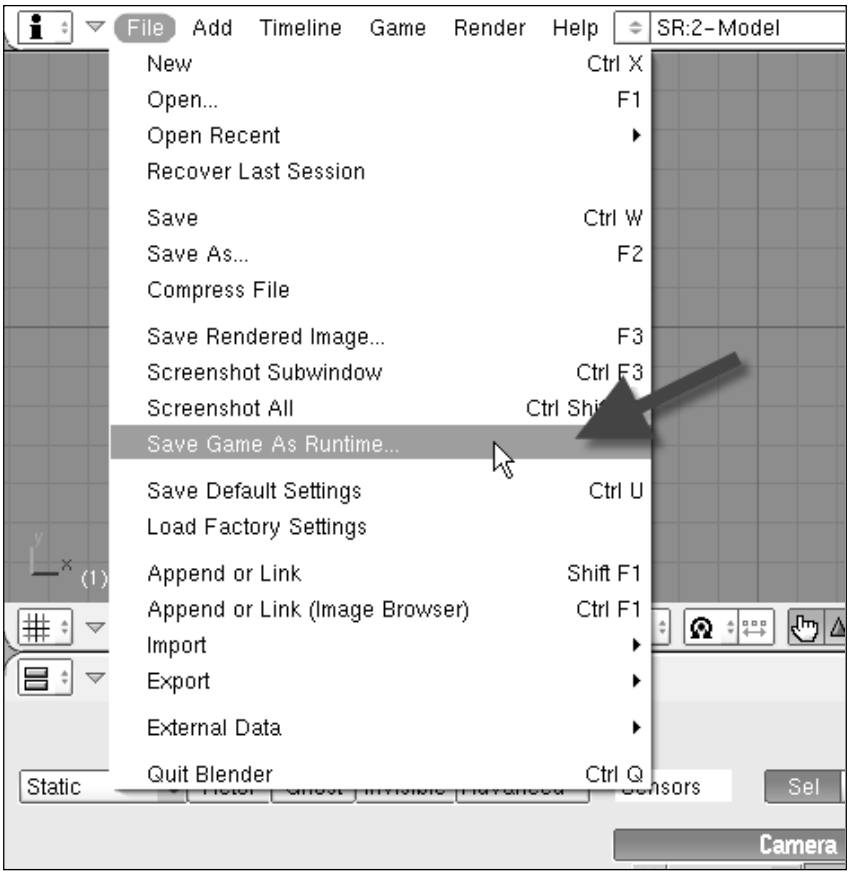
As you can see, it's simple to create a walk-through animation with Blender. When we show a project with this kind of animation, the audience will have that feeling of being in the real scene and controlling what they want to see, and not just what you want to show them. If you have the time, always use this type of animation; it's worth the setup time.

Walk-through template

The Blender Foundation has a template for the walk-through animation, available for anyone to download. This template has all the actions for the camera already set up and configured. All we have to do is place the scene, and use the camera. It may require some adjustments, because of the scale. If you want to try it, visit this link: `http://www.blender.org/education-help/tutorials/tutorial-folder/3d-walkthrough`.

Export walk-through

When the animation is ready, we may create a stand alone application to distribute it on a CD/DVD-ROM, or make it available on the internet. To do that, you have to pack all the textures on the scene, or they won't be shown in the animation. When everything is set up, use the **File** menu, and choose the **Save Game As Runtime…** option:



And that's it! This option will create a stand alone application, created especially for the system that you are using. For instance, for a Blender running on a Microsoft Windows system, a stand alone application for this system will be created.

Take Blender with you!

Because Blender is open source and can be installed without the need of a license, it's a good practice to take a copy of Blender together with the original file when yo need to make a presentation. It will avoid any kind of issues related to compatibility or missing files.

Summary

Even for a complex subject such as animation, this chapter presents the techniques and tools used to create animation in Blender. The focus was on camera animation, which required a lot less work than character animation, but not fewer adjustments.

Here is a list of subjects learned in this chapter:

- What is animation, and how to plan the process to avoid problems and issues during the creation of the animation
- Different types of planning for animation, such as animatic and storyboards
- The types of keyframes in Blender and how to use them to make animations
- Set up the animation with three special windows in Blender; Timeline, NLA Editor and the IPO Curve Editor
- How to create interactive animations and make standalone applications with them

14
Post-Production with Gimp

What would happen if you found out that your render that took many hours to complete, but it didn't look quite the way you wanted? In this situation, two things would probably happen. The first one would be to adjust the model or the problematic setup and render again. The second one, which will be the subject of this chapter, would be to take the rendered image and edit it in an image processor software package, such as Gimp. Of course, the second option is always the best choice in these situations, especially when the render time is long.

With software packages such as Gimp or Photoshop, we can make adjustments to images, which would otherwise require a new render, with just a few mouse clicks. If the bricks of a wall don't have the right brightness, or the color balance should be different, just select the area and edit it.

Well, that's just one case. There are a lot of advantages to the use of Gimp. If you manage to work with render passes, an image editor is important to gather all images together and build the final composition.

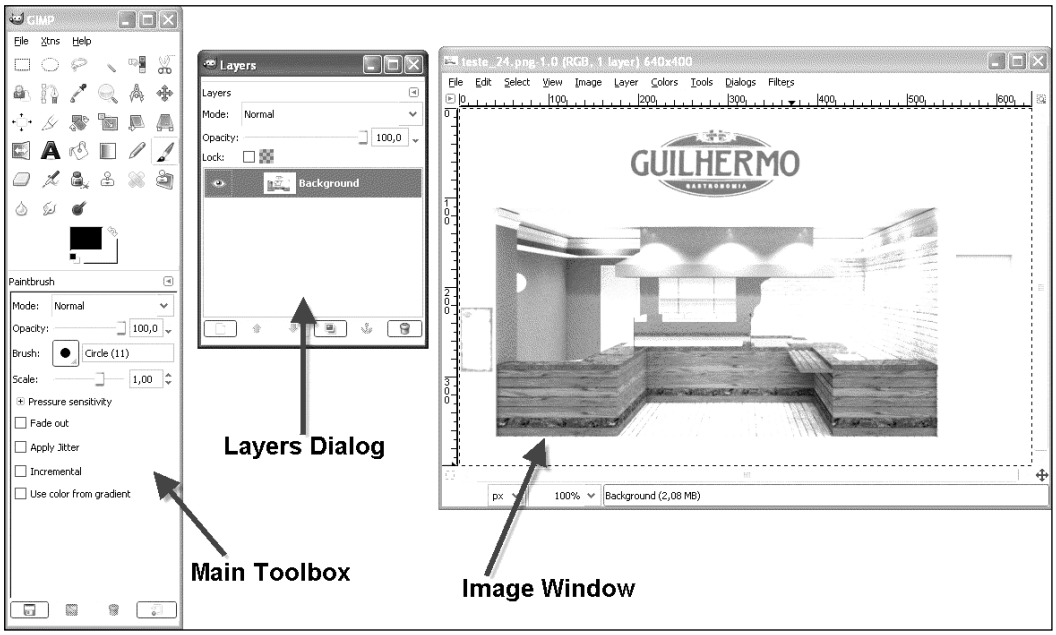
Gimp interface

Before anything else, let's take a look at Gimp and it's interface. It's important to know a few basics about it, even if the focus here is not to teach how to use all features of Gimp, but just the main tools and tasks related to post-production work.

The interface consists of different windows. Each window has tools and options to edit or organize the image. Here is a list of the main windows in Gimp:

- **Main Toolbox**: This menu holds the main options for Gimp, such as the selection tools, paint, and all the other main tools.
- **Image Window**: Here we have the canvas of the image, which has the actual composition. When we do any kind of adjustment, it's here that we will see the results.

- **Layers Dialog**: As the name suggests, here we find options related to layers.



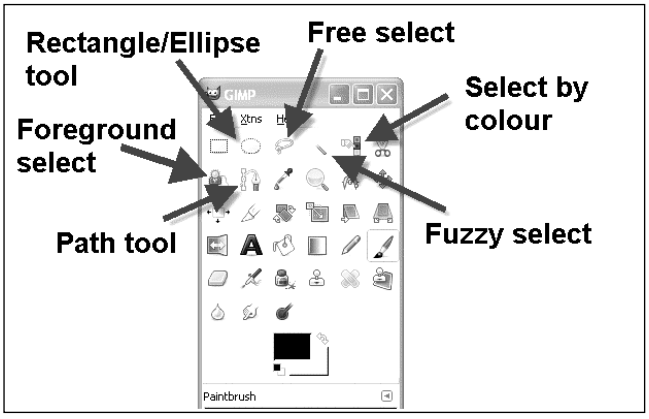
Selection tools

Now that we know how the interface looks, it's time to learn a bit about selection. If the adjustments required for an image concern the whole canvas, then the process is easier. We won't have to select anything. But if only a small part or object in the image requires adjustments, we will have to select it, and then apply the adjustments.

With the selection tools available in Gimp, things will be a lot easier. Here is a brief list of all the tools:

- **Rectangle/Ellipse Select tool**: Here we have tools to select areas with regular shapes.
- **Free Select tool**: This is also named the lasso tool in other image-editing software packages. On the top, left corner of the toolbox, we have the icons for the Rectangle selection, shaped as a small rectangle, and right next to it, the Ellipse selection, shaped as a small ellipse.

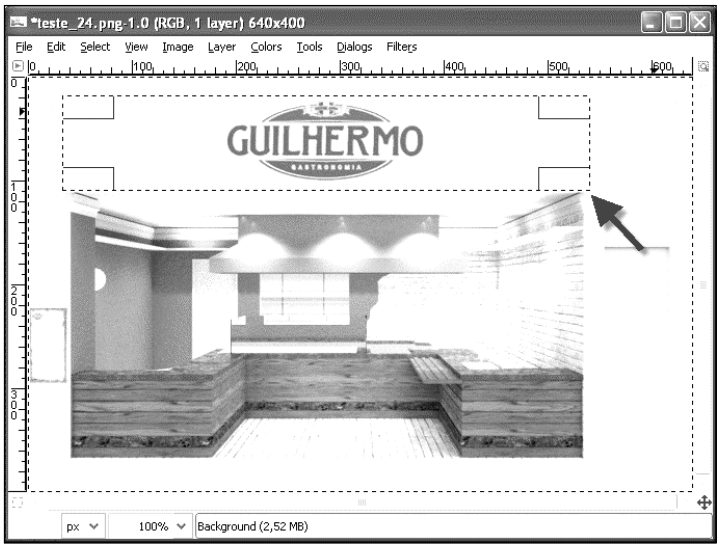
- **Select by Color tool**: With this tool, we can select a single color from the area of an image. For instance, click on the area you wish to select and all the pixels with the same color will be selected.
- **Fuzzy Select tool**: It works much like the **Select by Color tool**, which allows us to select pixels with the same color. The difference is that with the **Fuzzy Select tool**, only a continuous area is selected.
- **Foreground Select tool**: Here we can select an object or area, with the objective of splitting the image into layers. The selection process involves two parts. First, we use the free select tool, and then we paint the selection to fine-tune it.
- **Path tool**: If your selection must be done with precision, this is the best option. Here we mark the area that should be selected with a Bezier curve. Because these curves are actually vectors, the lines can be adjusted easily to fit an object or area.



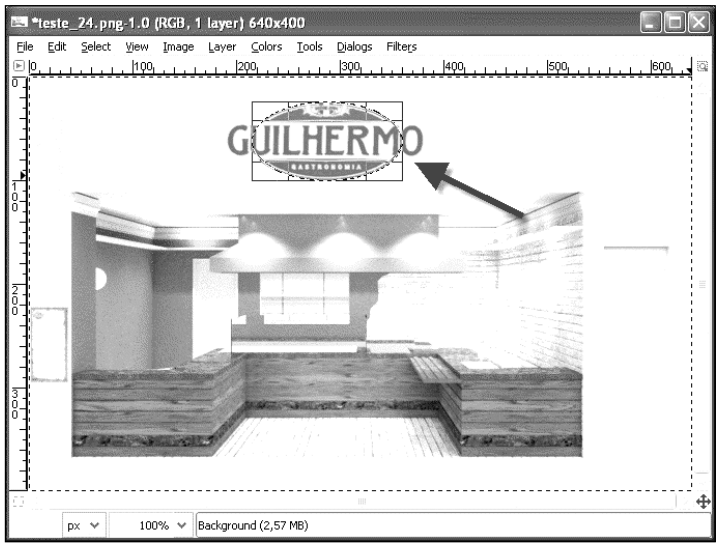
Selecting regular shapes

Instead of going through each tool, let's analyze the types of selection that you will have to make. The simplest and easiest type is the selection based on regular shapes, such as squares, circles, and polygons.

We will use an image as an example to make selections on regular shapes. First, we will make two selections, one with a rectangular shape, and the other with an elliptical shape. This can be done with both the Rectangle and the Elliptical selection tools:

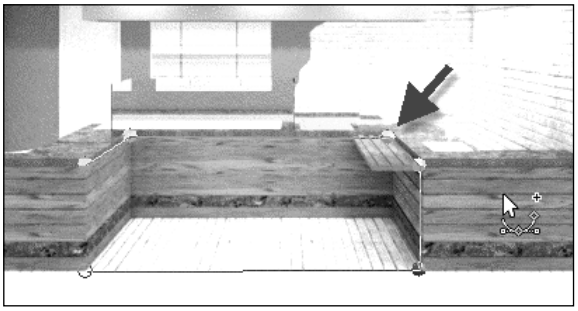


When the proper tool is selected, just click and drag over the image to select it. For this area, we want to select the sign, which has a rectangular shape. And after that, select the logo, which has an elliptical shape, in the same sign:



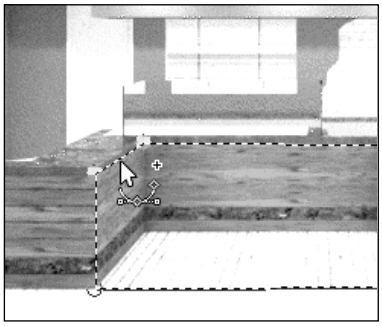
To select a regular shape, just press the *Shift* key while you drag the mouse. It will make the shape of the selection become regular.

If the shape to be selected is an irregular shape, we can use the Path tool. For instance, let's select the floor, which has a more irregular shape. Activate Path Select and click once on each vertex of the shape. Every time you click, a small black dot will appear to mark the point:



When all the points are marked, just press the **Selection from path** option to make it change into a selection. You don't have to close the path, just mark all the points, and the tool will complete the shape for you:

If something needs to be curved, just click on the line and drag the mouse. It will change the line into a curve. Along with the curve, control points will appear for the selected curve. Use these points to fine-tune the path, and create a shape that fits your area or object. It's possible to add more points; just press the *Ctrl* point, and click on a line, and a new point will be added:



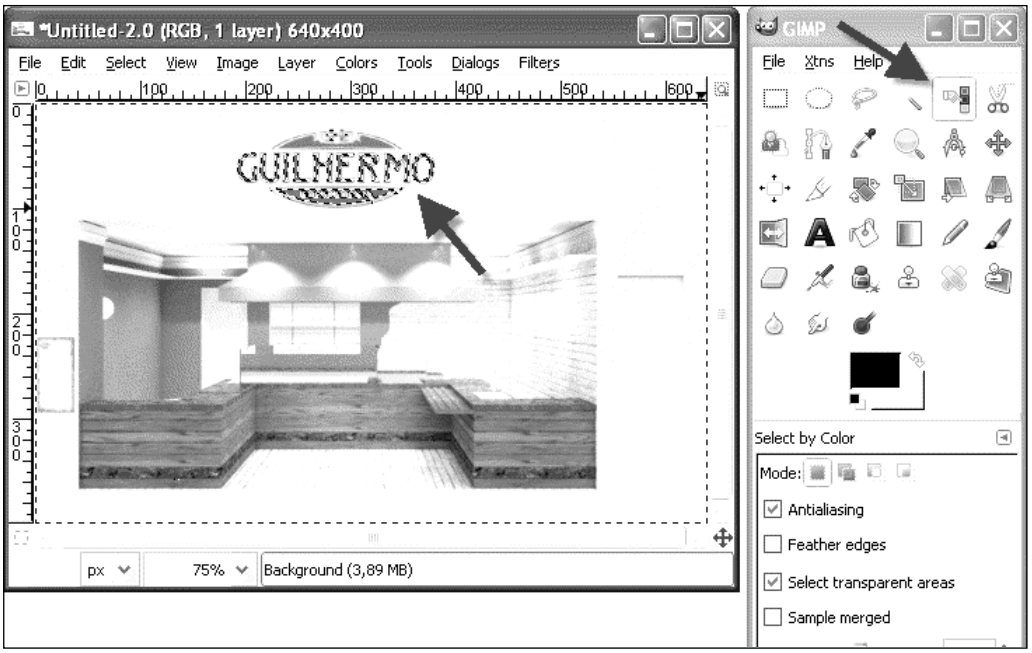
Complex selections

It's possible to add or remove a part of the selection. We can add new parts to the selection by holding the *Ctrl* key while we drag the mouse. To remove a part, just press the *Shift* key.

Selecting by color

If we have a large area to be selected, with a single or prominent color, there are two tools that can select those areas easily. We have the **Fuzzy Select** and the **Select by Color**. The first one can select a single area and add all pixels with a similar tone to the selection, but never outside of the area. For instance, if we have a white wall, and click on it with this tool turned on, and all white pixels from that wall will be selected.

If we have another white wall in the same image, but in a different location of the rendered image, this other wall won't be selected. The other tool, named **Select by Color** works in a very similar way, but with it, we will select all pixels with the same color:



Looking back at the wall example, if we have three walls with white pixels, all these walls will be added to the selection together with all other elements or areas that contain white pixels. Because of that, we have to be careful as to when and where we use this tool. Sometimes, it's a lot easier to use the **Path** tool, instead of the **Fuzzy Select** or **Color** Select. Use it only if you have an area with a solid color, which will make the process easier, and the adjustments for the selection will be very quick to implement.

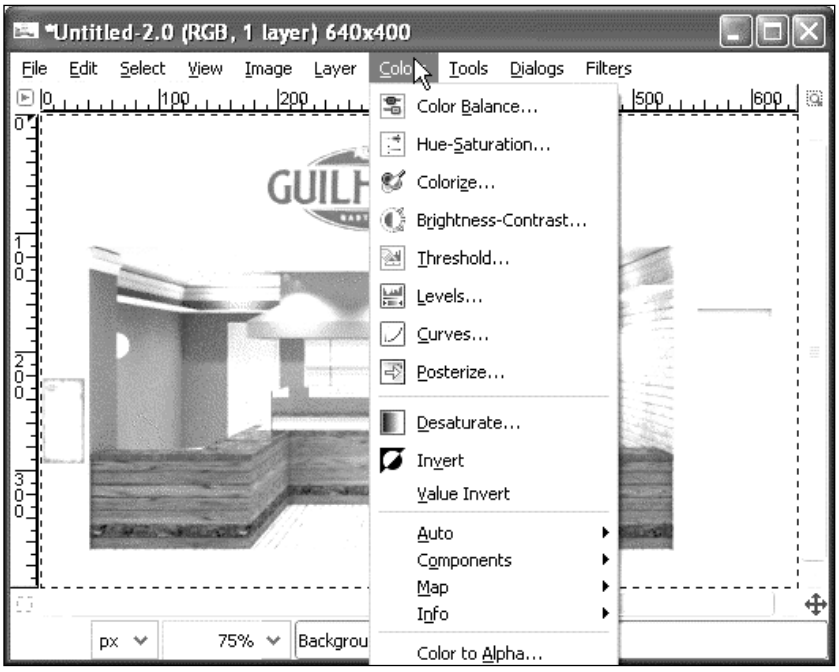
Quick undo a selection

If you want to remove a selection, always use the *Ctrl + Shift + A* shortcut. When we press this combination of keys, any active selection will be undone.

Color adjustment

Now that we know how to select just a small portion of our images, we can edit and adjust the color of the rendered images. If a part of the image is selected when we activate the color tools, just the selected part will be affected. To change all pixels from an image, just use the tools without selecting anything.

To see the options for colors, we use a menu named **Colors**, located in the canvas window:



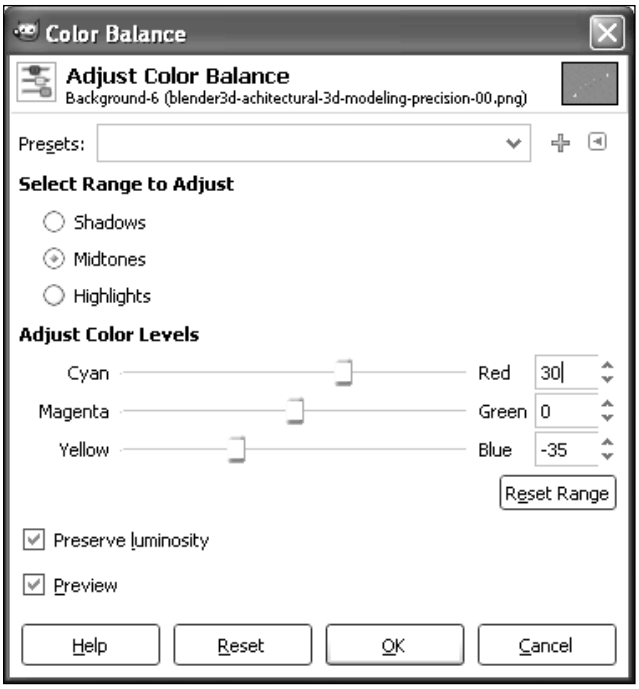
There are a lot of options from which to choose, but not every one of them is useful for us. Just a few have a particular use or can be helpful to edit an image.

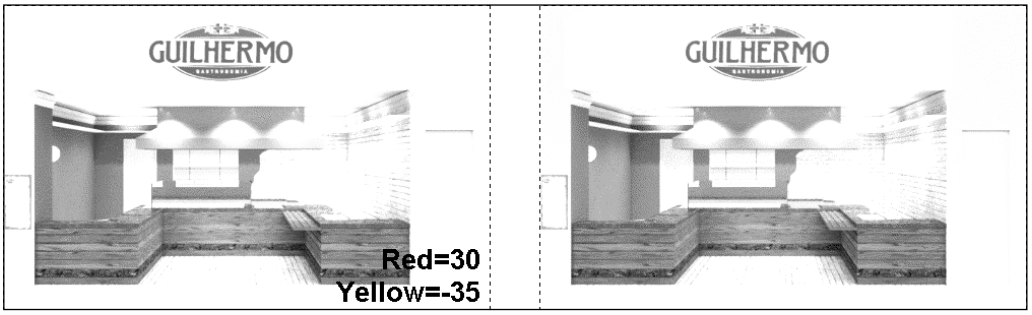
Let's see what we can do with them.

Color balance

Sometimes, when an image is rendered, we miss some tones, or the image don't give the desired feel that is required for the scene. For instance, if we work on the scene of a room illuminated by the sunlight, the whole room should have a warm feeling. This can be accomplished if the image has a lot of yellow and red pixels, but not too much. If you find that setting that up with the Blender lamps is too hard or time-consuming, you can try to adjust it with Gimp!

With the **Color Balance** option, we can make adjustments to an image to make it look warm or cold. Just call the menu option, and a dialog box will appear:

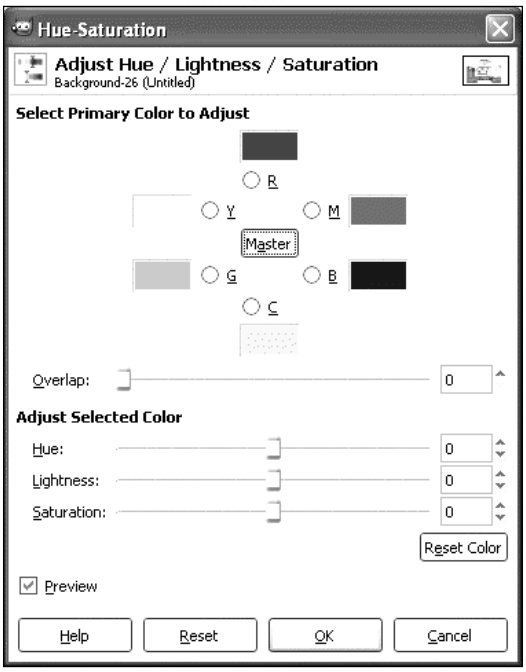




Use the sliders to adjust the amount of each color that will be added to or removed from the image. Another great use of that feature is that if a color is not right or the light reflection is wrong, we can remove the main tones, for example, by increasing the amount of Blue to remove the Yellow pixels.

Hue and saturation

If the color balance is not the problem, then we can try the **Hue and Saturation** menu item, which allows us to adjust the saturation of the overall image or just a primary color. To use this feature, access the **Hue-saturation** menu item just below **Color Balance**:



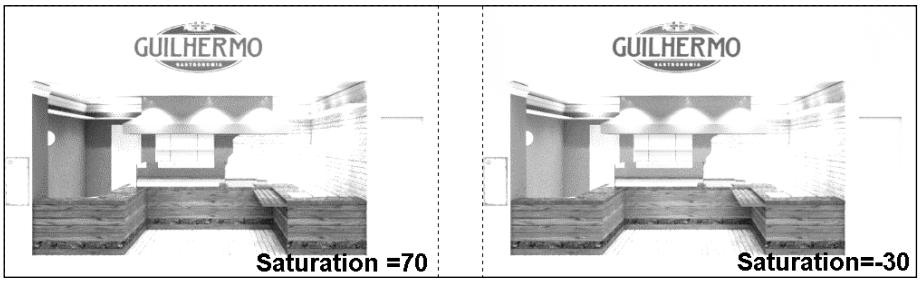
In the dialog box, we can choose to adjust only one of the **Primary Colors** or the **Master** channel, which contains all of the primary colors.

The most useful adjustments for us here will be the **Saturation** and **Lightness** sliders. With the first one, we can change the level of saturation of the colors. A highly saturated image has vivid colors, and a low level of saturation makes the colors look grayer.

We can use this option to improve our prints! For instance, if your printer makes your image look dark, with a high load of black ink, then we can increase the saturation to make the image look more vivid amount of black pixels.

Together with the **Saturation** option, we can use the **Lightness** slider to make the colors of the image tend towards white or black. Always use it with the **Saturation** option, and perform a lot of tests to determine the best setup for your image.

Take a look at the difference of two images edited with a fine tuning of **Saturation**:



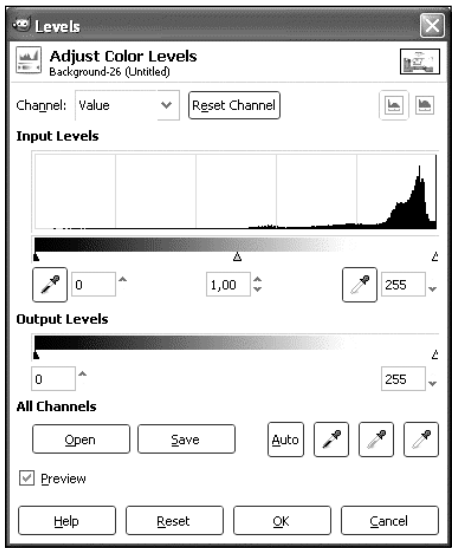
Color enhance

There is an option which allows us to make a quick adjustment for color. In the **Colors** menu, choose the **Auto** option, and then choose **Color Enhance** to enhance and make all colors in your image look a lot better automatically. If you don't want to make tests and tryouts, this is a good shortcut for color adjustment.

Color level

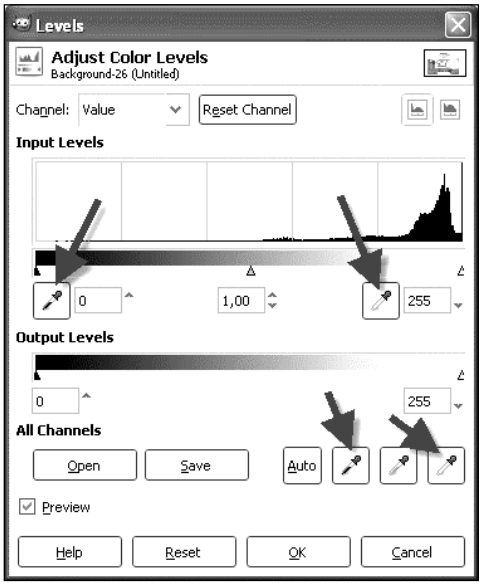
Just as in digital photography editing, we can use **Color Levels** to adjust our image. For that, we will work with the channels of RGB color (Red, Green, and Blue). It's also possible to work with the master channel, which has a mix of all those colors.

Before anything else, we have to access the **Levels** menu item:



The dialog is very simple to use. We can choose which **Channel** will be edited, and work with the sliders to fine-tune the levels. These levels deal with how much white or black color we have on the image. All the levels range from **0** to **255**, with the **0** for black and **255** for white.

To make things easier, we can pick some white and black pixels to make the editing work better:

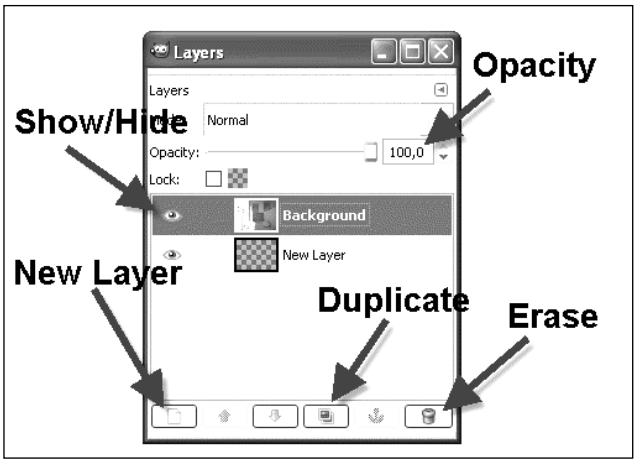


Try to find a good balance between the two opposites, which makes a good image. If you are working on a series of images, and all of them could make use of the same adjustments for the color level, there are the **Open** and **Save** buttons to save the adjustments, and use them again with another image.

Layers

Along with the color adjustments, we can use layers to split the image and work on the composition of the elements. Sometimes we will have to add a background, or simply split the image on layers to fine-tune the colors of a specific area.

To work with layers, we use the **Layers** dialog:



If we create a new layer, by pressing the corresponding button, an empty layer will be created above the image. To locate it below the image, just click on it and, drag the layer down.

Another good practice is to duplicate the main layers before you work on some adjustment. This way, any mistake will be applied to the duplicates, and the original layer will be untouched. We can also hide a layer to see only what needs to be edited. For that, click on the eye icon on the left side of the dialog.

Create a new layer

To create a new layer, just press the button pointed to in the previous image. But this new layer will be empty. If you want to add any graphics to this new layer, just use the painting tools or paste an object.

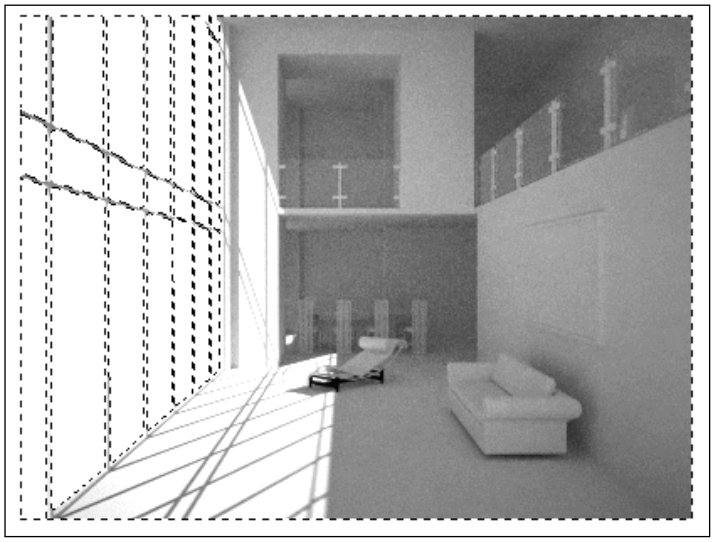
Create a new layer from a selection

Another way to create a new layer—just select an area of the image with any selection tool. When the area is selected, just copy and paste. Yes, use the famous *Ctrl + C* and then *Ctrl + V*, or simply use the **Edit** menu.

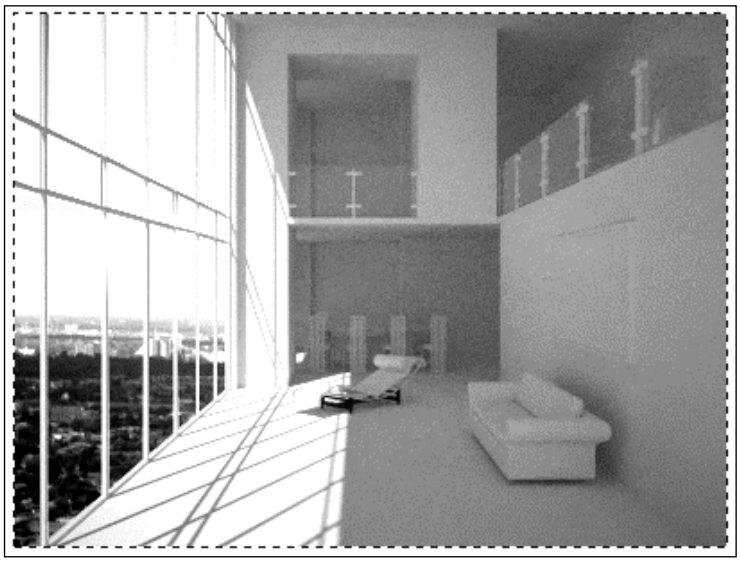
Adding a background image

If you create an external image, you will probably need to add a background. It can be a picture of a sunny sky, or simply a photo of the actual location where the building will be created.

To do that, we open the rendered image, and select the parts that will be erased. In this case, the parts that will be excluded are the ones representing the window. Use the regular selection tools to select the window, and then use the **Select** menu, and choose **Invert**. It will select everything but the window area. Now it's time to copy the image; press the *Ctrl + C* keys:



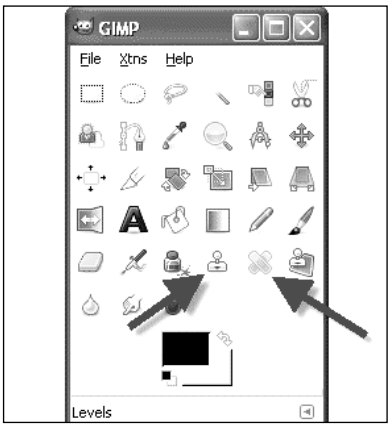
Open the image that will be the background, and it leave it opened. Now press the *Ctrl + V* keys to paste the image:



And that's it! After the image is placed, you may have to make color adjustments to fit the levels and balance of the photo and the rendered image. If the layer is selected, the adjustments of color will be applied only to that layer.

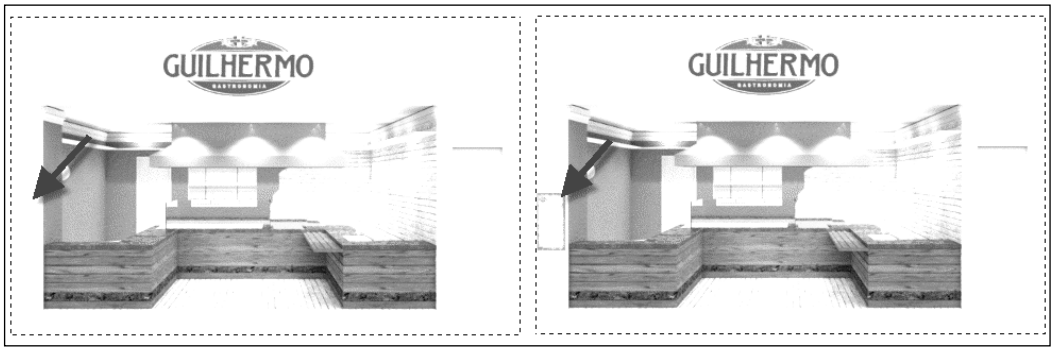
Fixing errors

What if something is rendered at a different location, or a part of the geometry is not right? If you don't want to render the image again, you will have to edit the image in order to fix the error. To do that, we have two incredible tools available in Gimp. The first one is named the **Clone** tool, and the other is the **Healing** tool:



With the **Clone** tool, we can clone pixels over selected areas. The process is quite simple. After activating the tool, we have to hold the *Shift* key, and click once on the area where we want to clone. This will give us the source of pixels. Then, we click and drag over the area that must be overlapped by the cloned pixels.

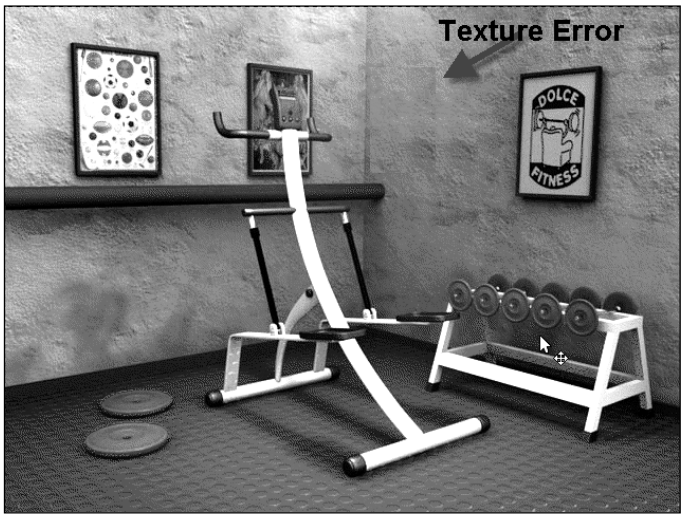
Here is an example where we clone an area of the image to fix an erroneous geometry:

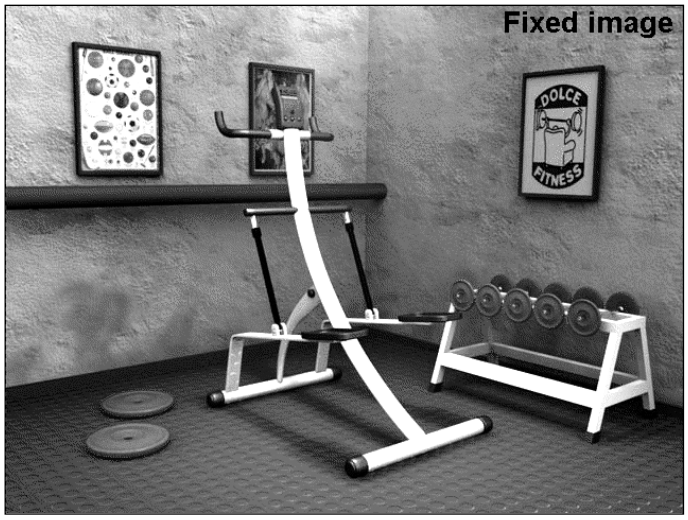


The **Heal** tool works in a similar way, but in a much smarter way. The pixels here are not simply copied, but they are analyzed and only the required pixels are changed.

This tool was developed to adjust photos, but for us it's especially useful to change and fix small spots in textures. For instance, if a textured surface is rendered with small defects, use the healing tool to repair it.

To use this tool, we follow the same steps as using the Clone tool. When the tool is activated, press the *Ctrl* key to mark an area as source. Then, click and drag the mouse over the area that should be healed:



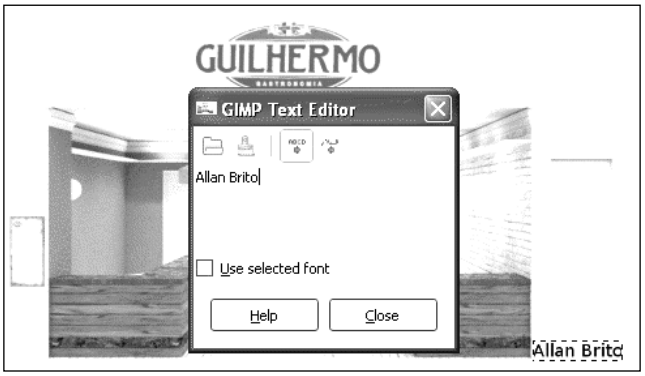


Watermark

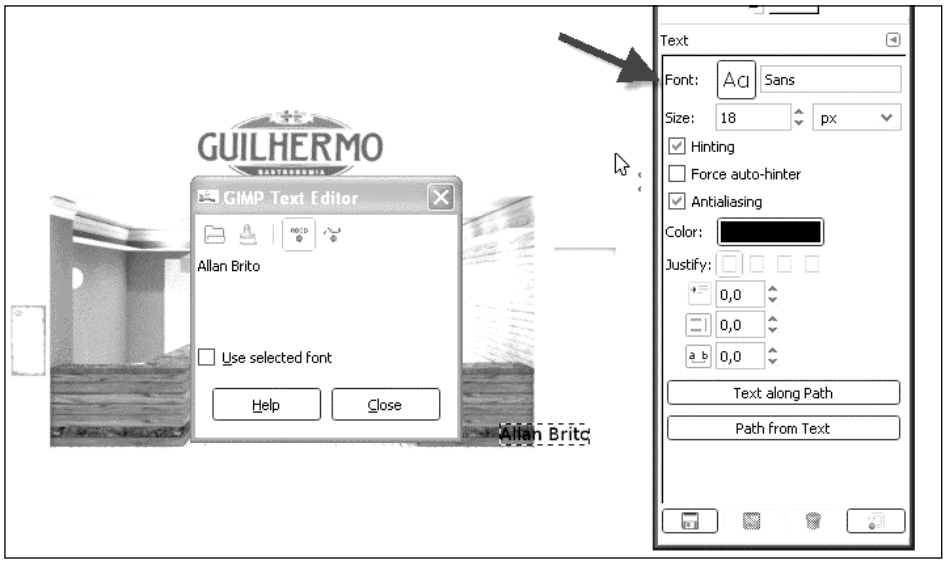
The best way to protect your work, and to show everybody who is responsible for a particular image, is with a watermark. To add a watermark in Gimp, we use the Text tool and the layers controls.

Let's see how to work with it by adding a simple watermark to an image. The first thing to do is choose the text. For my example, the text will be **Allan Brito**.

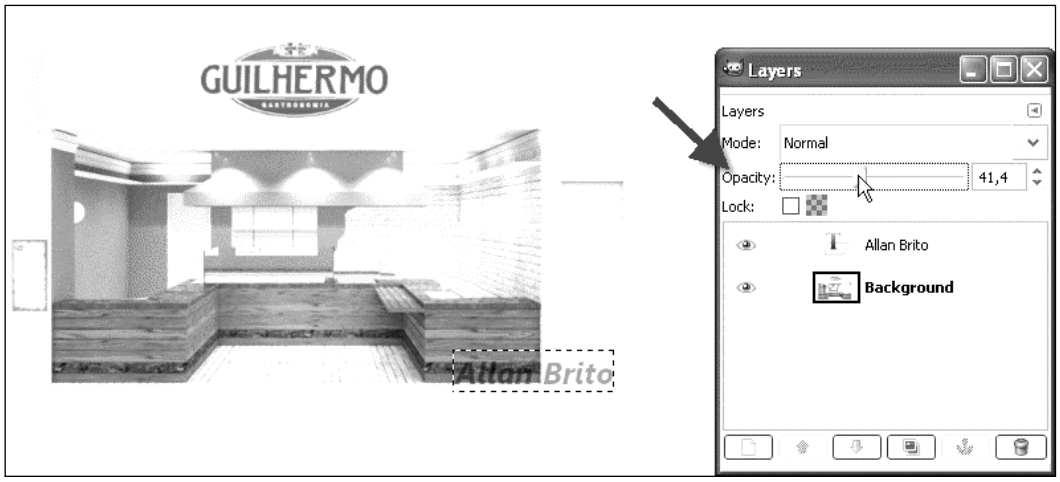
So, we select the **Text** tool and click on the image. It will make a dialog box open, where we should type the text:



If you want to change the Font and Size of the text, just use the **Text** options in the main window:



Now we can make the text transparent. To do it, just select the text layer, and decrease the **Opacity**:



And then you will have a watermark, which will show everybody that you are the author of the image, and help prevent unauthorized copies of your work.

Summary

In this chapter we didn't work with Blender, but learned how to make our life easier, avoiding the long render times of Blender and switching to Gimp at the end of the work for the post-production process. Now you know that adjustments can be done in Gimp with a lot less effort than in Blender, especially for color correction.

In addition to the color correction, we can even make improvements on erroneous geometry with the Stamp and Heal tools.

Here is a list of what we have learned in this chapter:

- Use Gimp tools to make color correction
- Correct errors caused by displaced geometry with the Stamp and Heal tools
- Use layers to composite the image with real photos
- Add text to the rendered images
- Add watermarks to the images, to protect your work

15
Blender 2.50 and Architectural Visualization

In this chapter, we will take a look at the next big update in Blender, which will be the highly awaited Blender 2.50. This is a major update for Blender users, and may affect architectural visualization artists who use Blender for their work. We will learn about the changes in the user interface and how the modeling and rendering processes will look like in the new Blender.

Development of Blender 2.50

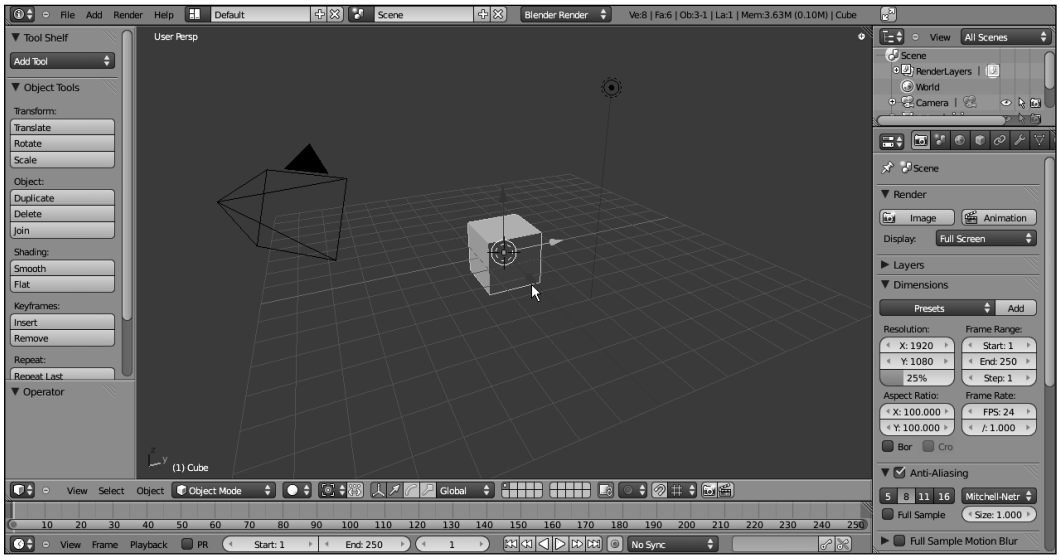
The development of Blender 2.50 follows the creation of the third open movie named Sintel, which you may know by the name of project Durian. A lot of features and enhancements of Blender 2.50 have Sintel as the point of reference, with character animation in mind. By the time this animation is finished, we will have a stable release of Blender 2.50, which will turn immediately into Blender 2.60. Yes, the 2.50 will be used only for the development of a stable version of Blender that will result in a 2.60 series.

To allow users and artists to experience and test the new tools and interface, many experimental and beta versions of Blender 2.50 will be released. Presently, the most recent of those versions is Blender 2.50 Alpha 2, which will be used in all screenshots and references for this chapter. There will be additional versions of Blender 2.50 before an official release of 2.60 is ready to see daylight. Until then, lots of new features and tools for character animation and modeling could be added to the development version.

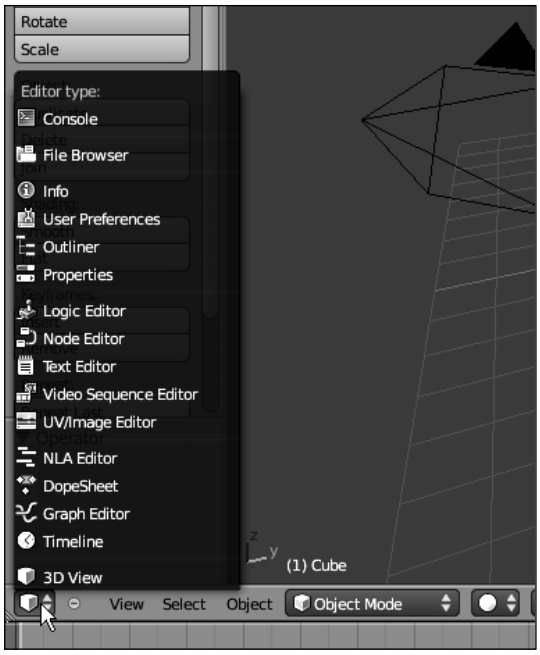
What about architectural visualization? With lots of character animation features in mind, can we expect some improvements for architectural visualization? Sure! There will be lots of new tools for 3D modeling and rendering that could be used for architectural visualization.

User interface

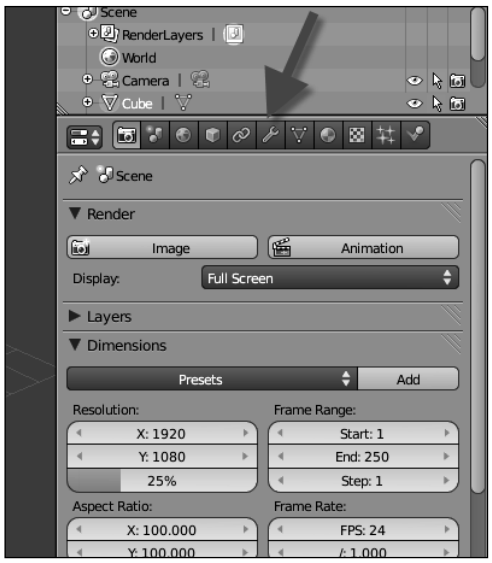
Before we get to those features, let's take a look at one of the biggest changes in Blender since it's release, the user interface! The following image shows the user interface of Blender 2.50 Alpha 2; this design probably will remain until the release of Blender 2.60:



As we can see in the screenshot, there are a lot of areas that remain familiar, and others that are completely new. The window system of Blender looks a lot like the one used in 2.49, with windows such as **3D View**, **Timeline**, **Outliner** and new options, such as **Properties** and **Graph Editor**. Well, some of the changes of Blender 2.50 are simply renames of old tools, such as the Buttons Window renamed to **Properties** and the Ipo Curve Editor renamed to **Graph Editor**. Some of those windows look the same, but other windows are completely redesigned:



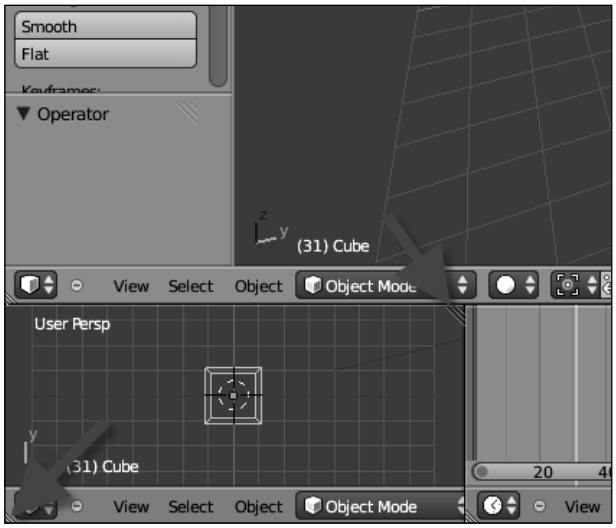
Of course, it's not only a matter of renaming the panel, but adding features and changing the way it works. For instance, in the **Properties** window, we will find lots of options from the Buttons Window of Blender 2.49 and also several new panels. There is a dedicated panel to hold modifiers only and another one for Physics simulations:



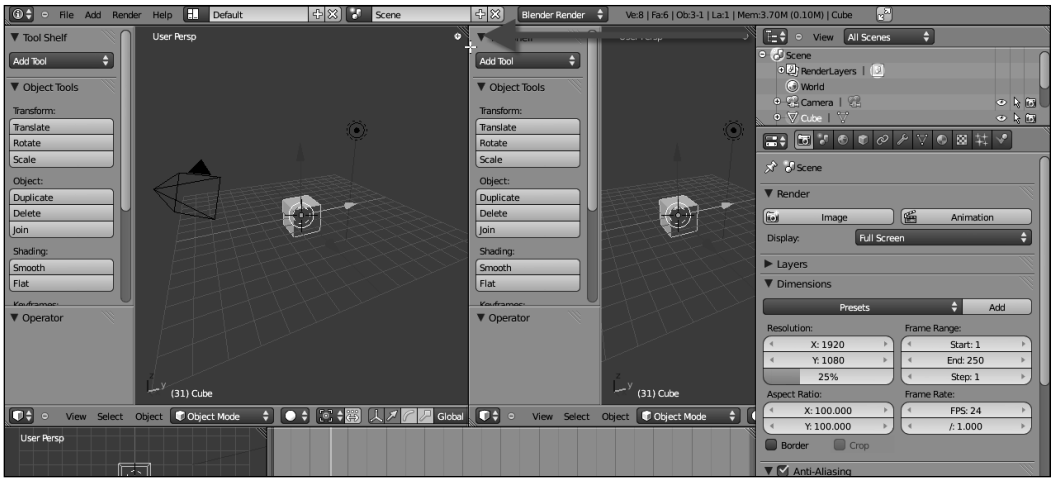
In Blender 2.49, we had a hierarchical structure to organize the panels, but in Blender 2.50, we don't have this hierarchy anymore. All panels are accessible in the Properties window, no matter what. Some of them depend on which object is selected in the **3D View** window, such as the option to change Lamps.

Managing windows

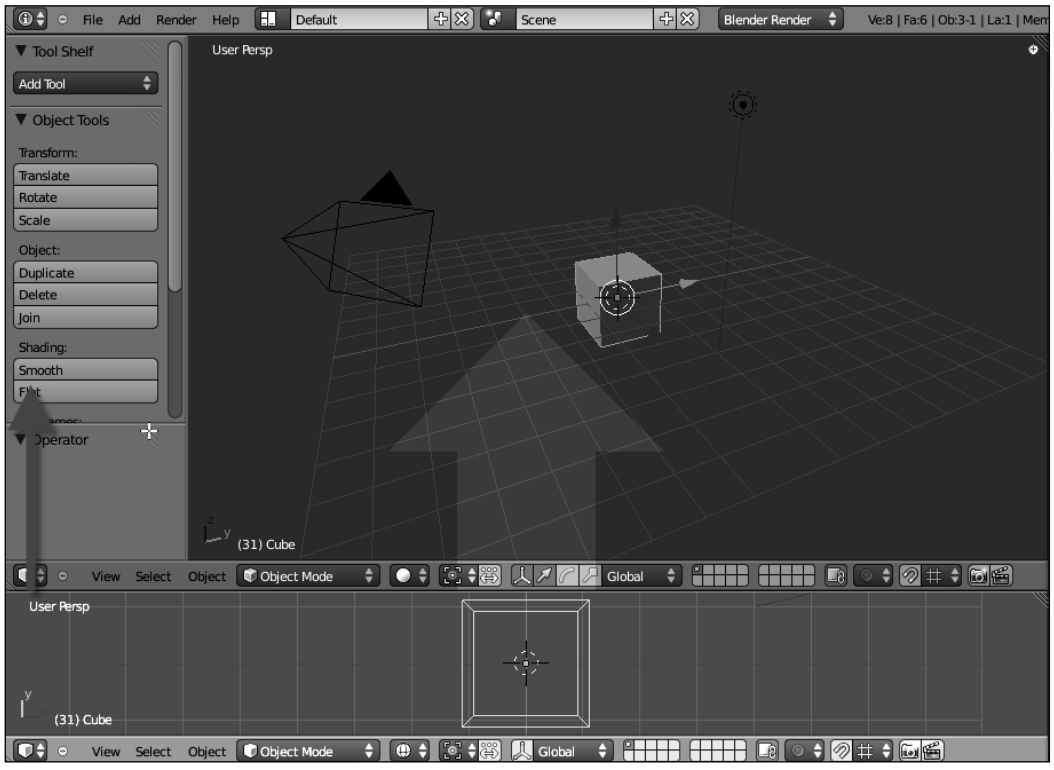
The way we manage windows in Blender 2.50 has changed in comparison to 2.49, and now we have to use mouse movements to merge and split windows. Back in the old Blender, we had to click with the right mouse button on the corner of a window, to either split or merge a window. Now, we have to use the lower-left or upper-right corners of the windows to do that. To create a new division in the windows, just left-click on the upper-right or lower-left corner of a window and hold down the mouse button:



If you clicked on the upper-right corner, just slide the mouse to the left to create a vertical division, or slide down to create a new horizontal division. The process is exactly the same with the lower-left corner, but we should slide the mouse to the right or up:

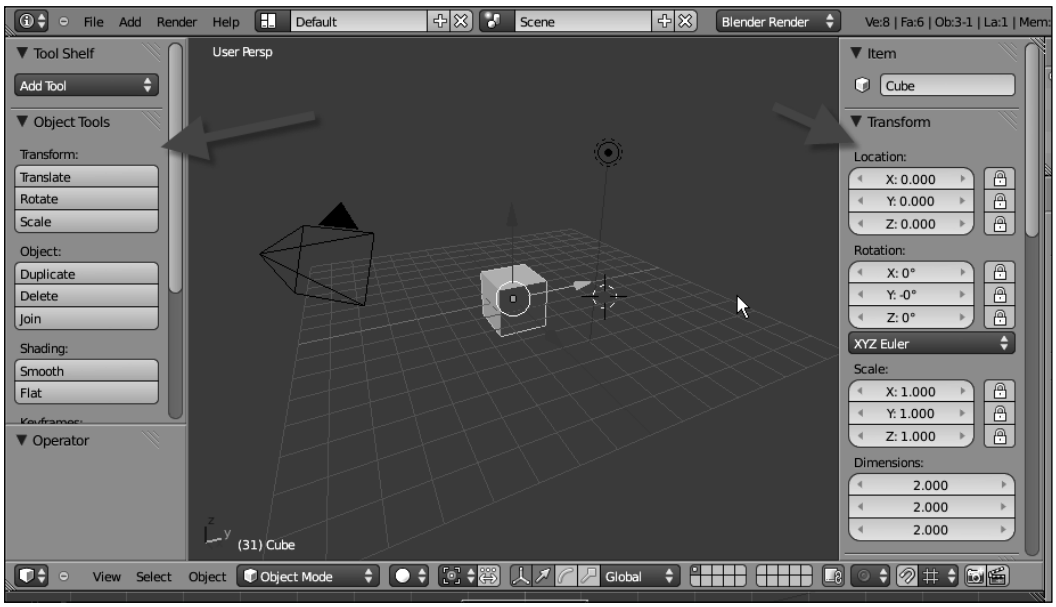


To merge a window, just use the same process described to create a division, but slide the mouse in the direction of the window that you want to merge:

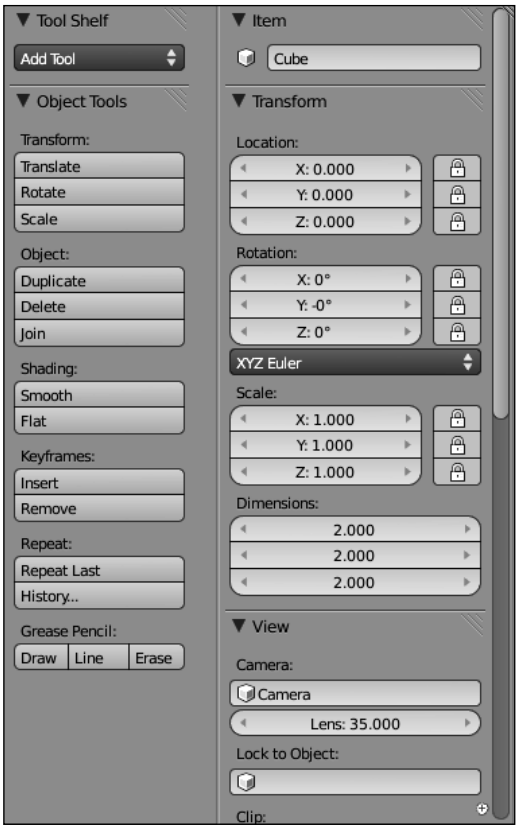


3D View

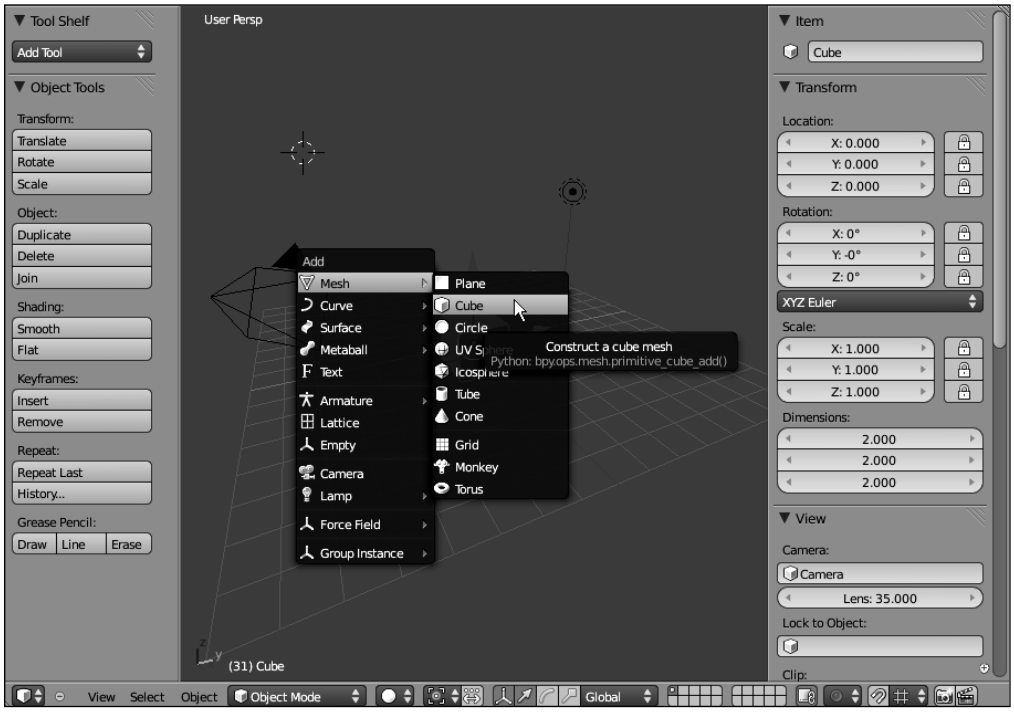
The 3D View looks exactly the same as in 2.49 except that we don't have the floating menus for properties and view options. The menus are docked at the sides of the 3D View:



To open the menus, we can use two shortcut keys, which are *N* for the properties toolbar, and the *T* key for a general toolbar. In the first one, we can add and transform many of the properties of a selected object. In the second one, we have general editing options and contextualized tools to change an object:



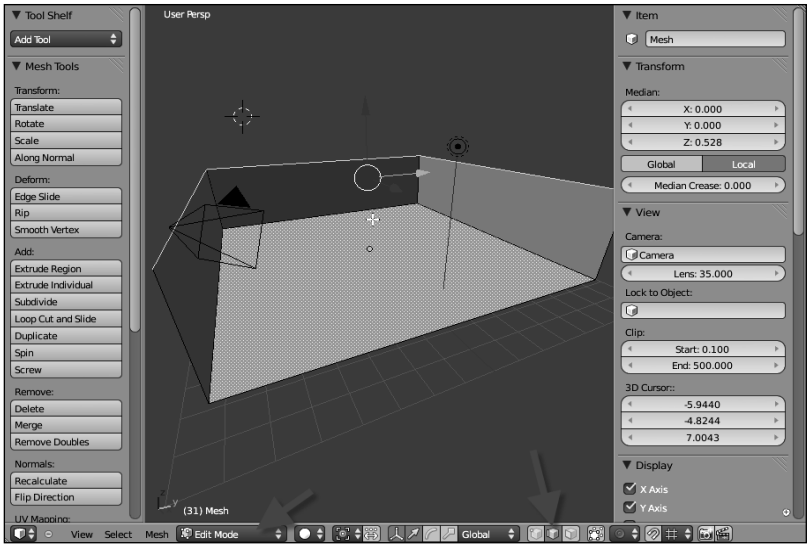
The management of objects in the 3D View is very similar to what we are used to in 2.49. Only a few shortcuts have changed, such as the *Space bar*. Now, when we press the *Space bar*, a search box will pop up. To open the toolbox, use the *Shift* + *A* keys:



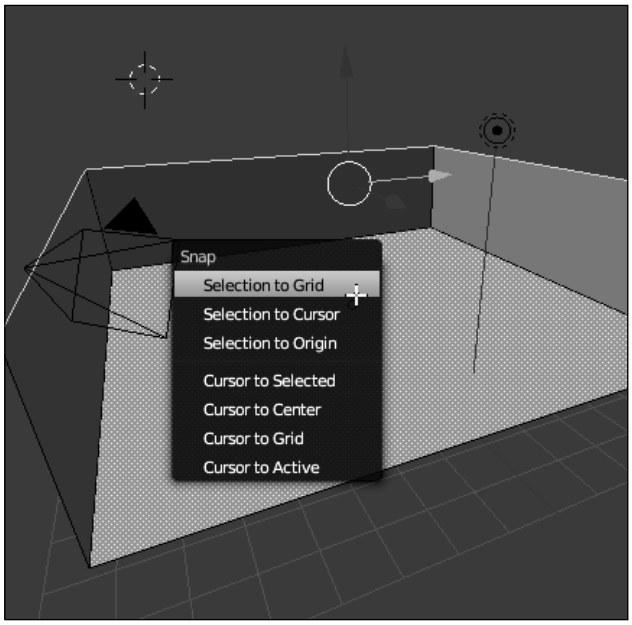
If you don't like the shortcuts of Blender in 2.50, we have the option to change them in any way we want. Just open the **User Preferences** window, and choose the **Input** option. There we can press the **Edit** button, and change the shortcuts of Blender in any way we want, even turning them back to the same way they used to work in 2.49:

Modeling

The modeling process in 2.50 hasn't changed much compared to other areas. We still have the Edit and Object mode, and the creation and editing of Mesh objects works just as in 2.49. An object such as, a plane, is created with the toolbox, and with the tools available in **Edit Mode**, we can extrude and transform the object:



The selection modes are there, along with the snapping tools. Press the *Ctrl* + *Tab* for selection mode and *Shift* + *S* for **Snap**:

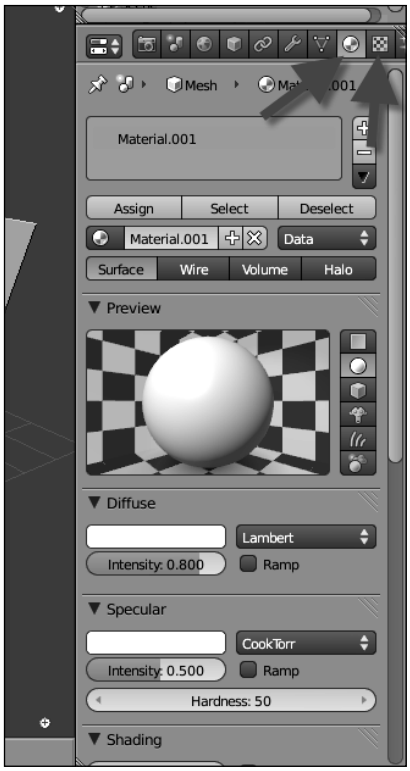


One of the biggest changes for the modeling process and architectural modeling will be the implementation of the new Mesh system for Blender named B-Mesh. This project will add support for N-Gons in Blender 2.50, allowing artists to use faces with more than four sides during mesh modeling. At this point, the B-Mesh is a separate project and is scheduled to be merged with Blender 2.50, but there are no guarantees that this project will be added to the final release.

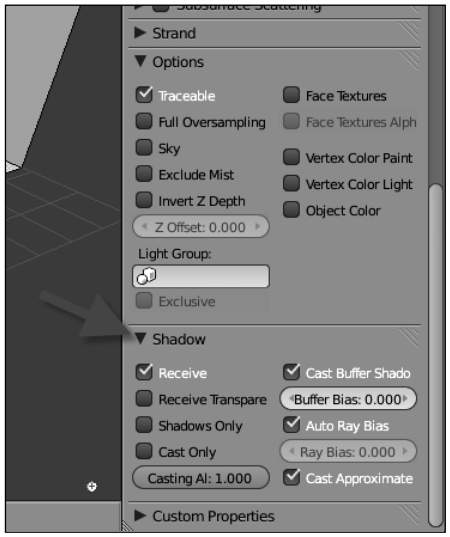
For architectural modeling based on Mesh objects, we can use almost the same tools and techniques from 2.49 in 2.50, with only a few adaptations needed for fine-tuning the modeling.

Materials and textures

There are two separate panels for materials and textures in Blender 2.50, and most of the tools and options from 2.49 were added to different categories and sections. To add a material or texture in Blender 2.50, we use the selector available at the top of each panel. Just press the + button, and a new material or texture is added to the scene. We can turn a material texture on or off, simply by pressing the checkbox located to the right side of the material or texture name:

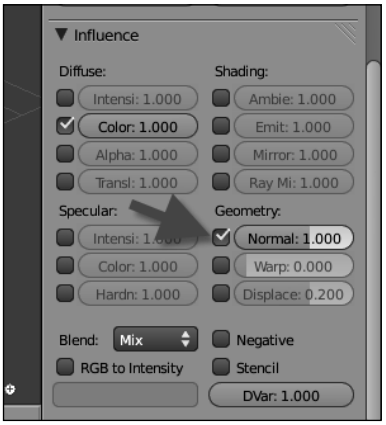


At a first glance we may find it confusing, but after a few minutes, the new sections will look more intuitive than the old ones from 2.49. For instance, all options related to the casting of shadows are grouped in a section named **Shadow**. If you remember, we had options related to shadows spread among shader options and other menus in 2.49:



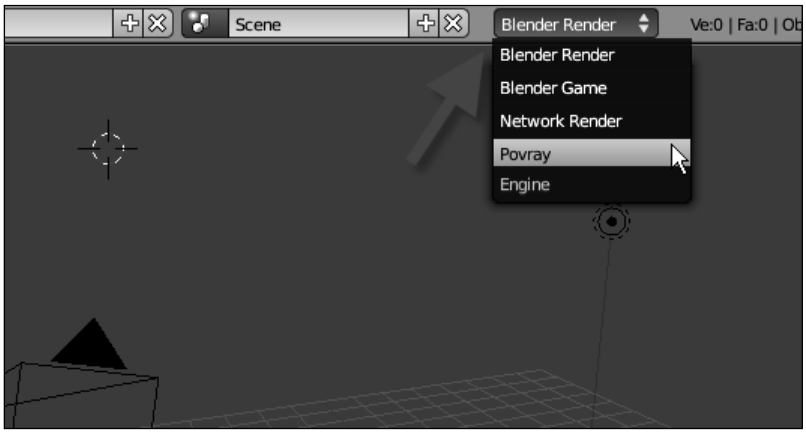
With the software still being developed, more tools and options may appear until a final release is available.

For textures, 2.50 works exactly the same as 2.49 with a reorganization of the old options into new sections and groups. For instance, the option to deform geometry using normal values was located in the **Materials** panel in 2.49, but now we find it under a **Geometry** section:

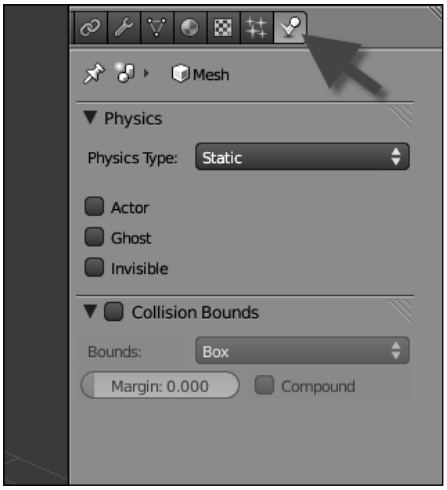


Rendering

The rendering panel of Blender 2.50 is represented by a small camera, and holds most of the options from Blender 2.49. Just the same as other panels from Blender 2.50, it's only a different way to present the same tools and options. But, we have a few important changes in the rendering process. First of all, the selection of the render engine used to generate the images or access the game engine is located at the top of the interface:



There, we will find options to change the render engine, and depending on the engine, some of the panels of Blender will change to fit the new context. For instance, by choosing the Game engine as the renderer, we will see options to use Rigid Body dynamics in the Physics panels that aren't available with **Blender Render**:

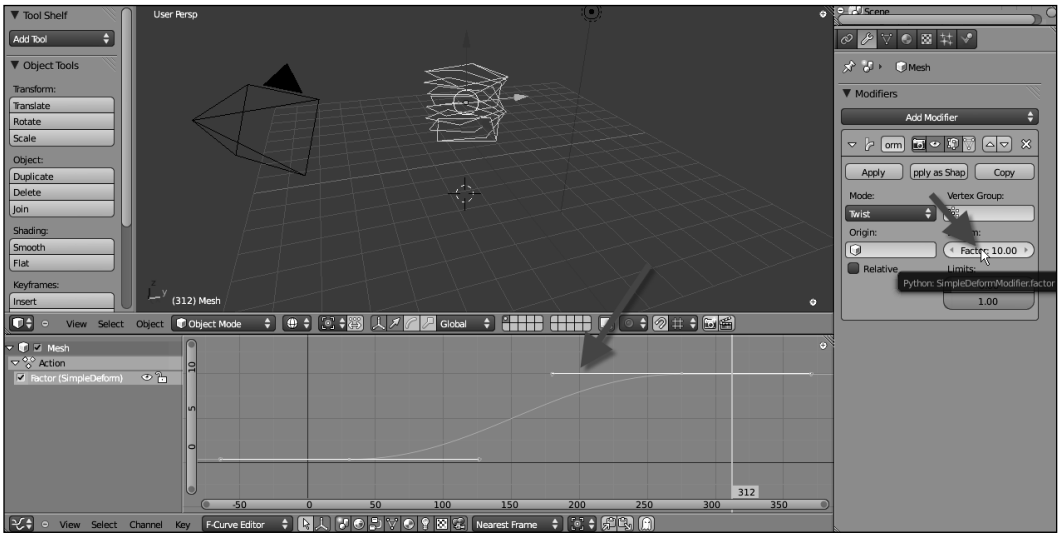


One thing is still uncertain about Blender 2.50; It's the integration with external renderers, such as YafaRay and LuxRender. So far, the API of Blender 2.50 is not finished, which prevents developers from creating a working exporter to use with any external renderers. This is one of the main reasons to delay migration to Blender 2.50 as a tool for architectural visualization projects, because we won't be able to use advanced global illumination features, such as Photon Mapping, Path Tracing, and so on.

Animation

This is one of the areas with many changes in Blender 2.50 starting with the managing of mesh deformations and other animation features. For architectural animation, one of the most significant changes is the new **Graph Editor**. Also, fact that everything in Blender can be animated now. For instance, if we have a simple modifier applied to a 3D model and want to animate the effect created by the modifier, one of the options in 2.49 was to use Hooks or Shape Keys.

Now, simply place the mouse cursor over the parameter that must be animated and press the *I* key to add a keyframe to it:



The possibilities for architectural animation and uses for staging animation are endless!

Summary

In this last chapter, we had a quick look at what will be the next big update in Blender. The upcoming Blender 2.50 will hopefully be released by the third quarter of 2010 and brings lots of new tools and a new user interface. For artists using Blender for architectural visualization, most of those changes won't affect the workflow of visualization projects. But, it's always a good idea to stay updated with the software with which we work.

One of the main concerns architectural visualization artists must have about Blender 2.50 is the background compatibility with old projects. If you open a file created with Blender 2.49 and edit it with 2.50 and try to save the file, you probably won't be able to open it again in 2.49. Another important point is the incredible number of scripts available for Blender 2.49, which could not be upgraded to 2.50, such as the scripts that export scenes for YafaRay or LuxRender. There might be some time between the release of Blender 2.50 and the updated scripts.

Here is a list of subjects learned in this chapter:

- The history and development of Blender 2.50
- The changes in the user interface
- How to divide and merge windows
- Using the modeling tools and options
- The new B-Mesh system
- How to manage and create materials and textures
- How to choose rendering options in 2.50
- Animation in 2.50

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Thank you for buying
Blender 3D 2.49 Architecture, Buildings, and Scenery

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

About Packt Open Source

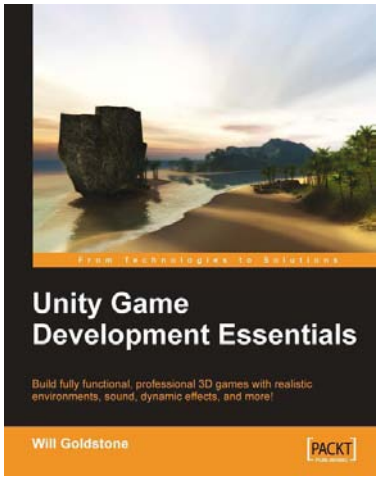
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
community experience distilled
PUBLISHING



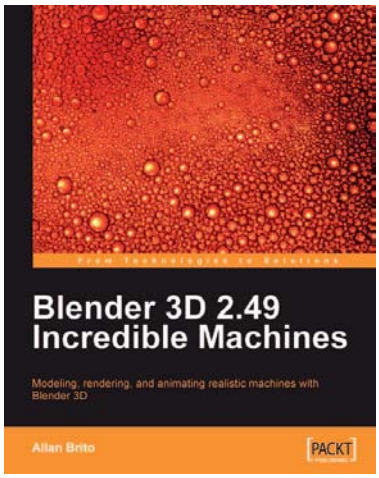
Unity Game Development Essentials

ISBN: 978-1-847198-18-1 Paperback: 316 pages

Create photorealistic 3D architectural visualizations of buildings, interiors, and environmental scenery

1. Kick start game development, and build ready-to-play 3D games with ease
2. Understand key concepts in game design including scripting, physics, instantiation, particle effects, and more
3. Test & optimize your game to perfection with essential tips-and-tricks

---



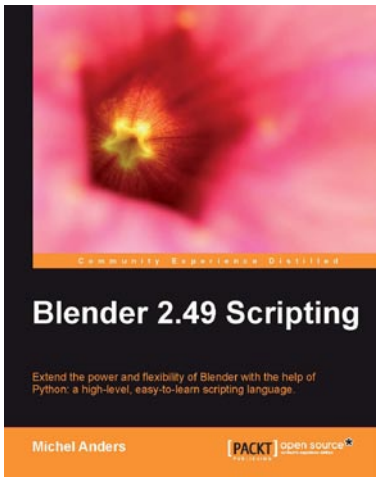
Blender 3D 2.49 Incredible Machines

ISBN: 978-1-847197-46-7 Paperback: 316 pages

Modeling, rendering, and animating realistic machines with Blender 3D

1. Walk through the complete process of building amazing machines
2. Model and create mechanical models and vehicles with detailed designs
3. Add advanced global illumination options to the renders created in Blender 3D using YafaRay and LuxRender
4. Create machines such as a handgun, a steam punk spacecraft, and a transforming robot

[PACKT] open source
community experience distilled
PUBLISHING

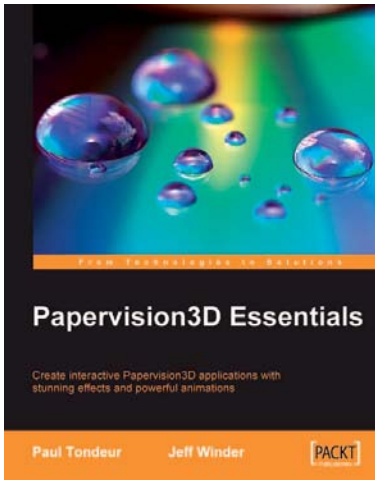


Blender 2.49 Scripting

ISBN: 978-1-849510-40-0 Paperback: 292 pages

Extend the power and flexibility of Blender with the help of the high-level, easy-to-learn scripting language, Python

1. Gain control of all aspects of Blender using the powerful Python language
2. Create complex meshes programmatically and apply materials and textures
3. Automate the rendering process and extend Blender's image manipulation capabilities
4. Extend Blender's built-in editor



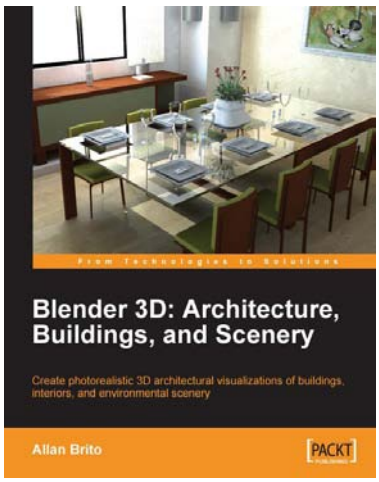
Papervision3D Essentials

ISBN: 978-1-847195-72-2 Paperback: 428 pages

Create interactive Papervision 3D applications with stunning effects and powerful animations

1. Build stunning, interactive Papervision3D applications from scratch
2. Export and import 3D models from Autodesk 3ds Max, SketchUp and Blender to Papervision3D
3. In-depth coverage of important 3D concepts with demo applications, screenshots and example code.
4. Step-by-step guide for beginners and professionals with tips and tricks based on the authors' practical experience

[PACKT] open source
community experience distilled
PUBLISHING

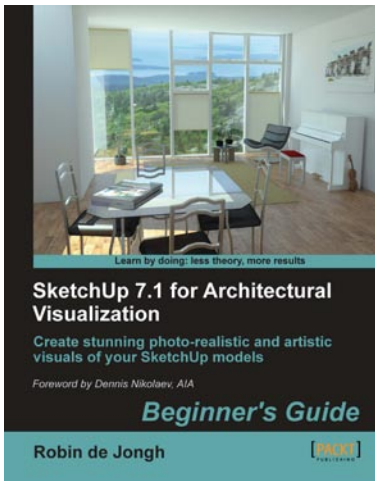


Blender 3D Architecture, Buildings, and Scenery

ISBN: 978-1-847193-67-4 Paperback: 332 pages

Create photorealistic 3D architectural visualizations of buildings, interiors, and environmental scenery

1. Turn your architectural plans into a model
2. Study modeling, materials, textures, and light basics in Blender
3. Create photo-realistic images in detail
4. Create realistic virtual tours of buildings and scenes



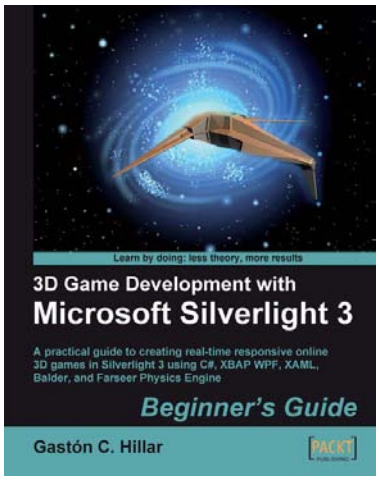
SketchUp 7.1 for Architectural Visualization: Beginner's Guide

ISBN: 978-1-847199-46-1 Paperback: 408 pages

Create stunning photo-realistic and artistic visuals for your SketchUp models

1. Create picture-perfect photo-realistic 3D architectural renders for your SketchUp models
2. Post-process SketchUp output to create digital watercolor and pencil art
3. Follow a professional visualization studio workflow
4. Make the most out of SketchUp with the best free plugins and add-on software to enhance your models

[PACKT] open source
community experience distilled
PUBLISHING



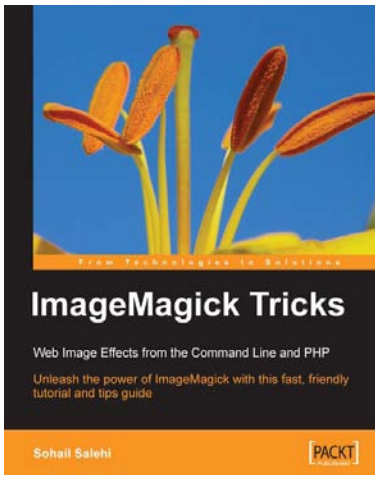
3D Game Development with Microsoft Silverlight 3

ISBN: 978-1-847198-92-1 Paperback: 452 pages

A practical guide to creating real-time responsive online 3D games in Silverlight 3 using C#, XBAP WPF, XAML, Balder, and Farseer Physics Engine

1. Develop online interactive 3D games and scenes in Microsoft Silverlight 3 and XBAP WPF
2. Integrate Balder 3D engine 1.0, Farseer Physics Engine 2.1, and advanced object-oriented techniques to simplify the game development process
3. Enhance development with animated 3D characters, sounds, music, physics, stages, gauges, and backgrounds

---



ImageMagick Tricks

ISBN: 978-1-904811-86-2 Paperback: 232 pages

Unleash the power of ImageMagick with this fast, friendly tutorial and tips guide

1. Complete tutorial and a gallery of tricks and techniques
2. Create impressive image manipulations and animations on-the-fly from the command line or within your programs
3. Complete PHP-based sample applications show how to use ImageMagick to add pizzazz your web site