

Blender 3D: Characters, Machines, and Scenes for Artists

Table of Contents

[Blender 3D: Characters, Machines, and Scenes for Artists](#)

[Blender 3D: Characters, Machines, and Scenes for Artists](#)

[Credits](#)

[Preface](#)

[What this learning path covers](#)

[What you need for this learning path](#)

[Who this learning path is for](#)

[Reader feedback](#)

[Customer support](#)

[Downloading the example code](#)

[Errata](#)

[Piracy](#)

[Questions](#)

[1. Module 1](#)

[1. Modeling the Character's Base Mesh](#)

[Introduction](#)

[Setting templates with the Images as Planes add-on](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Setting templates with the Image Empties method](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Setting templates with the Background Images tool](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Building the character's base mesh with the Skin modifier](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[2. Sculpting the Character's Base Mesh](#)

[Introduction](#)

[Using the Skin modifier's Armature option](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Editing the mesh](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Preparing the base mesh for sculpting](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Using the Multiresolution modifier and the Dynamic topology feature](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Sculpting the character's base mesh](#)

[Getting ready](#)

[How to do it...](#)

[There's more...](#)

[3. Polygonal Modeling of the Character's Accessories](#)

[Introduction](#)

[Preparing the scene for polygonal modeling](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Modeling the eye](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Modeling the armor plates](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Using the Mesh to Curve technique to add details](#)

[How to do it...](#)

[How it works...](#)

[4. Re-topology of the High Resolution Sculpted Character's Mesh](#)

[Introduction](#)

[Using the Grease Pencil tool to plan the edge-loops flow](#)

[Getting ready](#)

[How to do it...](#)

[There's more...](#)

[See also](#)

[Using the Snap tool to re-topologize the mesh](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Using the Shrinkwrap modifier to re-topologize the mesh](#)

[Getting ready](#)

[How to do it...](#)

[Using the LoopTools add-on to re-topologize the mesh](#)

[Getting ready](#)

[How to do it...](#)

Concluding the re-topologized mesh

Getting ready

How to do it...

There's more...

How it works...

5. Unwrapping the Low Resolution Mesh

Introduction

Preparing the low resolution mesh for unwrapping

Getting ready

How to do it...

UV unwrapping the mesh

Getting ready

How to do it...

Editing the UV islands

Getting ready

How to do it...

How it works...

There's more...

Using the Smart UV Project tool

Getting ready

How to do it...

Modifying the mesh and the UV islands

Getting ready

How to do it...

Setting up additional UV layers

Getting ready

How to do it...

Exporting the UV Map layout

Getting ready

How to do it...

6. Rigging the Low Resolution Mesh

Introduction

Building the character's Armature from scratch

Getting ready

How to do it...

Building the rig for the secondary parts

Completing the rig

How it works...

Perfecting the Armature to also function as a rig for the Armor

Getting ready

How to do it...

How it works...

Building the character's Armature through the Human Meta-Rig

Getting ready

How to do it...

How it works...

Building the animation controls and the Inverse Kinematic

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Generating the character's Armature by using the Rigify add-on](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[7. Skinning the Low Resolution Mesh](#)

[Introduction](#)

[Parenting the Armature and Mesh using the Automatic Weights tool](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Assigning Weight Groups by hand](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Editing Weight Groups using the Weight Paint tool](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Using the Mesh Deform modifier to skin the character](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Using the Laplacian Deform modifier and Hooks](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[8. Finalizing the Model](#)

[Introduction](#)

[Creating shape keys](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Assigning drivers to the shape keys](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Setting movement limit constraints](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Transferring the eyeball rotation to the eyelids](#)

[Getting ready](#)

[How to do it...](#)

[Detailing the Armor by using the Curve from Mesh tool](#)

[Getting ready](#)

[How to do it...](#)

[There's more...](#)

[See also](#)

[9. Animating the Character](#)

[Introduction](#)

[Linking the character and making a proxy](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Creating a simple walk cycle for the character by assigning keys to the bones](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Tweaking the actions in Graph Editor](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Using the Non Linear Action Editor to mix different actions](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[10. Creating the Textures](#)

[Introduction](#)

[Making a tileable scales image in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Preparing the model to use the UDIM UV tiles](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Baking the tileable scales texture into the UV tiles](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[Painting to fix the seams and to modify the baked scales image maps](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Painting the color maps in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Painting the color maps in Cycles](#)

[Getting ready](#)

[How to do it...](#)

[11. Refining the Textures](#)

[Introduction](#)

[Sculpting more details on the high resolution mesh](#)

[Getting ready](#)

[How to do it...](#)

[Baking the normals of the sculpted mesh on the low resolution one](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[The Armor textures](#)

[Getting ready](#)

[How to do it...](#)

[There's more...](#)

[See also](#)

[Adding a dirty Vertex Colors layer and baking it to an image texture](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[The Quick Edit tool](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[12. Creating the Materials in Cycles](#)

[Introduction](#)

[Building the reptile skin shaders in Cycles](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Making a node group of the skin shader to reuse it](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Building the eyes' shaders in Cycles](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Building the armor shaders in Cycles](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[13. Creating the Materials in Blender Internal](#)

[Introduction](#)

[Building the reptile skin shaders in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Building the eyes' shaders in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[Building the armor shaders in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[14. Lighting, Rendering, and a Little Bit of Compositing](#)

[Introduction](#)

[Setting the library and the 3D scene layout](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Setting image based lighting \(IBL\)](#)

[Getting ready](#)

[How to do it...](#)

[Image based lighting in Cycles](#)

[Image based lighting in Blender Internal](#)

[How it works...](#)

[See also](#)

[Setting a three-point lighting rig in Blender Internal](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[Rendering an OpenGL playblast of the animation](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[There's more...](#)

[See also](#)

[Obtaining a noise-free and faster rendering in Cycles](#)

[Getting ready](#)

[How to do it...](#)

[See also](#)

[Compositing the render layers](#)

[Getting ready](#)

[How to do it...](#)

[How it works...](#)

[See also](#)

[2. Module 2](#)

[1. Sci-Fi Pistol - Creating the Basic Shapes](#)

[A project overview](#)

[Creating the barrel](#)

[Modeling a handgrip and other pieces](#)

[Summary](#)

[2. Sci-Fi Pistol - Adding Details](#)

[Finishing the handgrip](#)

[Cutting shapes into our gun](#)

[Circles, angles, and edge splits](#)

[Adding final details](#)

[General workflow for detailing](#)

[Summary](#)

[3. Texturing and Rendering Your Sci-Fi Pistol](#)

[Preparing our scene](#)

[Enabling GPU rendering](#)

[Adding materials and textures](#)

[Summary](#)

[4. Spacecraft – Creating the Basic Shapes](#)

[Shaping the body](#)

[Creating the accessories](#)

[Summary](#)

[5. Spacecraft - Adding Details](#)

[Refining our ship](#)

[Adding detail with pipes](#)

[Finishing our main objects](#)

[Do it yourself - completing the body](#)

[Building the landing gear](#)

[Adding the cockpit](#)

[Creating small details](#)

[Working with Path objects](#)

[Finishing touches](#)

[Summary](#)

[6. Spacecraft – Materials, Textures, and Rendering](#)

[Preparing the model and scene](#)

[Creating materials](#)

[Adding decals with UV maps](#)

[Other possibilities](#)

[Summary](#)

[7. Modeling Your Freestyle Robot](#)

[Modeling for Freestyle](#)

[Blocking out your robot](#)

[Creating the body with Subdivision Surfacing](#)

[Finishing the robot](#)

[Summary](#)

[8. Robot - Freestyle Rendering](#)

[Preparing the scene](#)

[Marking Freestyle edges](#)

[Creating materials](#)

[Rendering and material options](#)

[Summary](#)

[9. Low-Poly Racer – Building the Mesh](#)

[Building for external applications](#)

[Starting the truck model](#)

[Manifold versus non-manifold meshes](#)

[Adding details](#)

[Summary](#)

[10. Low-Poly Racer – Materials and Textures](#)

[Texturing workflow](#)

[Adding materials](#)

[Unwrapping and baking](#)

[Adding detail in an Image Editor](#)

[Checking the Texture Map](#)

[Summary](#)

[1. All About Packt](#)

[Thank you for buying Blender 3D Incredible Machines](#)

[About Packt Publishing](#)

[About Packt Open Source](#)

[Writing for Packt](#)

[3. Module 3](#)

[1. Robot Toy – Modeling of an Object](#)

[Let's start the modeling of our robot toy](#)

[Preparing the workflow by adding an image reference](#)

[Adding the head primitive](#)

[The Edit Mode versus the Object Mode](#)

[Using the basic modeling tools](#)

[Modeling the head](#)

[Modeling the antenna](#)

[An introduction to the Subdivision Surface modifier](#)

[Improving the head shape](#)

[Modeling the thunderbolts](#)

[Modeling the eyes](#)

[Modeling the chest](#)

[Modeling the neck](#)

[Modeling the torso](#)

[Modeling the buttons](#)

[Modeling the fork](#)

[Modeling protections for the fork](#)

[Modeling the main wheel](#)

[Modeling the arm](#)

[Using Blender Internal to render our Robot Toy](#)

[Summary](#)

[2. Alien Character – Base Mesh Creation and Sculpting](#)

[Understanding the sculpting process](#)

[An introduction to sculpting](#)

[Choosing sculpting over poly modeling](#)

[Using a pen tablet](#)

[The sculpt mode](#)

[Optimizing the viewport](#)

[Anatomy of a brush](#)

[Dyntopo versus the Multires modifier](#)

[First touch with the Multires modifier](#)

[First touch with Dyntopo](#)

[Creating a base mesh with the Skin modifier](#)

[Visual preparation](#)

[An introduction to artistic anatomy](#)

[Sculpting the body](#)

[The head](#)

[The torso](#)

[The arms](#)

[The legs](#)

[The belt](#)

[Summary](#)

[3. Alien Character – Creating a Proper Topology and Transferring the Sculpt Details](#)

[Why make a retopology?](#)

[Possibilities of arranging polygons](#)

[Errors to avoid during the creation of retopology](#)

[Density of polygons](#)

[Making the retopology of the alien character](#)

[Preparing the environment](#)

[The head](#)

[The neck and the torso](#)

[The arms and the hands](#)

[The legs](#)

[Unwrapping UVs](#)

Understanding UVs

The placement of the seams

The placement and adjustment of the islands

The baking of textures

The baking of a normal map

What is a normal map?

Making of the bake

Displaying the normal map in the viewport

The baking of an ambient occlusion

Understanding the ambient occlusion map

Creation of the bake

Displaying the ambient occlusion in the viewport

Summary

4. Haunted House – Modeling of the Scene

Blocking the house

Working with a scale

Blocking the bases of the house

Refining the blocking

Adding instantiated objects

Reworking the blocking objects

Breaking and ageing the elements

Simulate a stack of wooden planks with physics

Creation of the simulation of a stack of planks

Modeling the environment (8 pages)

Modeling the cliff

Modeling a tree with curves

Enhancing the scene with a barrier, rocks, and a cart

Organizing the scene

Grouping objects

Working with layers

Summary

5. Haunted House – Putting Colors on It

Unwrapping UVs

Using Project From View

Unwrapping the rest of the house

The tree with the Smart UV Project

Unwrapping the rest of the environment

Tiling UVs

What is tiling for?

The UV layers

Adding colors

Basics of the Texture Paint tool

Discovering the brushes

The TexDraw brush

The Smear brush

The Soften brush

The Clone brush

[The Fill brush](#)
[The Mask brush](#)
[The Stroke option](#)
[Delimiting the zones of painting according to the geometry](#)
[Painting directly on the texture](#)
[Painting the scene](#)
[Laying down the colors](#)
[Tiled textures](#)
[The settings of our workspace](#)
[Advice for a good tiled texture](#)
[Painting the roof-tile texture](#)
[Quick tips for other kinds of hand-painted tiled textures](#)
[Baking our tiled textures](#)
[Why bake?](#)
[How to do it?](#)
[Creating transparent textures](#)
[The grass texture](#)
[The grunge texture](#)

[Doing a quick render with Blender Internal](#)
[Setting lights](#)
[Placing the camera](#)
[Setting the environment \(sky and mist\)](#)

[Summary](#)

[6. Haunted House – Adding Materials and Lights in Cycles](#)

[Understanding the basic settings of Cycles](#)
[The sampling](#)
[Light path settings](#)
[Performances](#)
[Lighting](#)
[Creating a testing material](#)
[Understanding the different types of light](#)
[Lighting our scene](#)
[Painting and using an Image Base Lighting](#)
[Creating materials with nodes](#)
[Creating the materials of the house, the rocks, and the tree](#)
[Adding a mask for the windows](#)
[Using procedural textures](#)
[Making and applying normal maps in Cycles](#)

[Creating realistic grass](#)
[Generating the grass with particles](#)
[Creating the grass shader](#)

[Baking textures in Cycles](#)
[Cycles versus Blender Internal](#)
[Baking the tree](#)

[Compositing a mist pass](#)
[Summary](#)

[7. Rat Cowboy – Learning To Rig a Character for Animation](#)

[An introduction to the rigging process](#)

[Rigging the Rat Cowboy](#)

[Placing the deforming bones](#)

[The leg and the foot](#)

[The arm and the hand](#)

[The hips](#)

[The tail](#)

[The head and the eyes](#)

[Mirroring the rig](#)

[Rigging the gun](#)

[Rigging the holster](#)

[Adding a root bone](#)

[Skinning](#)

[The Weight Paint tools](#)

[Manually assigning weight to vertices](#)

[Correcting the foot deformation](#)

[Correcting the belt deformation](#)

[Custom shapes](#)

[The shape keys](#)

[What is a shape key?](#)

[Creating basic shapes](#)

[Driving a shape key](#)

[Summary](#)

[8. Rat Cowboy – Animate a Full Sequence](#)

[Principles of animation](#)

[Squash and Stretch](#)

[Anticipation](#)

[Staging](#)

[Straight Ahead Action and Pose to Pose](#)

[Follow Through and Overlapping Action](#)

[Slow In and Slow Out](#)

[Arcs](#)

[Secondary Action](#)

[Timing](#)

[Exaggeration](#)

[Solid drawing](#)

[Appeal](#)

[Animation tools in Blender](#)

[The timeline](#)

[What is a keyframe?](#)

[The Dope Sheet](#)

[The Graph editor](#)

[The Non-Linear Action editor](#)

[Preparation of the animation](#)

[Writing a short script](#)

[Making a storyboard](#)

[Finding the final camera placements and the timing through a layout](#)

[Animation references](#)

[Organization](#)

[Animating the scene](#)

[The walk cycle](#)

[Mixing actions](#)

[Animation of a close shot](#)

[Animation of the gunshot](#)

[Animation of the trap](#)

[Render a quick preview of a shot](#)

[Summary](#)

[9. Rat Cowboy – Rendering, Compositing, and Editing](#)

[Creating advanced materials in Cycles](#)

[Skin material with Subsurface Scattering](#)

[Eye material](#)

[The fur of the rat](#)

[The Raw rendering phase](#)

[Enhance a picture with compositing](#)

[Introduction to nodal compositing](#)

[Depth Pass](#)

[Adding effects](#)

[Compositing rendering phase](#)

[Editing the sequence with the VSE](#)

[Introduction to the Video Sequence Editor](#)

[Edit and render the final sequence](#)

[Summary](#)

[A. Bibliography](#)

[Index](#)

Blender 3D: Characters, Machines, and Scenes for Artists

Blender 3D: Characters, Machines, and Scenes for Artists

Gain the insights and techniques you need to give life to your own custom characters, machines, and scenes in Blender 3D

A course in three modules



BIRMINGHAM - MUMBAI

Blender 3D: Characters, Machines, and Scenes for Artists

Copyright © 2016 Packt Publishing

All rights reserved. No part of this course may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this course to ensure the accuracy of the information presented. However, the information contained in this course is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this course.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this course by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Published on: October 2016

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-78712-966-5

www.packtpub.com

Credits

Authors	Content Development Editor
Enrico Valenza	Trusha Shriyan
Christopher Kuhn	
Romain Caudron	Kirk D'Phena
Pierre-Armand Nicq	
Reviewers	Production Coordinator
Luo Congyi	Deepika Naik
Waqas Abdul Majeed	
Reynante M. Martinez	
Jordan Matelsky	
Ian Smithers	
Jacek Herman	
Fernando Castilhos Melo	

Preface

Welcome to Blender 3D: Characters, Machines, and Scenes for Artists!

Blender 3D is one of the top pieces of 3D animation software. Machine modeling is an essential aspect of war games, space games, racing games, and animated action films. As the Blender software grows more powerful and popular, there is a demand to take your modeling skills to the next level.

Let your creative freedom begin!

What this learning path covers

[Module 1](#), *Blender 3D Cookbook*, will take you on a journey to understand the workflow normally used to create characters, from the modeling to the rendering stages, using the tools of the last official release of Blender exclusively. With the help of this module, you will be making production-quality 3D models and characters quickly and efficiently, which will be ready to be added to your very own animated feature or game.

[Module 2](#), *Blender 3D Incredible Machines*, will help you develop a comprehensive skill set that covers the key aspects of mechanical modeling. Through this module, you will create many types of projects, including a pistol, spacecraft, robot, and a racer. By the end of this module, you will have mastered a workflow that you will be able to apply to your own creations.

[Module 3](#), *Blender 3D By Example*, will help you to create many types of projects using a step-by-step approach. Each project in this module will give you more practice and increase your knowledge of the Blender tools and Blender game engine.

What you need for this learning path

A desktop or laptop computer (recommend at least 4 GB of RAM).

Windows 7, Mac OS X, or Linux.

The Blender 3D software package, free to download at www.blender.org.

Who this learning path is for

This learning path is for those who know the basics of Blender and have hands-on experience with the software. We will directly dive into creating characters first. If you wish to use Blender to create games, animated films, and architecture simulations, this learning path will benefit you.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this course—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail <feedback@packtpub.com>, and mention the course's title in the subject of your message.

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

Customer support

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

Downloading the example code

You can download the example code files for this course from your account at <http://www.packtpub.com>. If you purchased this course elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the course in the **Search** box.
5. Select the course for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this course from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the course's webpage at the Packt Publishing website. This page can be accessed by entering the course's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the course is also hosted on GitHub at <https://github.com/PacktPublishing/Blender-3D-Characters-Machines-and-Scenes-for-Artists>. We also have other code bundles from our rich catalog of books, videos, and courses available at <https://github.com/PacktPublishing/>. Check them out!

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 courses—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this course. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your course, 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 to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the course in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted 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

If you have a problem with any aspect of this course, you can contact us at <questions@packtpub.com>, and we will do our best to address the problem.

Part 1. Module 1

Blender 3D Cookbook

Build your very own stunning characters in Blender from scratch

Chapter 1. Modeling the Character's Base Mesh

In this chapter, we will cover the following recipes:

- Setting templates with the Images as Planes add-on
- Setting templates with the Image Empties method
- Setting templates with the Background Images tool
- Building the character's base mesh with the Skin modifier

Introduction

In this chapter, we are going to do two things: set up templates to be used as a reference for the modeling, and build up a base mesh for the sculpting of the character.

To set up templates in a Blender scene, we have at least three different methods to choose from: the **Images as Planes** add-on, the **Image Empties** method, and the **Background Images** tool.

A base mesh is usually a very low poly and simple mesh roughly shaped to resemble the final character's look. There are several ways to obtain a base mesh: we can use a ready, freely downloadable mesh to be adjusted to our goals, or we can model it from scratch, one polygon at a time. What's quite important is that it should be made from all quad faces.

To build the base mesh for our character, we are going to use one of the more handy and useful modifiers added to Blender: the **Skin** modifier. However, first, let us add our templates.

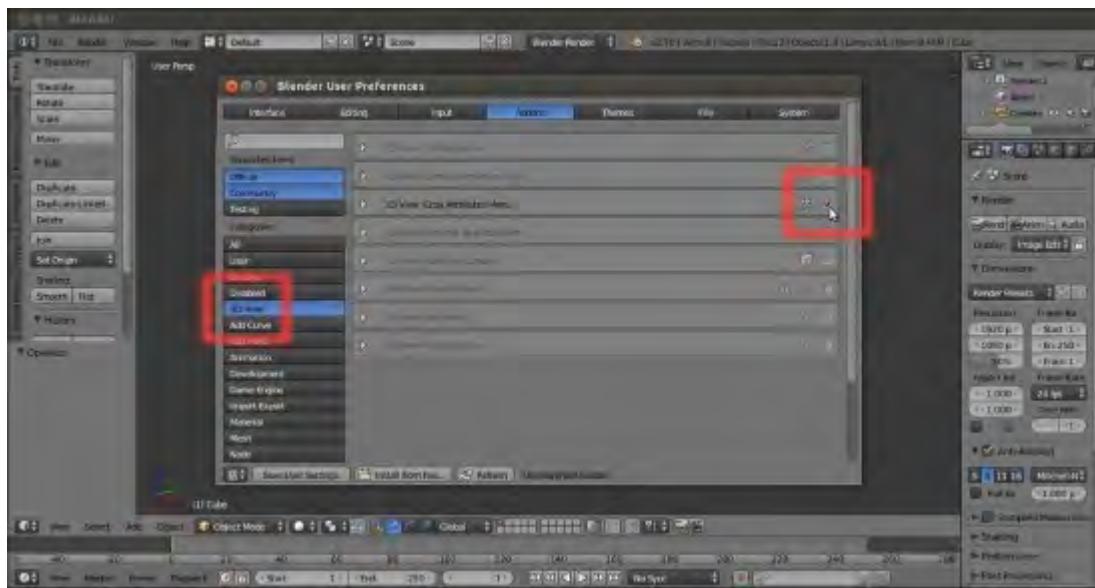
Setting templates with the Images as Planes add-on

In this recipe, we'll set the character's templates by using the **Images as Planes** add-on.

Getting ready

The first thing to do is to be sure that all the required add-ons are enabled in the preferences; in this first recipe, we need the **Images as Planes** and **Copy Attributes Menu** add-ons. When starting Blender with the factory settings, they appear gray in the **User Preferences** panel's **Add-ons** list, meaning that they are not enabled yet. So, we'll do the following:

1. Call the **User Preferences** panel (*Ctrl + Alt + U*) and go to the **Add-ons** tab.
2. Under the **Categories** item on the left-hand side of the panel, click on **3D View**.
3. Check the empty little checkbox on the right-hand side of the **3D View: Copy Attributes Menu** add-on to enable it.
4. Go back to the **Categories** item on the left-hand side of the panel and click on **Import-Export**.
5. Scroll down the add-ons list to the right-hand side to find the **Import-Export: Import Images as Planes** add-on (usually, towards the middle of the long list).
6. Enable it, and then click on the **Save User Settings** button to the left-bottom of the panel and close it.



The User Preferences panel with the Categories list and the Addons tab to enable the several add-ons

There are still a few things we should do to prepare the 3D scene and make our life easier:

7. Delete the already selected **Cube** primitive.

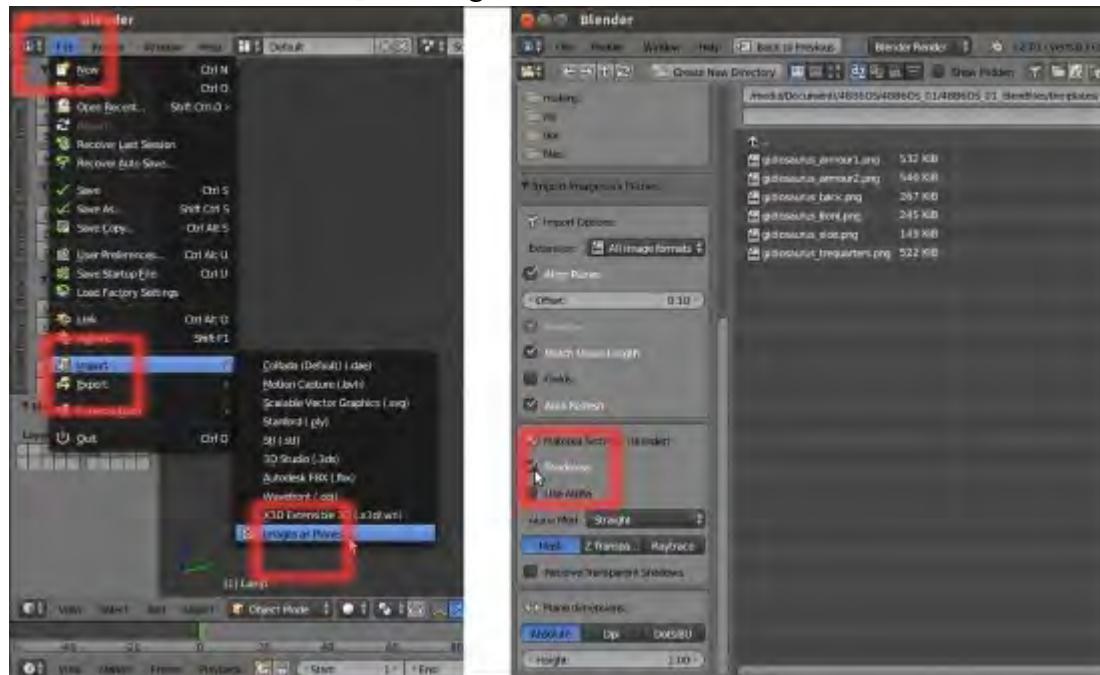
8. Select the **Lamp** and the **Camera** and move them on to a different layer; I usually have them on the **sixth** layer (*M* key), in order to keep free and empty both the first and second rows of the left layer's block.
9. The **Outliner** can be found in the top-right corner of the default workspace. It shows a list view of the scene. Set **Display Mode** of the **Outliner** to **Visible Layers**.
10. Lastly, save the file as `Gidiosaurus_base_mesh.blend`.

How to do it...

Although not strictly necessary, it would be better to have the three (at least in the case of a biped character, the **Front**, **Side**, and **Back** view) templates as separated images. This will allow us to load a specific one for each view, if necessary. Also, to facilitate the process, all these images should be the same height in pixels.

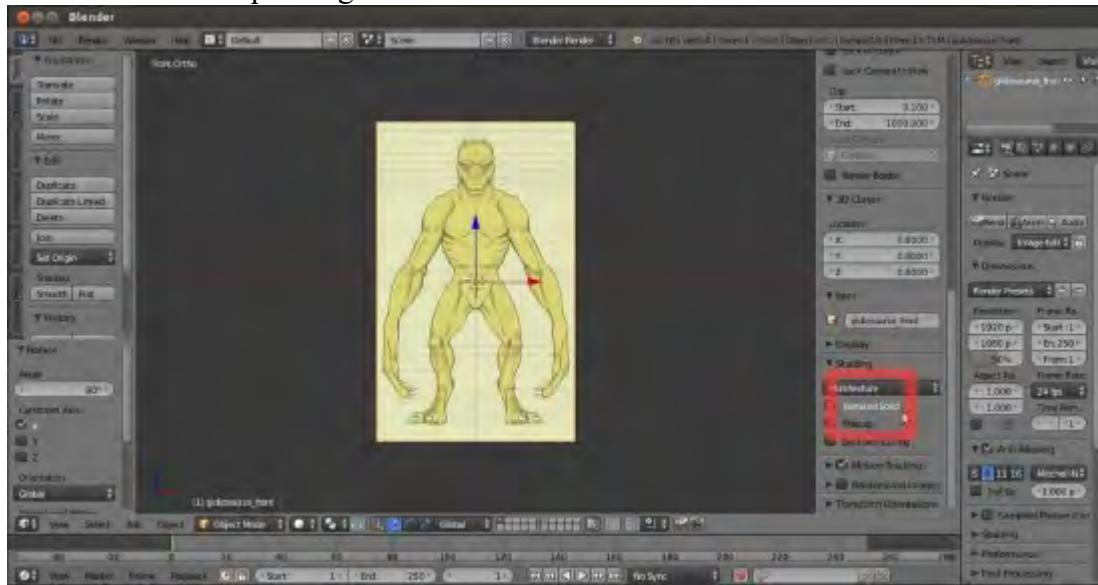
In our case, the required three views are provided for you in the files that accompany this book. You will find them in the `templates` folder. The **Import Images as Planes** add-on will take care of loading them into the scene:

1. Left-click on **File | Import | Images as Planes** in the top-left menu on the main header of the Blender UI.
2. On the page that just opened, go to the **Material Settings** column on the left-hand side (under the **Import Images as Planes** options) and enable the **Shadeless** item. Then, browse to the location where you placed your `templates` folder and load the `gidiosaurus_front.png` image:



The Import pop-up menu and the material settings subpanel of the Import Images as Planes add-on

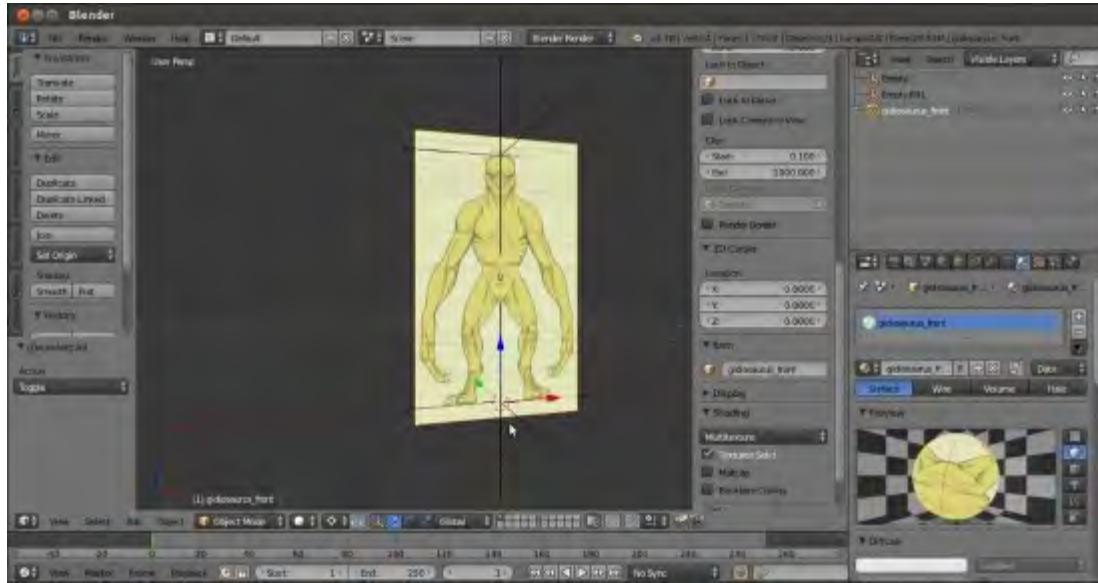
3. Rotate 90 degrees on the x axis ($R \mid X \mid 90 \mid Enter$) of the **Plane** that just appeared at the center of the scene (at the **3D Cursor** location, actually; to reset the position of the **3D Cursor** at the center of the scene, press the *Shift + C* keys).
4. Press *N* to call the **Properties** sidepanel on the right-hand side of the active 3D window, and then go to the **Shading** subpanel and enable the **Textured Solid** item.
5. Press *I* on the numpad to go to the **Front** view:



The imported plane with the relative UV-mapped image

Now, we know that our **Gidiosaurus** is a **2.5** meters tall beast. So, assuming that **1 Blender Unit** is equal to **1 meter**, we must scale the plane to make the character's front template **two and a half Blender Units** tall (Note that it is not the plane that must be 2.5 units tall, it's the character's shape inside the plane).

6. Add an **Empty** to the scene (*Shift + A* | **Empty** | **Plain Axes**).
7. Duplicate it and move it **2.5** units up on the z axis (*Shift + D* | $Z \mid 2.5 \mid Enter$).
8. Go to the **Outliner** and click on the arrows on the side of the names of the two **Empties** (**Empty** and **Empty.001**), in order to make them gray and the **Empties** not selectable.
9. Select the **Plane** and move it to align the bottom (feet) guideline to the horizontal arm of the first **Empty** (you actually have to move it on the z axis by **0.4470**, but note that by pressing the *Ctrl* key, you can restrict movements to the grid and with *Ctrl + Shift*, you can have even finer control).
10. Be sure that the **3D Cursor** is at the object origin, and press the period key to switch *Pivot center for rotation/scaling* to the **3D Cursor**.
11. Press *S* to scale the **Plane** bigger and align the top-head guideline to the horizontal arm of the second **Empty** (you have to scale it to a value of **2.8300**):



The properly scaled plane in the 3D scene

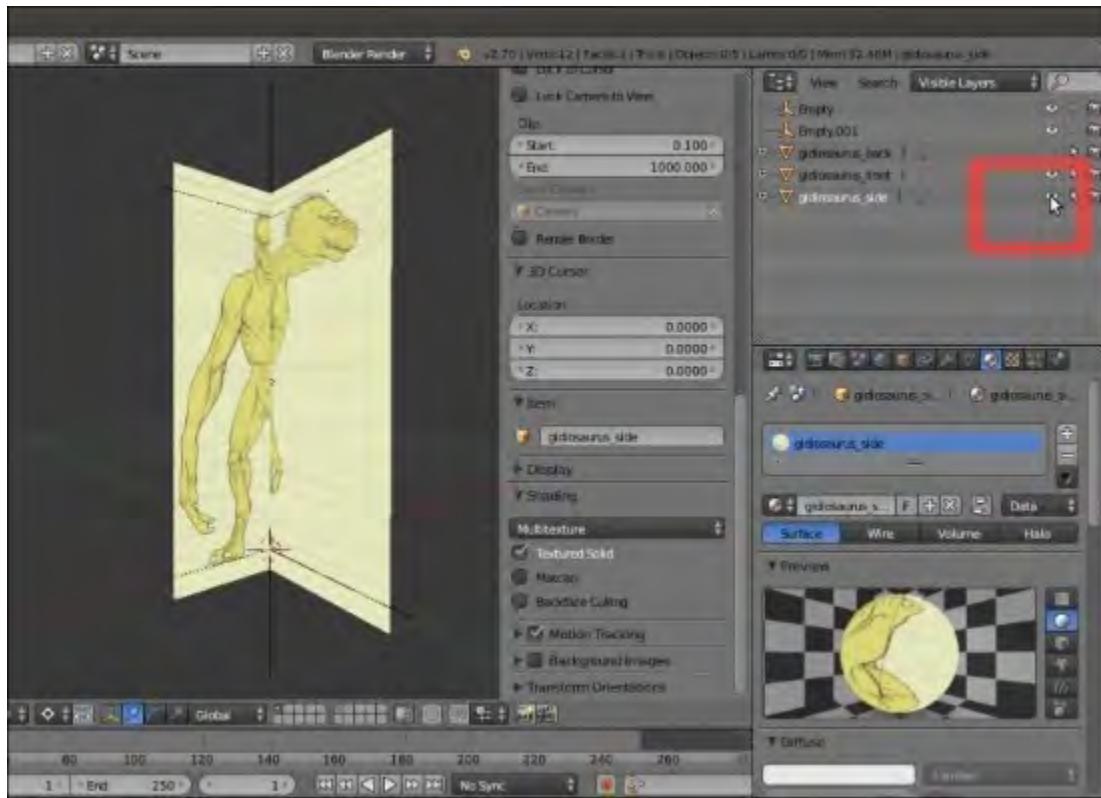
12. Left-click again on **File | Import | Images as Planes** in the top-left menu on the main header of the Blender UI.
13. Browse to the location where you placed your `templates` folder and this time load the `gidiosaurus_side.png` image.
14. *Shift + right-click* on the first **Plane** (`gidiosaurus_front.png`) to select it and make it the active one. Then, press *Ctrl + C* and from the **Copy Attributes** pop-up menu, select **Copy Location**.
15. Press *Ctrl + C* again and this time select **Copy Rotation**; press *Ctrl + C* one more time and select **Copy Scale**.
16. Right-click to select the second **Plane** (`gidiosaurus_side.png`) in the 3D view, or click on its name in the **Outliner**, and rotate it 90 degrees on the *z* axis (*R | Z | 90 | Enter*).
17. Optionally, you can move the second **Plane** to the second layer (*M* | second button on the **Move to Layer** panel).
18. Again, left-click on **File | Import | Images as Planes**, browse to the `templates` folder, and load the `gidiosaurus_back.png` image.
19. Repeat from step 12 to step 15 and move the third **Plane** on a different layer.
20. Save the file.

How it works...

We used a Python script, which is an add-on, to import planes into our scene that are automatically UV-mapped with the selected image, and inherit the images' height/width aspect ratio.

To have the textures/templates clearly visible from any angle in the 3D view, we have enabled the **Shadeless** option for the **Planes** materials; we did this directly in the importer preferences. We can also set each material to shadeless later in the **Material** window.

We then used another add-on to copy the attributes from a selected object, in order to quickly match common parameters such as location, scale, and rotation:



The template planes aligned to the x and y axis (Front and Side views)

The imported **Planes** can be placed on different layers for practicality; they can also be on a single layer and their visibility can be toggled on and off by clicking on the eye icon in the **Outliner**.

Setting templates with the Image Empties method

In this recipe, we'll set the character's templates by using **Image Empties**.

Getting ready

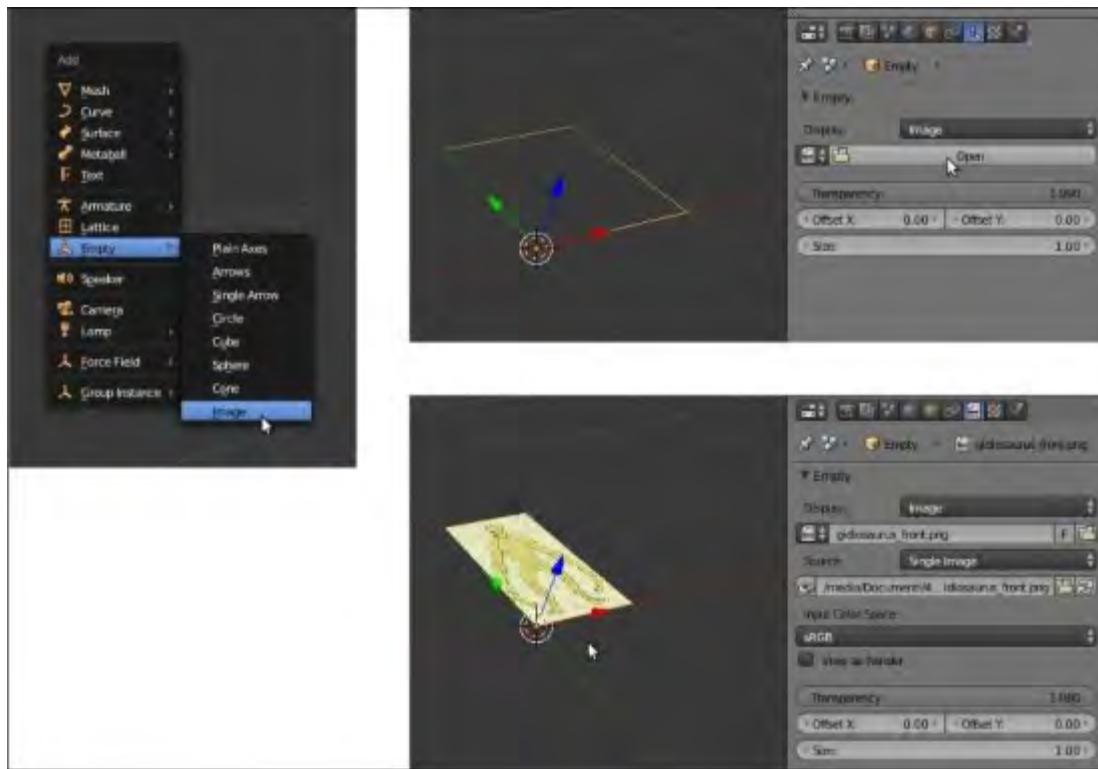
For this and the following recipes, there is no need for any particular preparations. Anyway, it is handy to prepare the two **Empties** to have markers in the 3D view for the **2.5** meters height of the character; so we'll do the following:

1. Start a brand new Blender session and delete the already selected **Cube** primitive.
2. Select the **Lamp** and **Camera** and move them on a different layer; I usually have them on the **sixth** layer, in order to keep free and empty both the first and second rows of the left layer's block.
3. Add an **Empty** to the scene (*Shift + A | Empty | Plain Axes*).
4. Duplicate it and move it **2.5** units up on the *z* axis (*Shift + D | Z | 2.5 | Enter*).
5. Go to the **Outliner** and click on the arrows on the side of the names of the two **Empties** (**Empty** and **Empty.001**), in order to make them gray and the **Empties** not selectable.
6. Save the file as `Gidiosaurus_base_mesh.blend`.

How to do it...

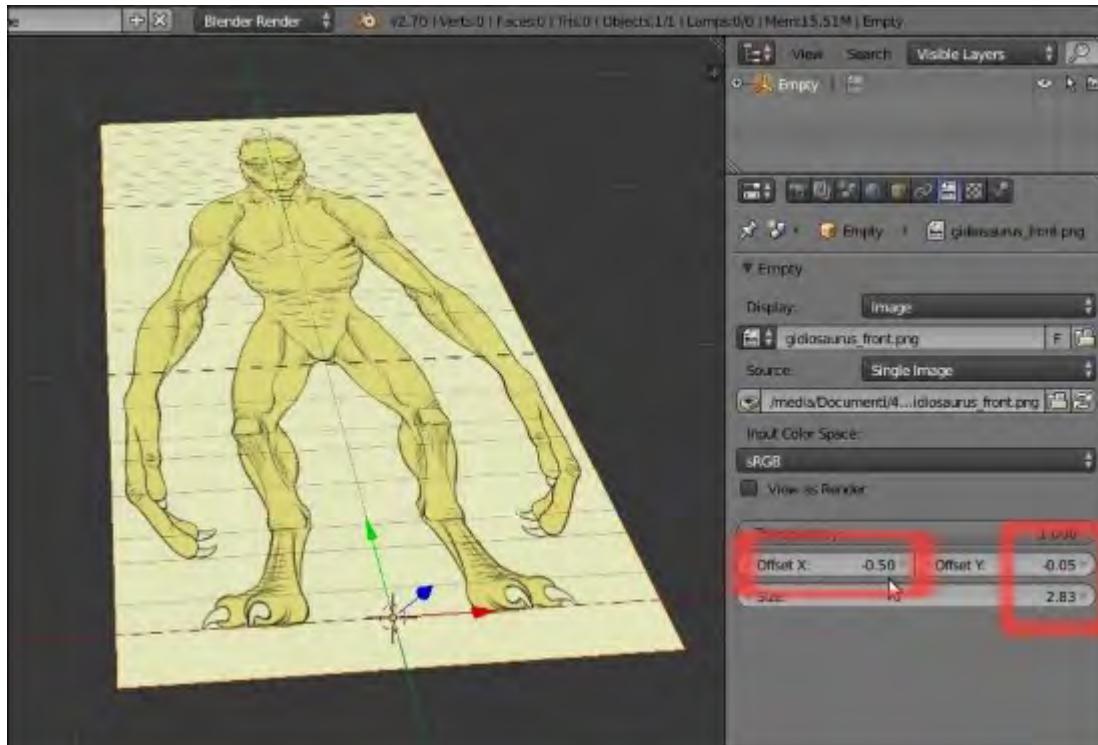
So, now we are going to place the first **Image Empty** in the scene:

1. Add an **Empty** to the scene (*Shift + A | Empty | Image*; it's the last item in the list).
2. Go to the **Object Data** window in the main **Properties** panel on the right-hand side of the Blender UI; under the **Empty** subpanel, click on the **Open** button.
3. Browse to the **templates** folder and load the `gidiosaurus_front.png` image.



The Add pop-up menu and the Image Empty added to the 3D scene, with the settings to load and set the image

- Set the Offset X value to **-0.50** and Offset Y to **-0.05**. Set the Size value to **2.830**:

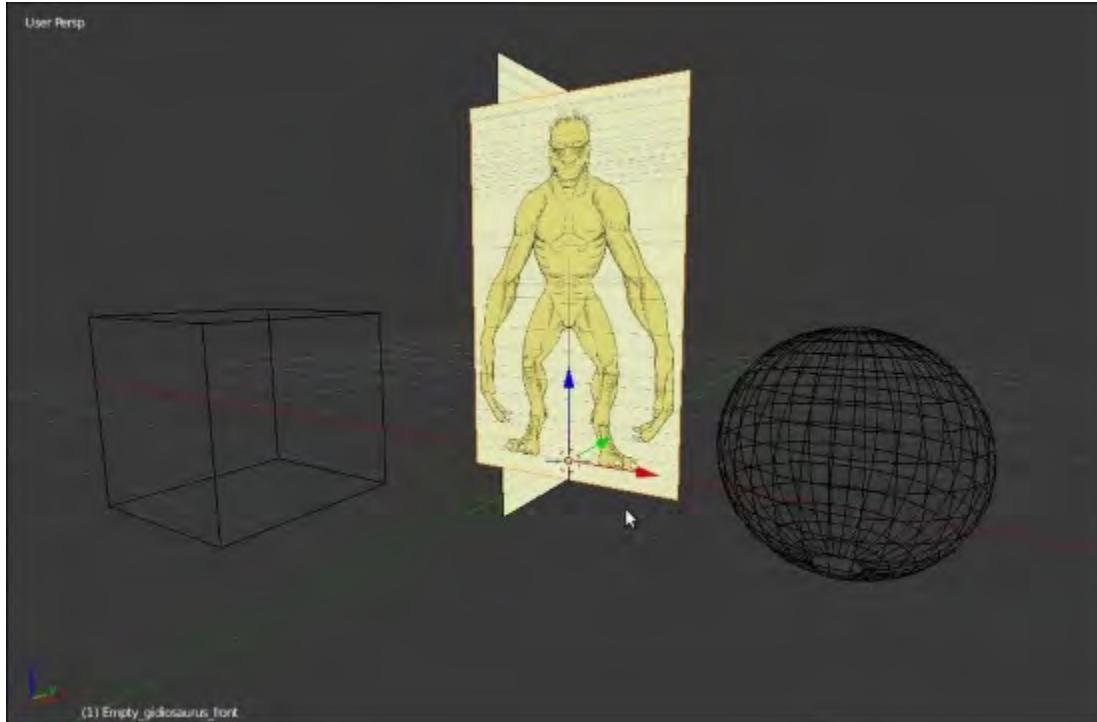


The Offset and Size settings

5. Rotate the **Empty** 90 degrees on the *x* axis (*R | X | 90 | Enter*).
6. Go to the **Outliner** and rename it **Empty_gidiosaurus_front**.
7. Duplicate it (*Shift + D*), rotate it 90 degrees on the *z* axis, and in the **Outliner**, rename it as **Empty_gidiosaurus_side**.
8. In the **Empty** subpanel under the **Object Data** window, click on the little icon (showing **3** users for that data block) on the right-hand side of the image name under **Display**, in order to make it a single user. Then, click on the little folder icon on the right-hand side of the image path to go inside the `templates` folder again, and load the `gidiosaurus_side.png` image.
9. Reselect **Empty_gidiosaurus_front** and press *Shift + D* to duplicate it.
10. Go to the **Empty** subpanel under the **Object Data** window, click on the little icon (showing **3** users for that datablock) on the right-hand side of the image name under **Display**, in order to make it a single user. Then, click on the little folder icon on the right-hand side of the image path to go inside the `templates` folder again, and this time load the `gidiosaurus_back.png` image.
11. Go to the **Outliner** and rename it **Empty_gidiosaurus_back**.

How it works...

We have used one of the most underrated (well, in my opinion) tools in Blender: **Empties**, which can show images! Compared to the **Images as Planes** add-on, this has some advantages: these are not 3D geometry and the images are also visible in the 3D view without the **Textured Solid** option enabled (under **Shading**) and in **Wireframe** mode.



The Image Empties appear as textured also in Wireframe viewport shading mode

Exactly, as for the imported **Planes** of the former recipe, the visibility in the 3D view of the **Image Empties** can be toggled on and off by clicking on the eye icon in the **Outliner**.

Setting templates with the Background Images tool

In this recipe, we'll set the character's templates by using the **Background Images** tool.

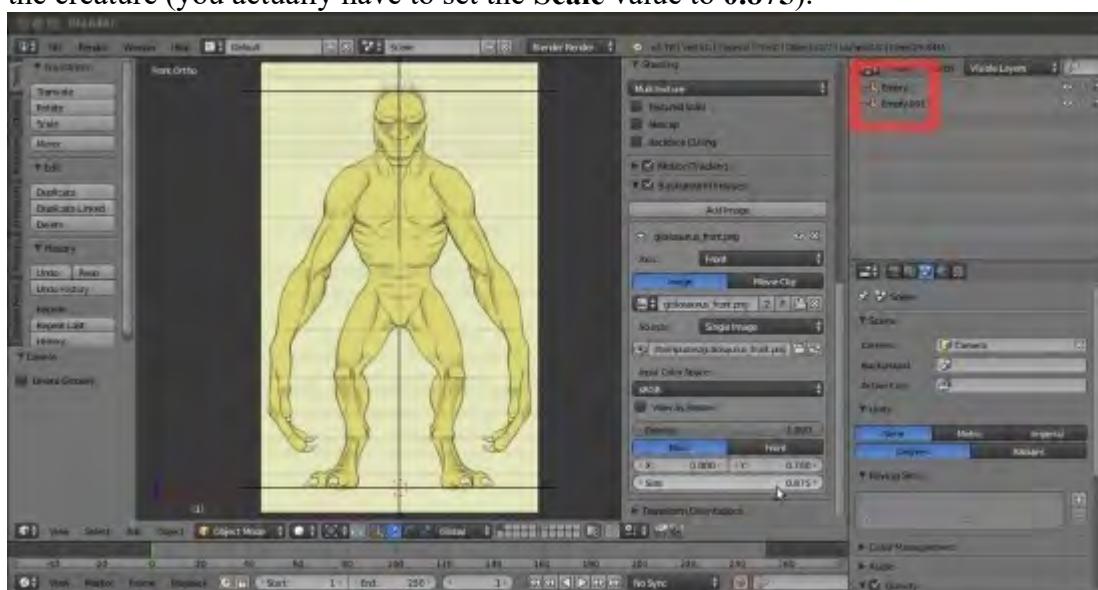
Getting ready

As in the former recipe, no need for any particular preparations; just carry out the preparatory steps as mentioned in the *Getting ready* section of the previous recipe.

How to do it...

So let's start by adding the templates as background images; that is, as reference images only visible in the background in **Ortho** view mode and, differently from the previous recipes, not as 3D objects actually present in the middle of the scene:

1. Press *I* on the numpad to switch to the orthographic **Front** view and press *Alt + Home* to center the view on the **3D Cursor**.
2. If not already present, press *N* to bring up the **Properties** sidepanel to the right-hand side of the 3D window; scroll down to reach the **Background Images** subpanel and enable it with the checkbox. Then click on the little arrow to expand it.
3. Click on the **Add Image** button; in the new option panel that appears, click on the **Open** button and browse to the `templates` folder to load the `gidiosaurus_front.png` image.
4. Click on the little window to the side of the **Axis** item and switch from **All Views** to **Front**, and then set the **Opacity** slider to **1.000**.
5. Increase the **Y offset** value to make the bottom/feet guideline of the reference image aligned to the horizontal arm of the first **Empty** (you have to set it to **0.780**).
6. Scale **Size** smaller, using both the **Empties** that we set as references for the **2.5** meters height of the creature (you actually have to set the **Scale** value to **0.875**).



The background image scaled and positioned through the settings in the N sidepanel

7. Click on the little white arrow on the top-left side of the `gidiosaurus_front.png` subwindow to collapse it.
8. Click on the **Add Image** button again; then, in the new option panel, click on the **Open** button, browse to the `templates` folder, and load the `gidiosaurus_side.png` image. Then, set the **Axis** item to **Right**, **Opacity** to **1.000**, **Scale** to **0.875**, and **Y** to **0.780**.
9. Repeat the operation for the `gidiosaurus_back.png` image, set **Axis** to **Back**, and so on.

Press 3 on the numpad to switch to the **Side** view, 1 to switch to the **Front** view, and **Ctrl + 1** to switch to the **Back** view, but remember that you must be in the **Ortho** mode (5 key on the numpad) to see the background templates:



The N sidepanel settings to assign the background image to a view

Building the character's base mesh with the Skin modifier

In the previous recipes, we saw three different ways to set up the template images; just remember that one method doesn't exclude the others, so in my opinion, the best setup you can have is: **Image Empties** on one layer (visibility toggled using the eye icons in the **Outliner**) together with **Background Images**. This way you can not only have templates visible in the three orthographic views, but also in the perspective view (and this can sometimes be really handy).

However, whatever the method you choose, now it's time to start to build the character's base mesh. To do this, we are going to use the **Skin** modifier.

Getting ready

First, let's prepare the scene:

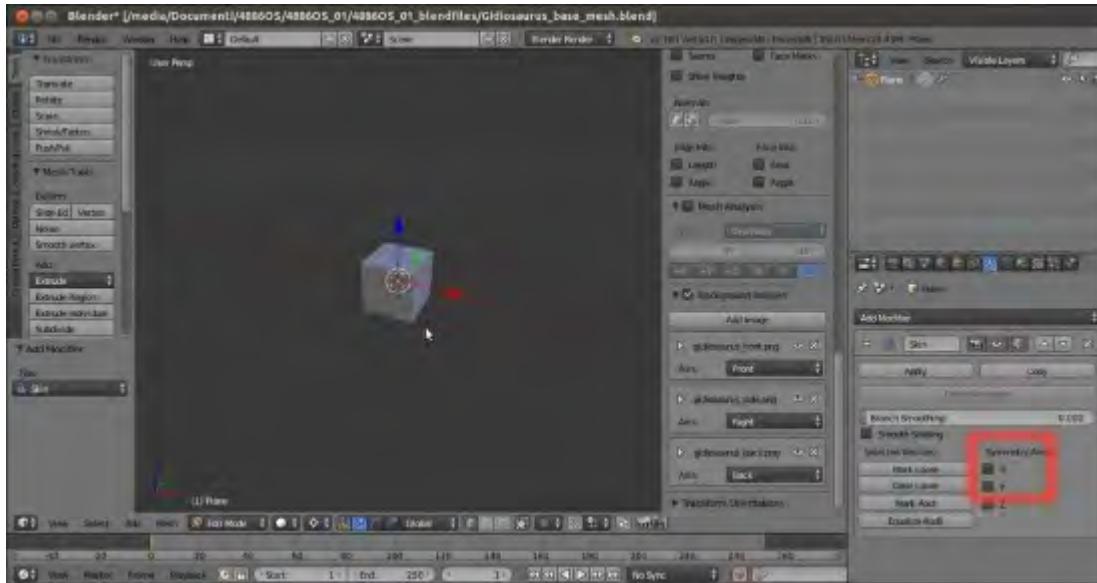
1. In case it's needed, reopen the `Gidiosaurus_base_mesh.blend` file.
2. Click on an empty scene layer to activate it; for example, the **11th**.



The starting empty scene and the scene layer's buttons on the 3D window toolbar

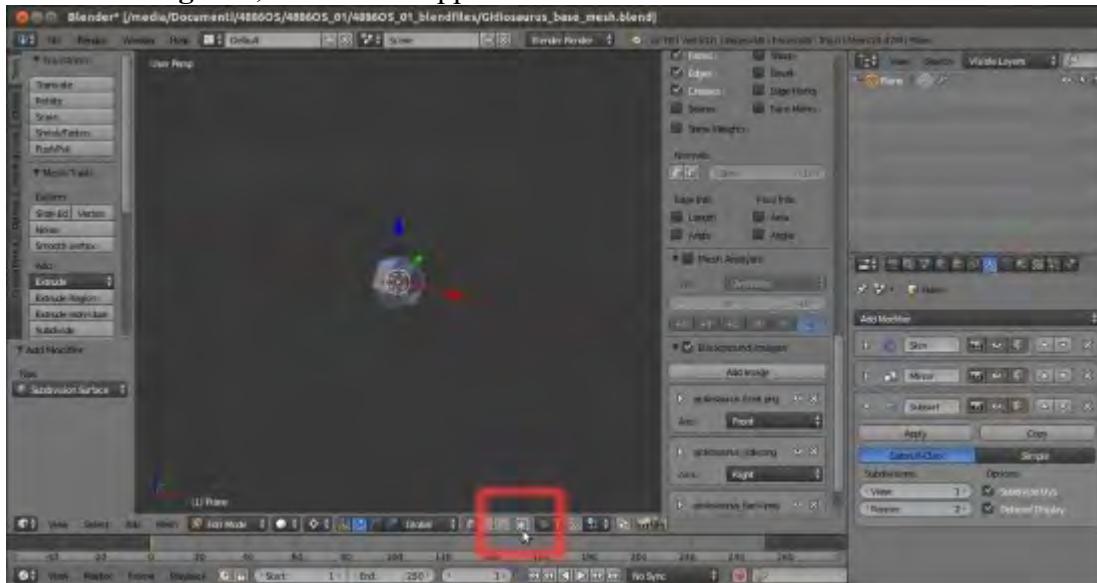
3. Be sure that the **3D Cursor** is at the center of the scene (*Shift + C*).
4. Add a **Plane** (press *Shift + A* and go to **Mesh | Plane**). If you are working with the **Factory Settings**, you must now press *Tab* to go in to **Edit Mode**, and then *Shift + right-click* to deselect just one vertex.
5. Press *X* and delete the three vertices that are still selected.
6. Right-click to select the remaining vertex and put it at the cursor location in the center of the scene (*Shift + S*, and then select **Selection to Cursor**).

7. Go to the **Object Modifiers** window on the main **Properties** panel, to the right-hand side, and assign a **Skin** modifier; a cube appears around the vertex. Uncheck **X** under **Symmetry Axes** in the modifier's panel:



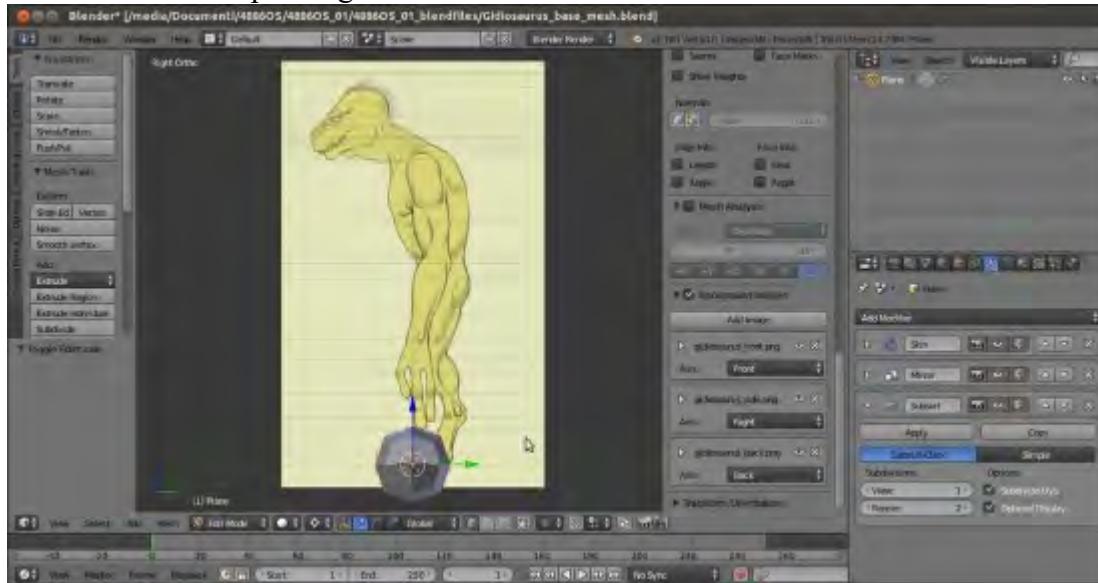
The cube geometry created by just one vertex and the Skin modifier

8. Assign a **Mirror** modifier and check **Clipping**.
9. Assign a **Subdivision Surface** modifier and check **Optimal Display**.
10. Go to the toolbar of the 3D view to click on the *Limit selection to visible* icon and disable it; the icon appears only in **Edit Mode** and in all the viewport shading modes, except for **Wireframe** and **Bounding Box**, and has the appearance of a cube with the vertices selected:



The “Limit selection to visible” button on the 3D viewport toolbar and the cube geometry subdivided through the Subdivision Surface modifier

11. Press 3 on the numpad to go in the **Side** view:



The created geometry and the side-view template reference

How to do it...

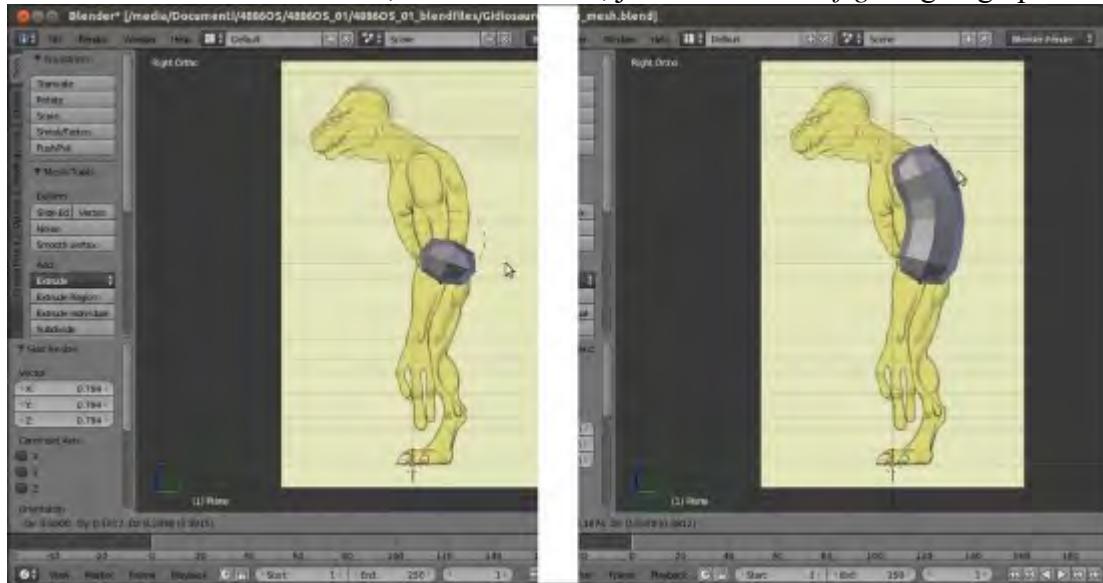
We are now going to move and extrude the vertex according to our template images, working as guides, and therefore generating a 3D geometry (thanks to the **Skin** modifier):

1. Press *G* and move the vertex to the pelvis area. Then, press *Ctrl + A* and move the mouse cursor towards the vertex to lower the weight/influence of the vertex itself on the generated mesh; scaling it smaller to fit the hip size showing on the template:



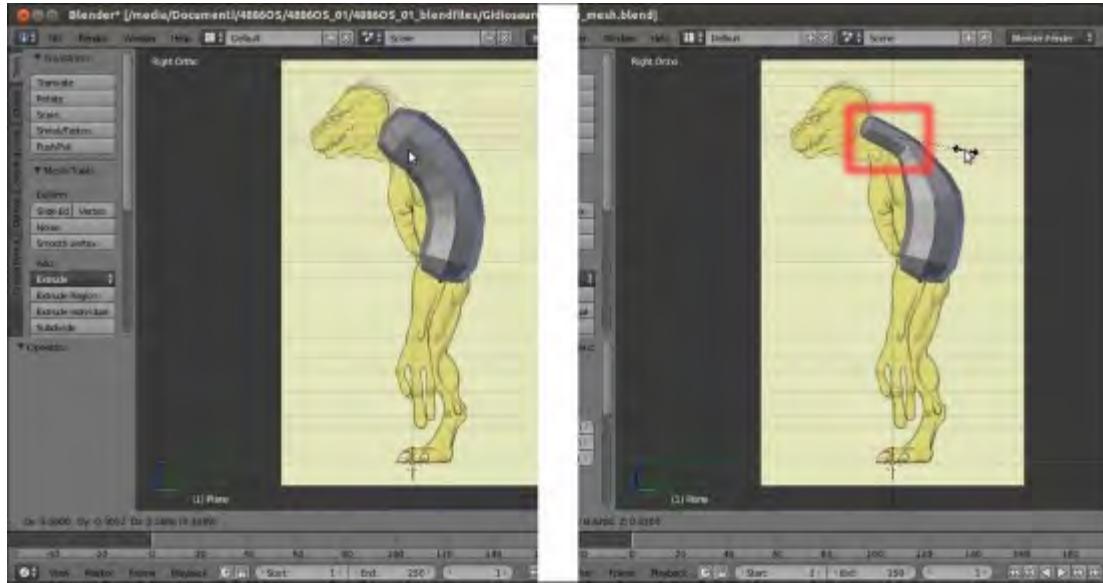
Moving the geometry to the character's pelvis area

2. Press *E* and extrude the vertex by moving it up on the *z* axis; place it at the bottom of the rib cage.
3. Go on extruding the vertex by following the lateral shape of the character in the template. Don't be worried about the volumes; for the moment, just build a *stick-figure* going up the torso:



Extruding the vertices to create a new geometry

4. Proceed to the neck and stop at the attachment of the head location.
5. Select the last two vertices you extruded; press *Ctrl + A* and move the mouse cursor towards them to scale down their influence in order to provide a slim-looking neck:



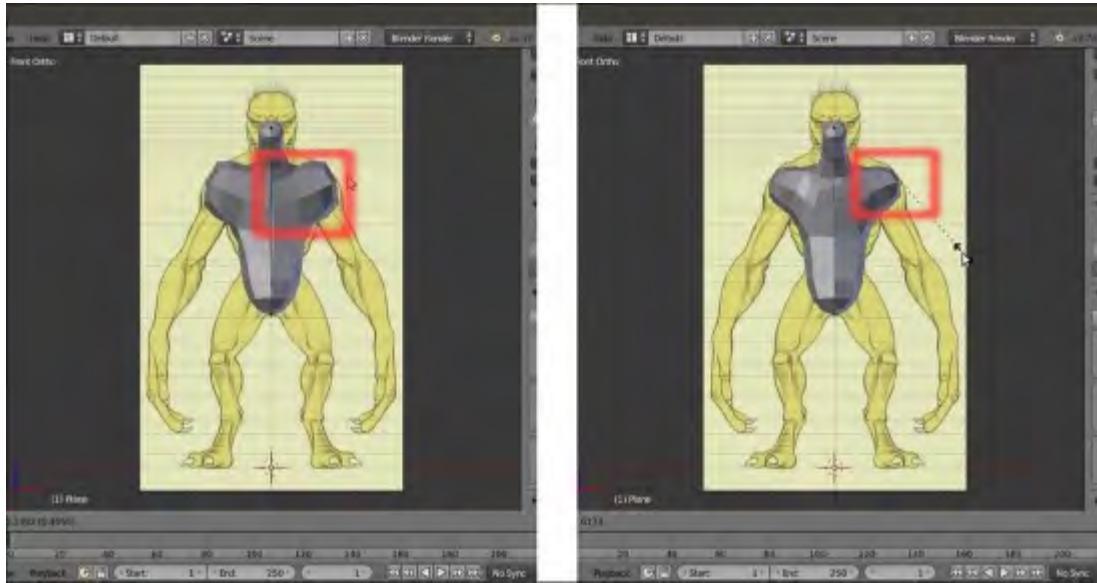
Scaling down the influence of the vertices

6. Press **I** on the numpad to switch to the **Front** view, and then select the bottom vertex and extrude it down to cover the base of the creature's pelvis. Press **Ctrl + A | X** to scale it only on the **x** axis:



Adjusting the weight of the vertices in the Front view

7. Go to the **Mirror** modifier and uncheck the **Clipping** item.
8. Select the middle thorax vertex and extrude it to the right-hand side to build the shoulder. Press **Ctrl + A** to scale it smaller:



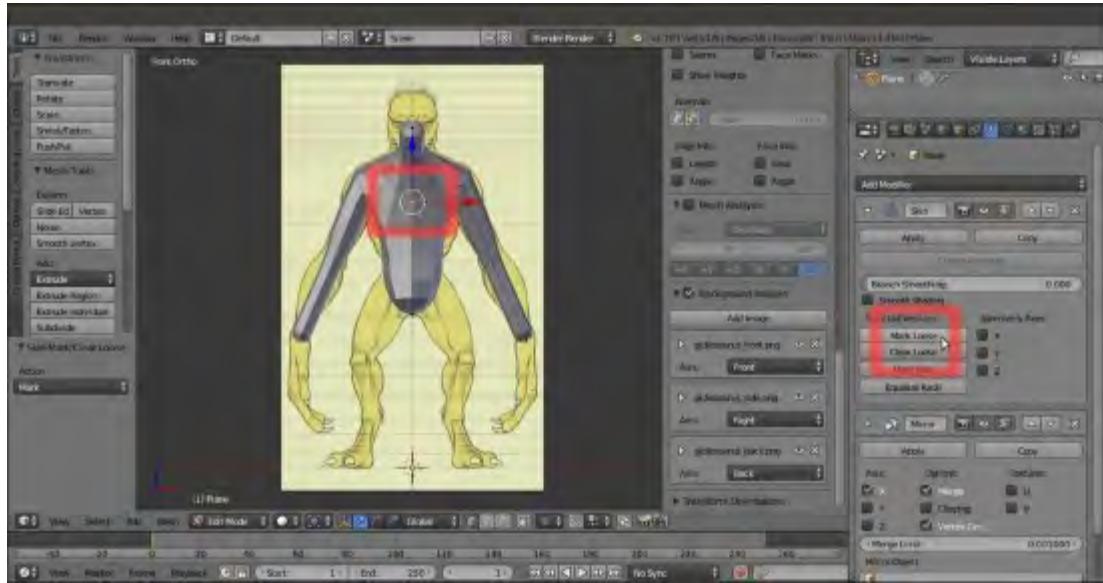
Creating the shoulders

9. Extrude the shoulder vertex, following the arm shape, and stop at the wrist; select the just-extruded arms' vertices and use *Ctrl + A* to scale them smaller.
10. Reselect the shoulder vertex, and use *Shift + V* to slide it along the shoulder's edge in order to adjust the location and fix the area shape:



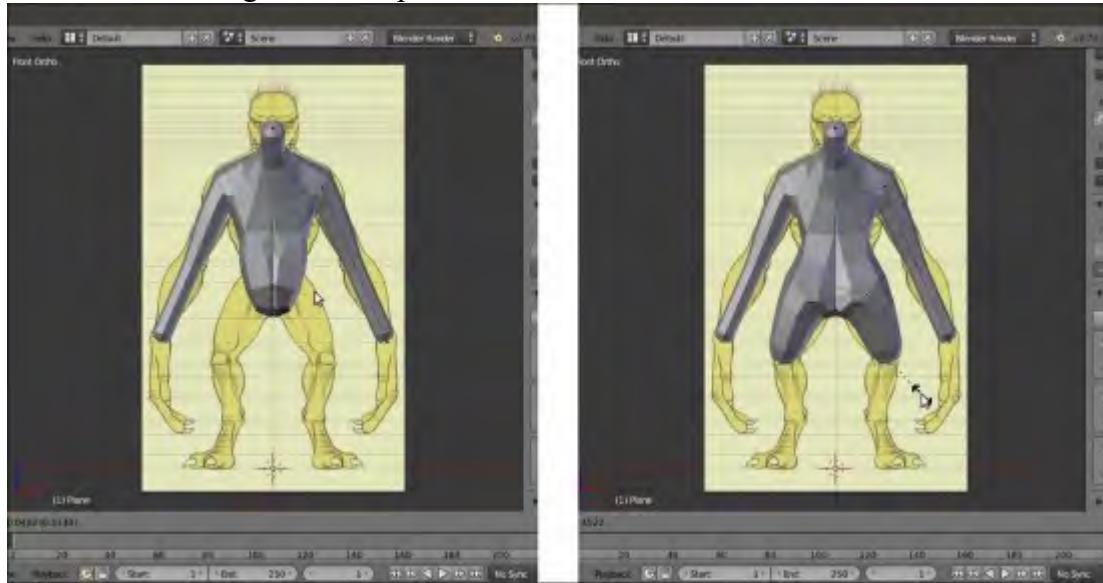
Creating the arms

11. Select the middle thorax vertex we extruded the shoulder from and go to the **Skin** modifier; click on the **Mark Loose** button:



Making a more natural transition from the thorax to the arms

12. Select the second vertex from the bottom and extrude it to the right-hand side to build the hip, and then extrude again and stop at the knee. Use **Ctrl + A** on the vertex to make it smaller:



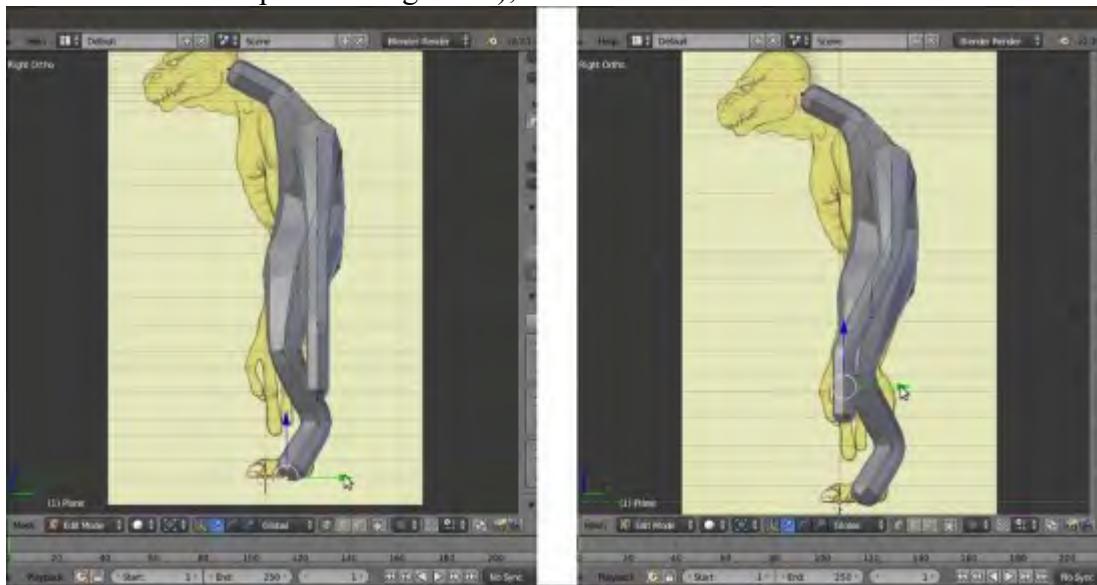
Extruding the thighs

13. Go on extruding the vertex to build the leg. Then, select the wrist vertex and extrude it to build the hand:



Extruding to complete the leg

14. Press 3 to go to the **Side** view.
15. Individually, select the vertices of the knee, ankle, and foot, and move them to be aligned with the character's posture (you can use the widget for this and, if needed, you can press Z to go in to **Wireframe** viewport shading mode); do the same with the vertices of the arm:



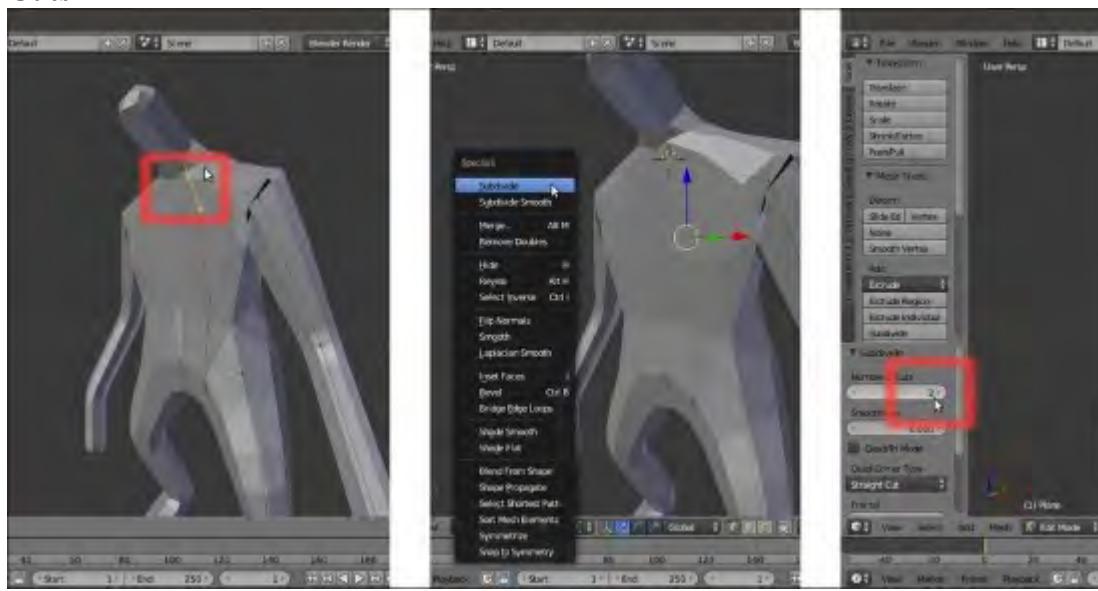
Adjusting the arm's position

16. Select the vertices of the shoulder and elbow, and move them forward according to the template position; do the same with the vertices of the neck and waist:



Adjusting the position of the shoulders, thorax, and neck

17. Select the vertex connecting the shoulder to the thorax and use *Shift + V* to slide it upwards, in order to make room for more vertices in the chest area. Use *Shift* to select the vertex at the bottom of the rib cage and press *W*; in the **Specials** pop-up menu, select **Subdivide** and, right after the subdivision, in the option panel at the bottom-left of the Blender UI, set **Number of Cuts** to 2:



Subdividing an edge

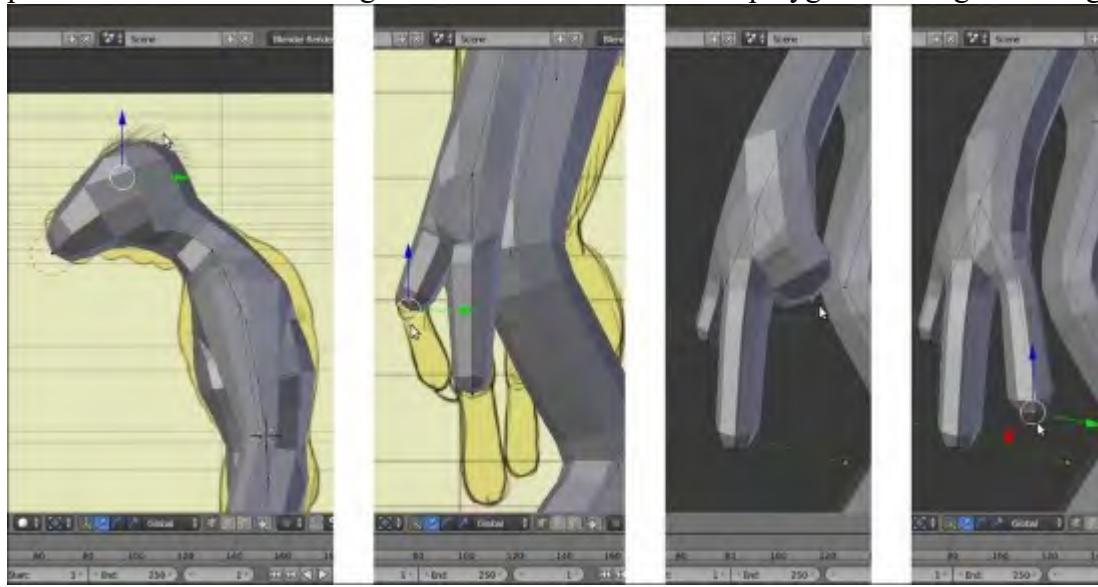
18. In the **Side** view, select the upper one of the new vertices and use *Ctrl + A* to scale it bigger. Adjust the position and scale of the vertices around that area (neck and shoulder) to obtain, as

much as possible, a shape that is more regular and similar to the template. However, don't worry too much about a perfect correspondence, it can be adjusted later:



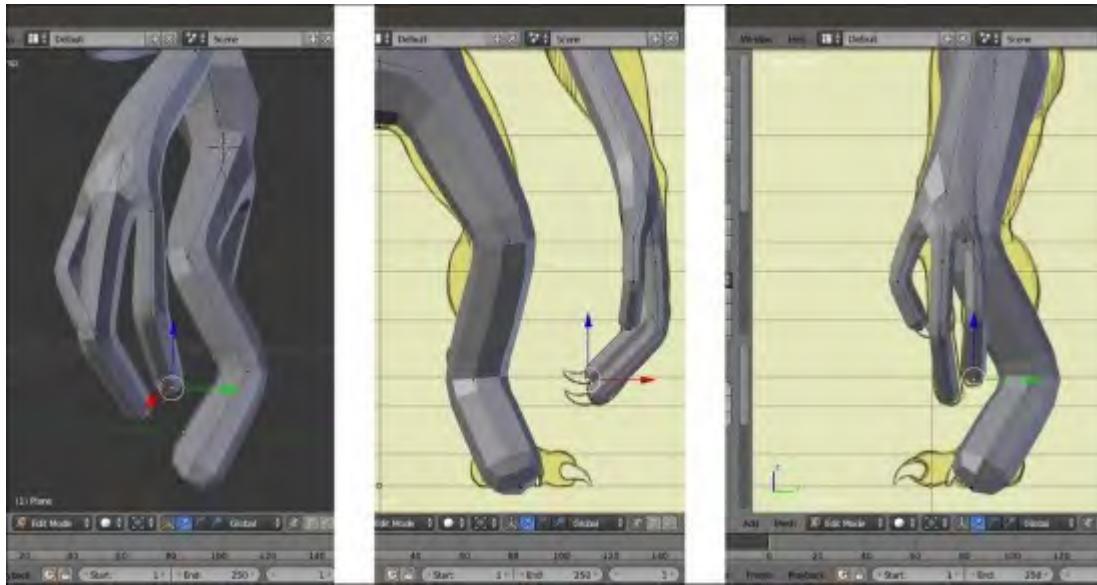
Refining the shoulder's shape

19. Extrude the bulk of the head. Select the last hand vertex and scale it smaller. Then, select the upper hand vertex and extrude two more fingers (scale their influence smaller and adjust their position to obtain a more regular and ordinate flow of the polygons in the generated geometry):



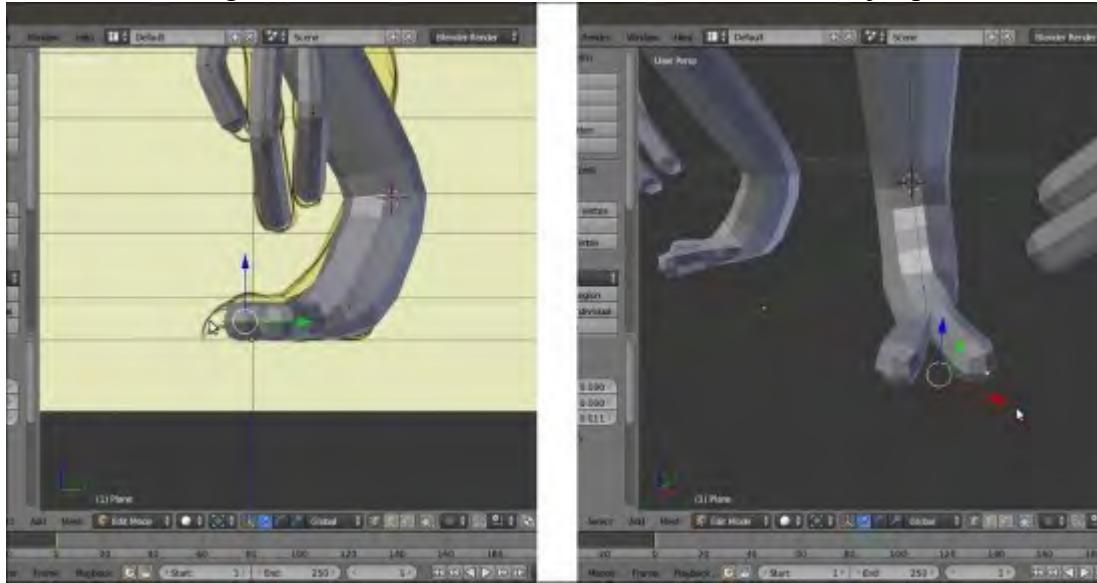
Creating the head, hands, and fingers

20. As always, following the templates as reference, extrude again to complete the fingers; use all the templates to check the accuracy of the proportions and positions, and the **Front**, **Side**, and **Back** views too:



Adjusting the position of the fingers according to the templates

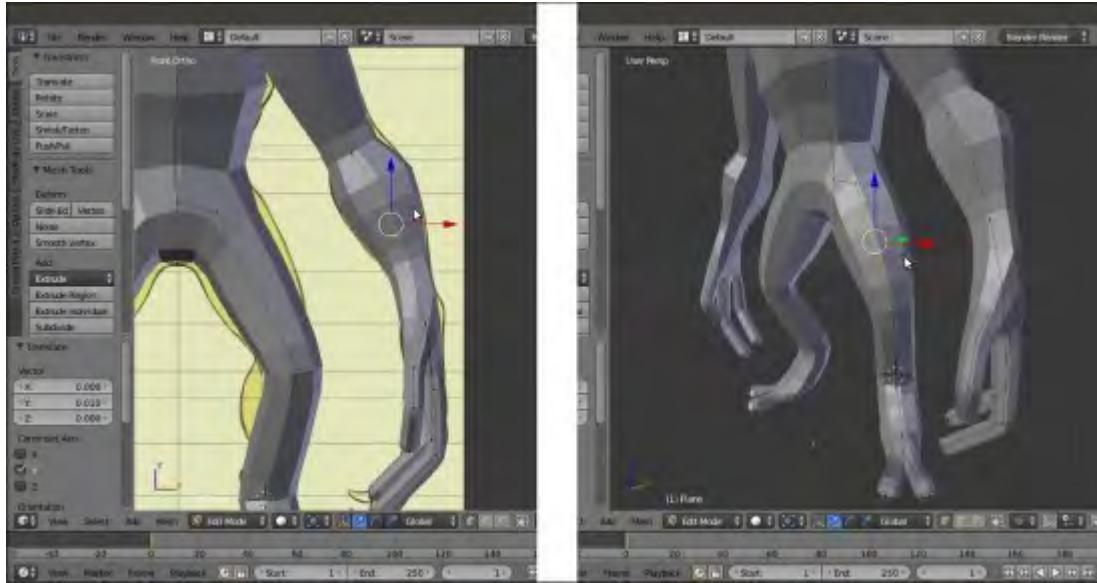
21. Do the same thing for the foot, and we are almost done with the major part of the mesh:



Creating the feet toes

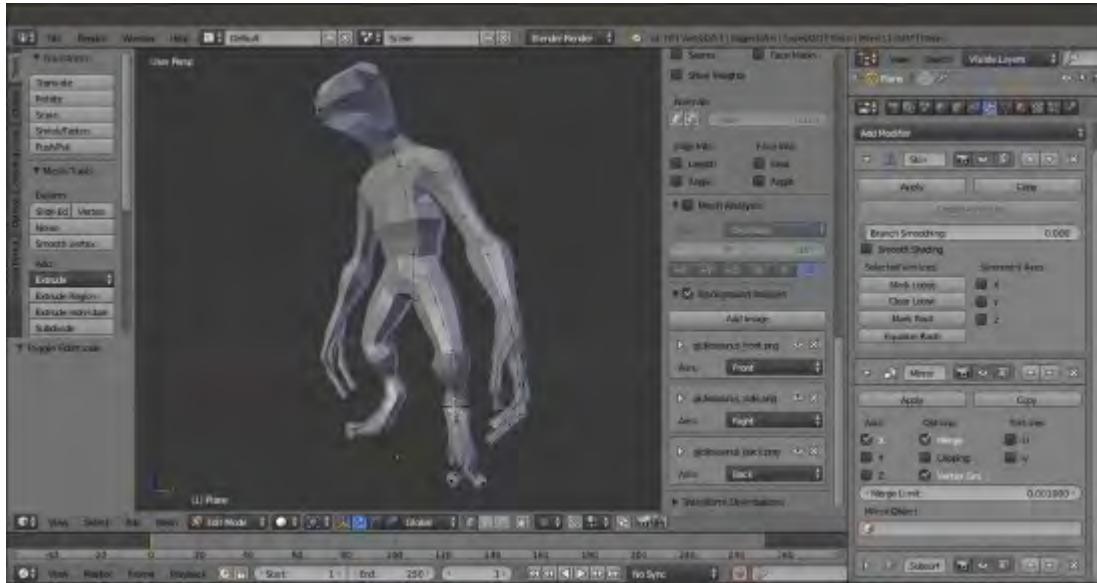
Now, it's only a matter of refining, as much as possible, the mesh's parts to resemble best the final shape of the character. Let's try with the arm first:

22. Select the two extreme vertices of the forearm and press **W | Subdivide | 2** (in the bottom **Tool** panel) to add 2 vertices in the middle. Then, use **Ctrl + A** to scale and move them outward to curve the forearm a little bit. Do the same for the thigh by slightly moving the vertices outward and backward:



Refining the shape of arms and legs

23. Repeat the same procedure with the upper arm, shin, foot, and fingers; any part where it's possible, but don't go crazy about it. The goal of such a technique is just to quickly obtain a mesh that is good enough to be used as a starting point for the sculpting, and not an already finished model:



The completed base mesh

24. Press *Tab* to go out of the **Edit Mode**; go to the **Outliner** and rename the base mesh as *Gidiosaurus*. Then, save the file.

How it works...

The **Skin** modifier is a quick and simple way to build almost any shape; its use is very simple: first, you extrude vertices (actually, it would be enough to add vertices; it's not mandatory to extrude them, but certainly it's more handy than using *Ctrl + left-click* to add them at several locations), and then using the *Ctrl + A* shortcut, you scale smaller or bigger the influence that these vertices have on the 3D geometry generated on the fly.

If you have already tried it, you must have seen that the more the complexity of the mesh grows, the more the generated geometry starts to become a little unstable, often resulting in intersecting and overlapping faces. Sometimes this seems unavoidable, but in any case it is not a big issue and can be easily fixed through a little bit of editing. We'll see this in the next chapter.

Chapter 2. Sculpting the Character's Base Mesh

In this chapter, we will cover the following recipes:

- Using the Skin modifier's Armature option
- Editing the mesh
- Preparing the base mesh for sculpting
- Using the Multiresolution modifier and the Dynamic topology feature
- Sculpting the character's base mesh

Introduction

In the previous chapter, we built the base mesh by using the **Skin** modifier and on the base of the reference templates; in this chapter, we are going to prepare this basic mesh for the sculpting, by editing it and cleaning up any *mistakes* the **Skin** modifier may have made (usually, overlapping and triangular faces, missing edge loops, and so on).

Using the Skin modifier's Armature option

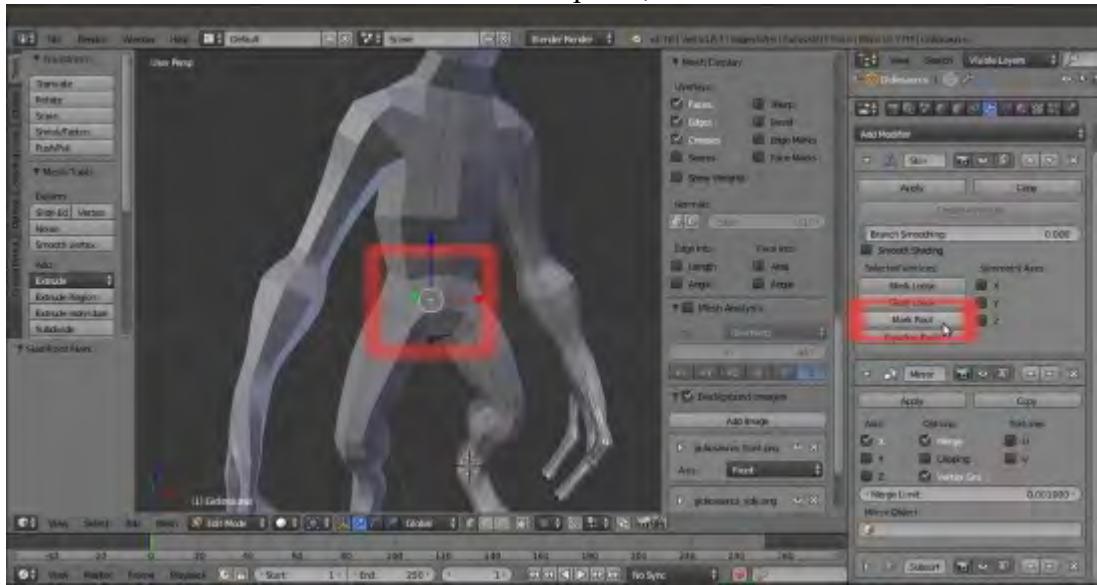
The **Skin** modifier has an option to create an **Armature** on the fly to pose the **generated mesh**. This **Armature** can just be useful in cases where you want to modify the position of a part of the generated mesh.

Note that using the generated **Armature** to pose the base mesh, in our case, is not necessary, and therefore this recipe is treated here only as *an example* and it won't affect the following recipes in the chapter.

Getting ready

So, let's suppose that we want the **arms** to be posed more horizontally and widely spread:

1. If this is the case, reopen the `Gidiosaurus_base_mesh.blend` file and save it with a different name (something like `Gidiosaurus_Skin_Armature.blend`).
2. Select the **Gidiosaurus** mesh and press `Tab` to go into **Edit Mode**; then, select the central pelvis vertex.
3. Go to the **Object Modifiers** window under the main **Properties** panel to the right-hand side of the screen and then to the **Skin** modifier subpanel; click on the **Mark Root** button:



The root vertex

4. Press `Tab` again to exit **Edit Mode**.

How to do it...

Creating the rig (that is the skeleton **Armature** made by bones and used to deform, and therefore, animate a mesh) for our character's base mesh is really simple:

- Again, in the **Skin** modifier subpanel, click on the **Create Armature** button. The **Armature** is created instantly and an **Armature** modifier is automatically assigned to the mesh; in the modifier stack, move it to the top so that it is above the **Mirror** modifier and our posed half-mesh will be correctly mirrored:



The Armature created by the Skin modifier

- Press **Ctrl + Tab** to enter **Pose Mode** for the already selected **Armature** and then select the upper bone of the **arm**.
- In the toolbar of the 3D viewport, find the widget manipulators panel, click on the rotation **Transformation manipulators** (the third icon from the left), and set **Transform Orientation** to **Normal**.
- By using the rotate widget, rotate the selected bone and consequently the arm (be careful that, as already mentioned, the newly created **Armature** modifier is at the top of the modifier stack, otherwise the rotation will not correctly deform the mirrored mesh):



Rotating the arms through the Armature

5. Exit **Pose Mode** and reselect the **Gidiosaurus** mesh.
6. Go to the **Skin** modifier subpanel under the **Object Modifiers** window; click on the **Apply** button to apply the modifier.
7. Go to the **Armature** modifier and click on the **Apply** button to also apply the rig transformations.
8. At this point, we can also select the **Armature** object and delete it (X key).

How it works...

By clicking on the **Create Armature** button, the **Skin** modifier creates a bone for each edge connecting the extruded vertices, it adds an **Armature** modifier to the generated base mesh, and automatically assigns vertex groups to the base mesh and skins them with the corresponding bones.

The bones of this **Armature** work in **Forward Kinematics**, which means they are chained following the **child/parent** relation, with the first (**parent**) bone created at the **Root** location we had set at step 3 of the *Getting ready* section.

There's more...

Note that the bones of the **Armature** can be used not only to rotate limbs, but also to scale bigger or smaller parts of the mesh, in order to further tweak the shape of the base mesh.

See also

- <http://www.blender.org/manual/modifiers/generate/skin.html>
- <http://www.blender.org/manual/rigging/posing/editing.html#effects-of-bones-relationships>

Editing the mesh

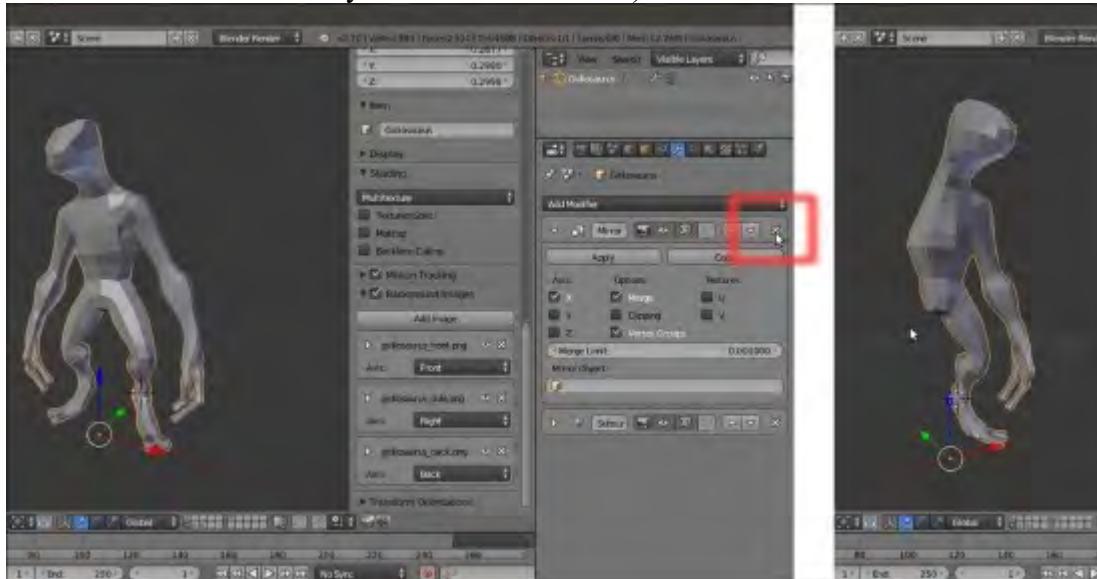
Once we have applied the **Skin** and **Armature** modifiers, we are left with an almost ready-to-use base mesh; what we need to do now is clean the possibly overlapping faces and whatever other mistakes were made by the **Skin** modifier.

Be careful not to be confused by the previous recipe, which was meant only as a possible example; we didn't actually use the **Skin** modifier's **Armature** to change the pose of the base mesh.

Getting ready

Let's prepare the mesh and the view:

1. Go to the **Object Modifiers** window under the main **Properties** panel and then to the **Mirror** modifier subpanel and click on the little X icon to the right in order to delete the modifier; you are left with half of the mesh (actually the half that is really generated by the **Skin** modifier; the other side was *simulated* by the **Mirror** modifier):

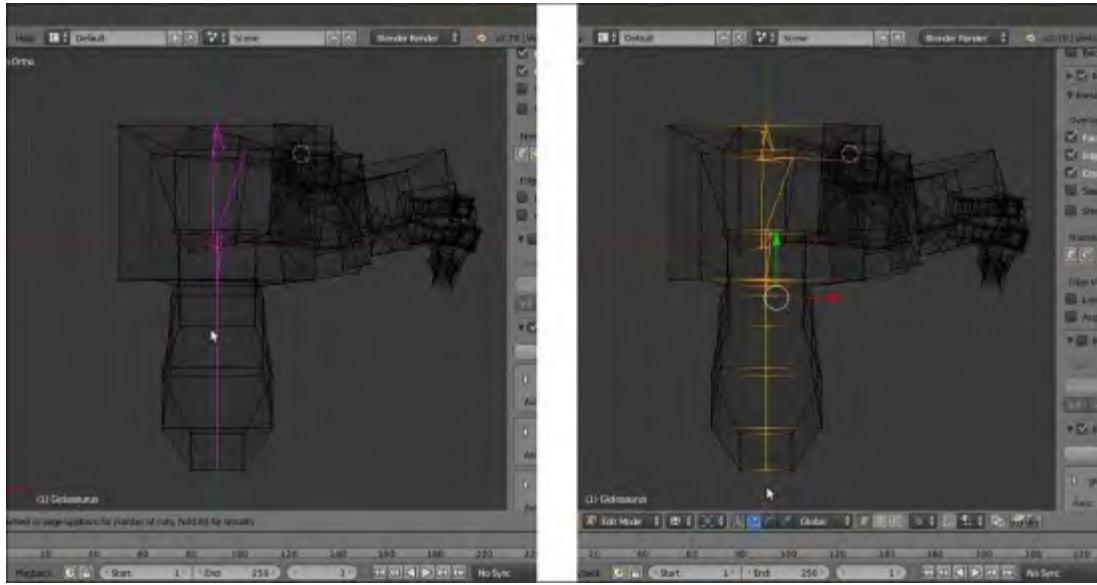


Deleting the Mirror modifier

2. Press **Tab** to go into **Edit Mode**, **7** on the numpad to go into **Top** view, and **Z** to go into the **Wireframe** viewport shading mode.

How to do it...

1. Press **Ctrl + R** to add an edge-loop to the middle of the mesh; don't move the mouse, and left-click a second time to confirm that you want it at **0.0000** location:

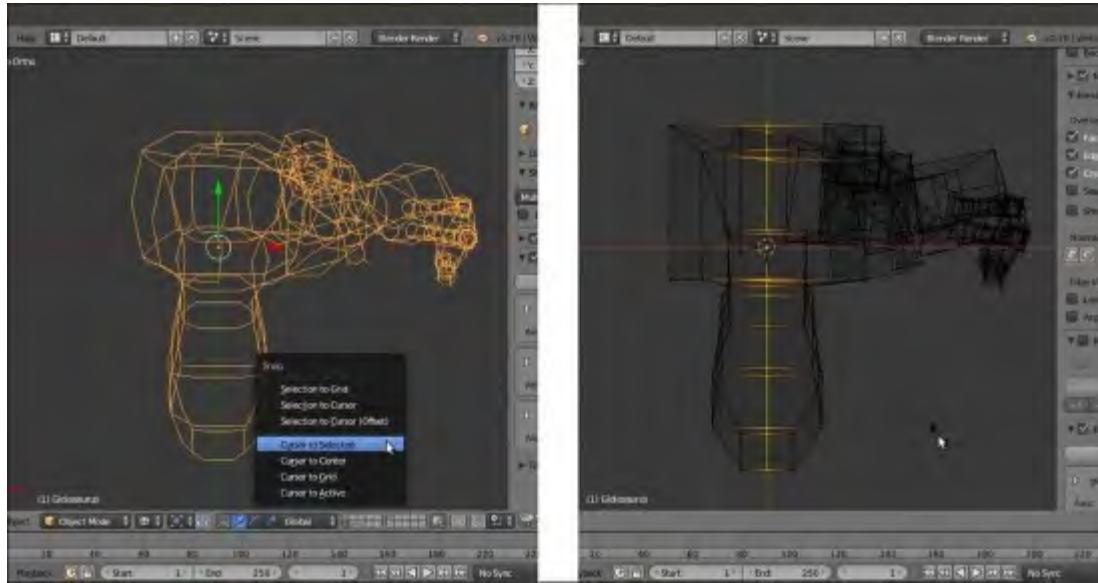


Adding a central edge-loop

Sometimes, depending on the topology created by the **Skin** modifier, you may not be able to make a single clean loop cut by the *Ctrl + R* key shortcut. In this case, still in **Edit Mode**, you can press the *K* key to call the **Knife Tool**, left-click on the mesh to place the cuts, and press *Enter* to confirm (press *Shift + K* if you want only the newly created edge-loops selected after pressing *Enter*). This way, you can create several loop cuts, connect them together and, if necessary, move and/or scale them to the middle along the *x* axis.

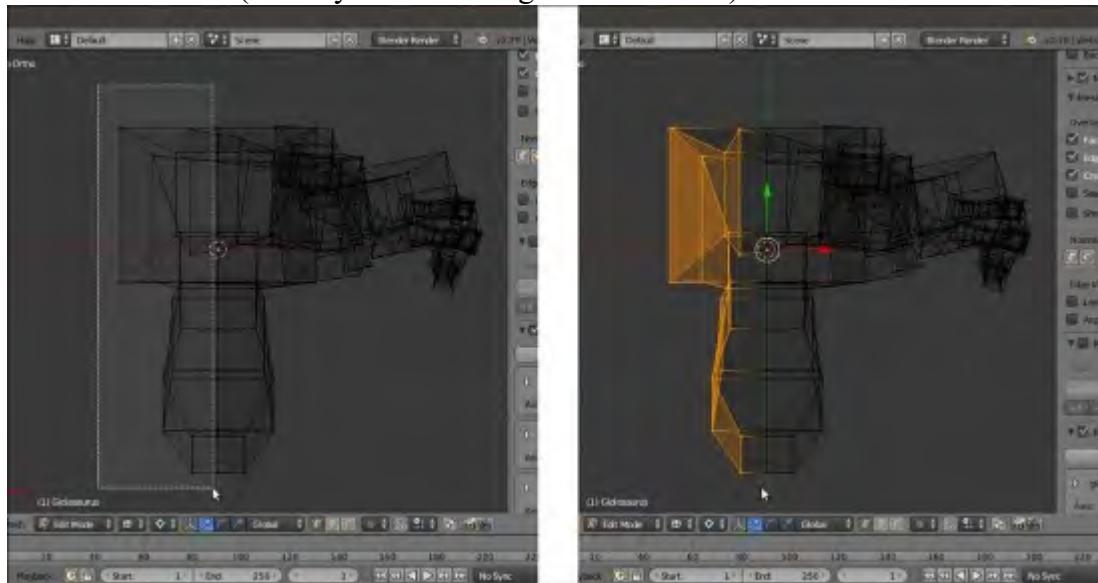
In fact, you can do the following:

2. Go out of **Edit Mode** and press *Shift + S*; in the **Snap** pop-up menu, select **Cursor to Selected** (to center the cursor at the middle of the mesh).
3. Press the period (.) key to switch **Pivot Point to 3D Cursor** and then press *Tab* to go again into **Edit Mode**.
4. With the middle edge-loop already selected, press *S | X | 0 | Enter* to scale all its vertices to the **3D Cursor** position along the *x* axis and align them at the perfect center:



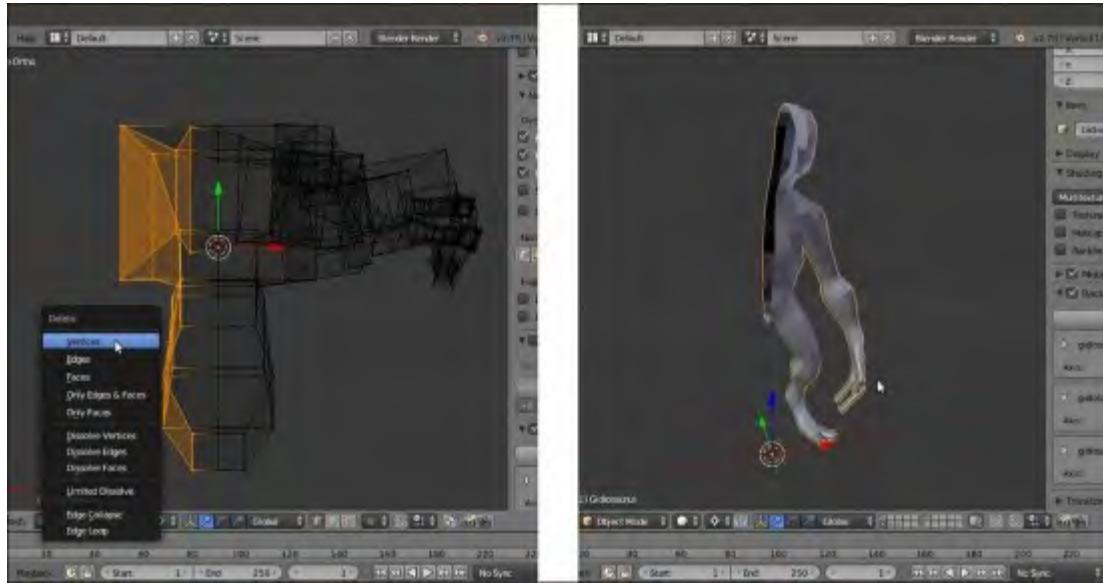
Scaling the central edge-loop vertices along the x axis

5. Press A to deselect all the vertices and then press B and box-select the vertices on the left-hand side of the screen (actually the mesh's right-side vertices):



Box-selecting the left vertices

6. Press X and, in the **Delete** pop-up menu, select the **Vertices** item to delete them:



Deleting the left vertices

7. Go out of **Edit Mode** and, in the **Object Modifiers** window, assign a new **Mirror** modifier (check **Clipping**) to the mesh; move it before the **Subdivision Surface** modifier in the stack.
8. If needed, this is the point where you can manually edit the mesh by converting triangle faces to quads (select two consecutive triangular faces and press *Alt + J*), creating, closing, or moving edge-loops (by using the **Knife Tool**, for example, around the arms and legs attachments to the body), and so on.
9. Save the file as `Gidiosaurus_base_mesh.blend`.

Well, in our case, everything went right with the **Skin** modifier, so there is no need for any big editing of the mesh! In effect, it was enough to delete the first **Mirror** modifier (that we actually used mostly for visual feedback) to get rid of all the overlapping faces and obtain a clean base mesh:



The "clean" mesh with new **Mirror** and **Subdivision Surface** modifiers

In the preceding screenshot, the base mesh geometry is showing with a level 1 of subdivision; in **Edit Mode**, it is still possible to see the low-level cage (that is, the *real* geometry of the mesh) as wireframe.

There are a couple of triangular faces (that, if possible, we should always try to avoid; quads faces work better for the sculpting) near the shoulders and on the feet, but we'll fix these automatically later, because before we start with the sculpting process, we will also apply the **Subdivision Surface** modifier.

How it works...

To obtain a clean half-body mesh, we had to delete the first **Mirror** modifier and the vertices of the right half of the mesh; to do this, we had also added a middle edge loop. So, we obtained a perfect left-half mesh and therefore we assigned again a **Mirror** modifier to restore the missing half of the body.

Preparing the base mesh for sculpting

Once we have our base mesh completed, it's time to prepare it for the sculpting.

Getting ready

Open the `Gidiosaurus_base_mesh.blend` file and be sure to be out of **Edit Mode**, and therefore in **Object Mode**.

How to do it...

1. Select the character's mesh and go to the **Object Modifiers** window under the main **Properties** panel to the right.
2. Go to the **Mirror** modifier panel and click on the **Apply** button.
3. If this is the case, expand the **Subdivision Surface** modifier panel, be sure that the **View** level is at **1**, and click on the **Apply** button.
4. Press *Tab* to go into **Edit Mode** and, if necessary, select all the vertices by pressing *A*; then, press *Ctrl + N* to recalculate the normals and exit **Edit Mode**.
5. Go to the **Properties** sidepanel on the right-hand side of the 3D view (or press the *N* key to make it appear) and under the **View** subpanel, change the **Lens** angle to **60.000** (more natural looking than **35.000**, which is set by default).
6. Under the **Display** subpanel, check the **Only Render** item:



Setting the view through the 3D window *N* sidepanel

7. Go to the **Shading** subpanel on the sidepanel on the right-hand side of the 3D viewport and check the **Matcap** item.
8. Left-click on the preview window that just appeared and, from the pop-up panel, select the red colored brick material, the one that looks like ZBrush material; obviously, you can choose a

different one if you prefer, but in my experience, this is the one that gives the best visual feedback in the 3D view:



The available matcaps menu and the selected Zbrush-like matcap

9. Put the mouse cursor inside the active 3D window and press **Ctrl + Spacebar** to disable the widget:



The matcap assigned to the mesh and the widget button in the 3D window toolbar

10. Press **N** to get rid of the **Properties** 3D window sidepanel.
11. Save the file as **Gidiosaurus_Sculpt_base.blend**.

How it works...

By checking the **Only Render** item in the **Display** subpanel under the **Properties** 3D window sidepanel, all the possible disturbing elements that cannot be rendered (such as the **Grid Floor**, **Empties**, **Lamps**, and so on) are hidden, in order to give a clean 3D viewport ready for sculpting.

Note that with this option enabled, sadly, the **Image Empties** we set in the previous chapter to work as templates for references are not visible—instead, the templates we had set as **Background Images** are perfectly visible in the 3 orthographic views.

Matcaps can in some cases slow the performance of your computer, depending on the hardware; in any case, **Matcaps** is a very useful feature, especially for sculpting, as you can see the mesh shape easily.

Changing the **Lens** angle from **35.000** to **60.000** makes the perspective view look more similar to the natural human field of view.

Using the Multiresolution modifier and the Dynamic topology feature

To be sculpted, a mesh needs a big enough amount of vertices to allow the adding of details; in short, we now need a way to add (a lot of!) geometry to our simple base mesh.

Besides the usual subdividing operation in **Edit Mode** (press **Tab**, then **A** to select all the vertices, then press **W** to call the **Specials** menu, click on **Subdivide**, and then set the **Number of Cuts** value in the last operation subpanel at the bottom of the **Tool Shelf**) and the **Subdivision Surface** modifier, in Blender, there are two other ways to increase the amount of vertices: one is by assigning a **Multiresolution** modifier to the mesh (a nondestructive way) and the other is by using the **Dynamic topology** feature. We are going to see both of them.

Getting ready

As usual, let's start from the last .blend file we saved: in this case, **Gidiosaurus_Sculpt_base.blend**.

How to do it...

Let's start with the **Multiresolution** modifier method:

1. First of all, save the file as **Gidiosaurus_Multires.blend**.
2. Select the base mesh and go to the **Object Modifiers** window under the main **Properties** panel on the right-hand side of the screen; assign a **Multiresolution** modifier.
3. Click on the **Subdivide (Add a new level of subdivision)** button 3 times; the mesh has now reached **143,234** vertices and **143,232** faces.
4. Check the **Optimal Display** item in the modifier panel:



The mesh with a Multiresolution modifier assigned at level 3 of subdivision

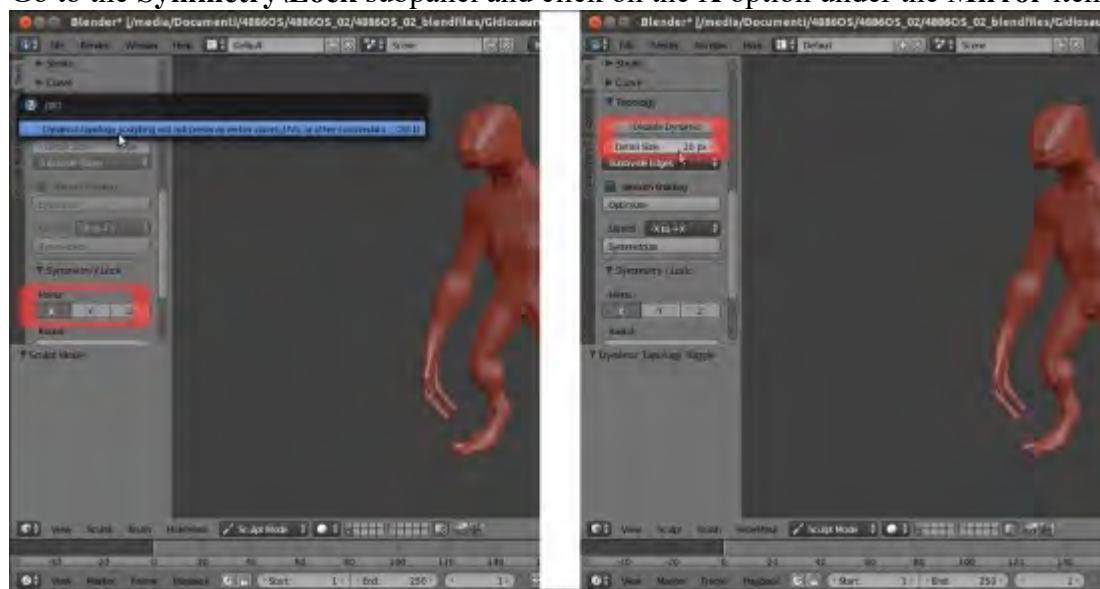
5. On the toolbar of the 3D window, click on the mode button to go into **Sculpt Mode**.
6. On the **Tools** tab on the left-hand side of the screen (if necessary, press the **T** key to make the **Tool Shelf** containing the tabs appear), go to the **Symmetry\Lock** subpanel and click on the **X** button under the **Mirror** item.
7. Click on the **Options** tab and, under the **Options** subpanel, uncheck the **Size** item under **Unified Settings**.
8. Start to sculpt.

At this point, to proceed with the sculpting, you should jump to the next recipe, *Sculpting the character's base mesh*; instead, let's suppose that we have already sculpted our base mesh, so let's move ahead:

9. Exit **Sculpt Mode**.
10. Save the file.

Now, let's see the quick and easy preparation necessary to use the **Dynamic topology** feature for sculpting:

1. Reload the **Gidiosaurus_Sculpt_base.blend** blend file.
2. Then, save it as **Gidiosaurus_Dynatopo.blend**.
3. On the toolbar of the 3D window, select **Sculpt Mode**.
4. On the **Tools** tab on the left-hand side of the screen (press the **T** key to make the **Tool Shelf** containing the tabs appear), go to the **Topology** subpanel and click on the **Enable Dynatopo** button; a popup appears to inform you that the **Dynamic topology** feature doesn't preserve any already existing **Vertex Color**, **UV layer**, or other custom data (only if the mesh has them). Then click on the popup to confirm and go on.
5. Change the **Detail Size** value to **15/20 pixels**.
6. Go to the **Symmetry\Lock** subpanel and click on the **X** option under the **Mirror** item:



7. Start to sculpt.

Again, here you can jump to the next recipe, *Sculpting the character's base mesh*; in any case, remember to save the file.

How it works...

The **Multiresolution** modifier increasingly subdivides the mesh at each level by adding vertices; we have seen that from **2,240** starting vertices of the base mesh, we have reached **143,234** vertices at level **3**, and clearly this allows for the sculpting of details and different shapes. The vertices added by the modifier are *virtual*, exactly as the vertices added by the **Subdivision Surface** modifier are; the difference is that the vertices added by the latter are not editable (unless you apply the modifier, but this would be counterproductive), while it's possible to edit (normally through the sculpting) the vertices at each level of subdivision of a **Multiresolution** modifier. Moreover, it's always possible to go back by lowering the levels of subdivision, and the sculpted details will be stored and shown only in the higher levels; this means that the **Multiresolution** method is a nondestructive one and we can, for example, rig the mesh at level **0** and render it at the highest/sculpted level.

The **Dynamic topology** setting is different from the **Multiresolution** modifier because it allows you to sculpt the mesh without the need to heavily subdivide it first, that is, the mesh gets subdivided on the fly *only where needed*, according to the workflow of the brushes and settings, resulting in a much lower vertex count for the final mesh in the end.

As you can see in the screenshots (and in the .blend files provided with this cookbook), starting to sculpt the character with the **Multiresolution** modifier or the **Dynamic topology** is quite different. In the end, the process of sculpting is basically the same, but in the first case, you have an already smoothed-looking mesh where you must add or carve features; in the second case, the low resolution base mesh doesn't change its raw look at all until a part gets sculpted and therefore subdivided and modified, that is, all the corners and edges must first be softened, in order to round an otherwise harsh shape.

Sculpting the character's base mesh

Whatever the method you are going to use, it's now time to start with the effective sculpting process.

However, first, a disclaimer: in this recipe, I'm not going to teach you *how to sculpt*, nor is this an anatomy lesson of any kind. For these things, a book itself wouldn't be enough. I'm just going to demonstrate the use of the Blender sculpting tools, showing what brush I used for the different tasks, the sculpting workflow following the reference templates, and some of the more frequently used shortcut keys.

Getting ready

In this recipe, we'll use the **Dynamic topology** method. If you haven't followed the instructions of the previous recipe, just follow the steps from 12 to 17; otherwise, just open the `Gidiosaurus_Dynatopo.blend` file that is provided.

How to do it...

As usual, it's a good habit to save the file with the proper name as the first thing; in this case, save it as `Gidiosaurus_Dynatopo_Sculpt.blend`.

If you are going to use a graphic tablet to sculpt, remember to enable the tablet pressure sensitivity for both size and strength; in any case, it is better to set the respective sliders to values lower than **100 percent**; I usually set the size slider around **30/35** and the strength slider to **0.500**, but this is subjective:



The tablet pressure sensitivity buttons for the size and the strength

1. If you haven't already, go into **Sculpt Mode** and enable the **Dynamic topology** feature by clicking on the **Enable Dyntopo** button in the subpanel with the same name under the **Tool Shelf** panel or by directly pressing *Ctrl + D*.
2. Set the **Detail Size** value to **15**, either by using the slider under the **Enable Dyntopo** button or by pressing *Shift + D* and then moving the mouse to scale it bigger or smaller:



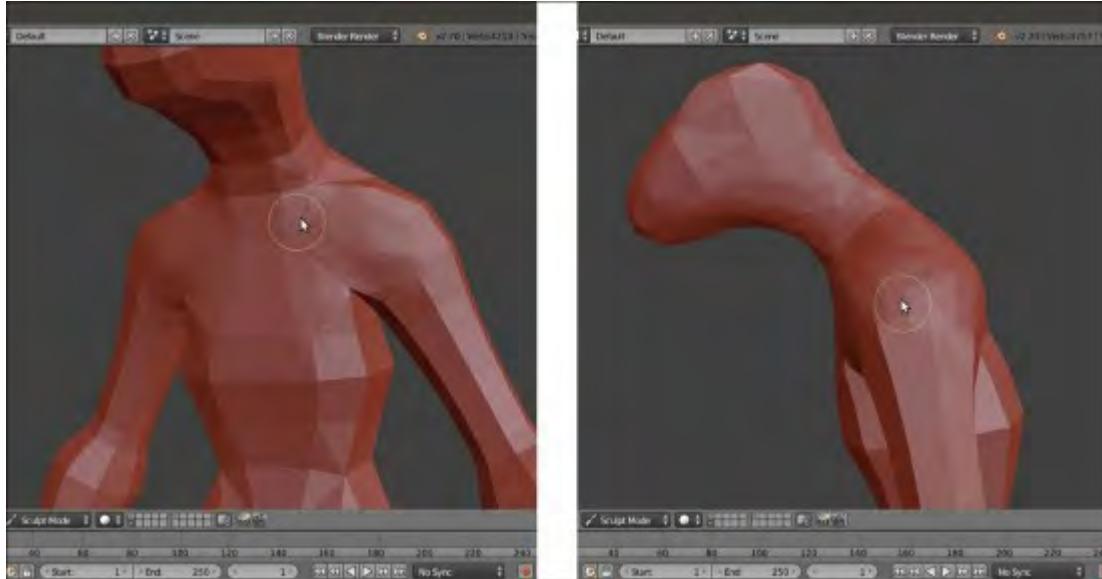
Starting to sculpt

3. Click on the **Brush** selection image (*Brush datablock for storing brush settings for painting and sculpting*) at the top of the **Tools** tab under the **Tool Shelf** panel, and, from the pop-up menu, select the **Scrape/Peaks** brush (otherwise press the *Shift + 3* key shortcut):



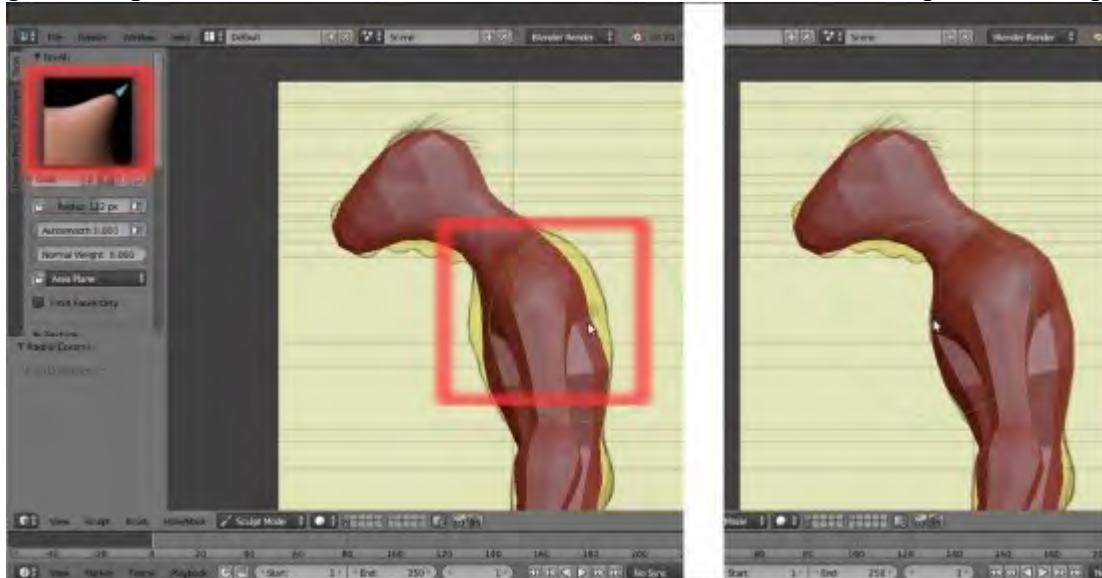
Selecting the Scrape/Peaks brush in the sculpt brushes menu

4. Start to scrape all the edges and soften the corners to obtain a smooth rounded surface:



Softening the edges of the mesh

5. Change the brush; select the **Grab** brush (**G** key) and press **3** in the numpad to go into **Side** view; press the **F** key and move the mouse cursor to scale the brush, in this case, to scale it much bigger, around **120** pixels (**Shift + F** is to change the strength of a brush, instead).
6. Using the **Background Image** showing in the orthographic view (the **5** key in the numpad), grab the **spine** and **chest** areas of the mesh and move them to fit the shape of the template:



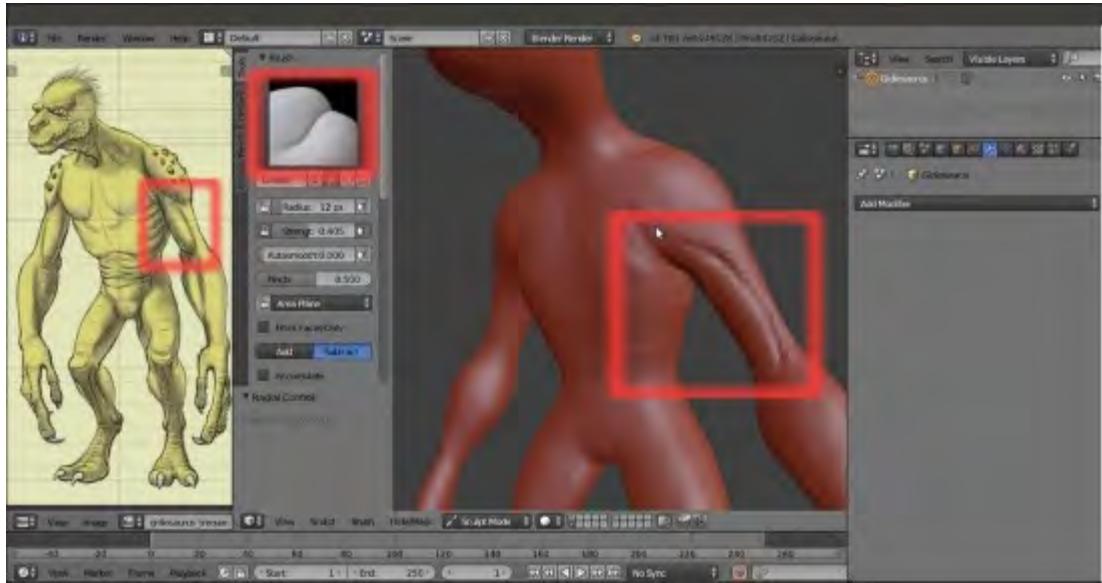
Using the Grab brush to modify the mesh

- Do the same for the other parts of the mesh that don't fit yet, and do it also in **Front** view (*I* key in the numpad) and **Back** view (*Shift + I* in the numpad).
- Select the **Scrape/Peaks** brush again (*Shift + 3* keys) and keep on softening the mesh until almost every part gets rounded and more organic-looking; you can also use the **Smooth** (*S* key) brush to further soften the mesh:



The character is starting to take a shape

- Open a new window, switch **Editor Type** to **UV/Image Editor** and click on the **Open** button in the toolbar; browse to the **templates** folder and select the **gidiosaurus_tquarters.png** image. Then, click on the little pin icon on the right-hand side of the image name on the window toolbar (*Display current image regardless of object selection*).
- Select the **Crease** brush (*4* key); using it as a chisel and following the loaded image as a reference, start to outline the character's more important features on the mesh, *drawing* the character's anatomy:



Using the Crease brush as a chisel

11. By pressing *Ctrl* while sculpting, we can temporarily reverse the effect of the brush; so, for example, the **Crease** brush, which usually carves lines in the mesh, can sculpt ridges and spike protrusions. We can use this to add details to the **elbow bones** and **knees** on the fly.
12. By pressing the *Shift* key while sculpting instead, we can temporarily switch whatever brush we are using with the **Smooth** brush, in order to instantly soften any newly added detail or feature.



Outlining the major body features

13. When finished with the **body**, exchange the brush for the **Clay Strips** brush (3 key), start to add stuff (the **nose**, **eyebrows**, and so on), and outline the features of the **head**. Again, press *Ctrl* to subtract clay (for the **eye sockets**, for instance) and *Shift* to soften.



Using the Clay Strips brush to add details and/or carve stuff

14. Always use the templates to check for the proportions and positions of the character's features. Also, use **Wireframe** mode if necessary, by going into **Ortho** view and comparing the sculpted mesh outline with the background template image; use the **Grab** brush to quickly move and shape proportionate features in the right places:



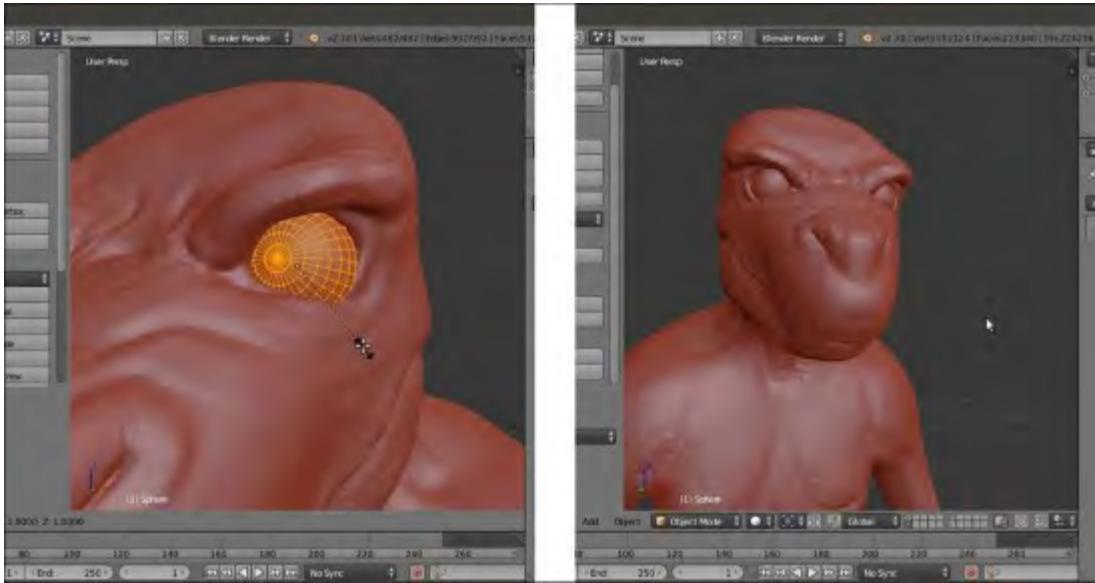
Temporarily switching to the Wireframe viewport shading mode to check the proportions in Side view

15. Using the **Clay Strips** (3 key), **Smooth** (S key), **SculptDraw** (*Shift + 4* key), **Crease** (4 key), and **Pinch** (*Shift + 2* key) brushes, build the **head** of the creature and define as many details as possible such as the **eyebrows**, **mouth rim**, **nostrils**, and **eye sockets**; experiment with all the different brushes:



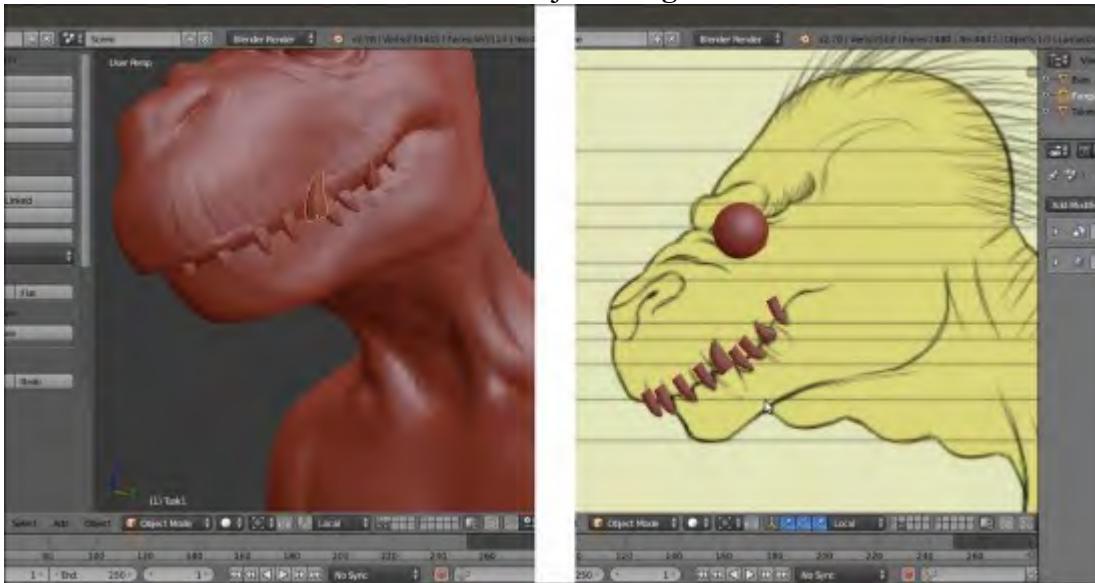
Detailing the head

16. Go out of **Sculpt Mode** and press *N* to make the **Properties** 3D window sidepanel appear; uncheck the **Only Render** item under the **Display** subpanel.
17. Press *Shift + A* and add a **UV Sphere** to the scene. Go into **Edit Mode**, if you haven't done so already, select all the vertices and rotate them **90** degrees on the *x* axis; then, scale them to **0.1000**. Finally, scale them again to **0.3600**.
18. Exit **Edit Mode** and move the **UV Sphere** to fit inside the **left eye socket** location.
19. Select the character's mesh and press *Shift + S*; then, in the **Snap** pop-up menu, choose **Cursor to Selected**. Select the **UV Sphere** and go to the **Tools** tab under the **Tool Shelf**; click on the **Set Origin** button and choose **Origin to 3D Cursor**. This way we have set the origin of the **UV Sphere** object at the same place as the character's mesh, while the **UV Sphere** mesh itself is located inside the **left eye socket**.
20. Go to the **Object Modifiers** window under the main **Properties** panel and assign a **Mirror** modifier to it.
21. Go to the **Outliner**, press *Ctrl + left-click* on the **UV Sphere** item, and rename it **Eyes**:



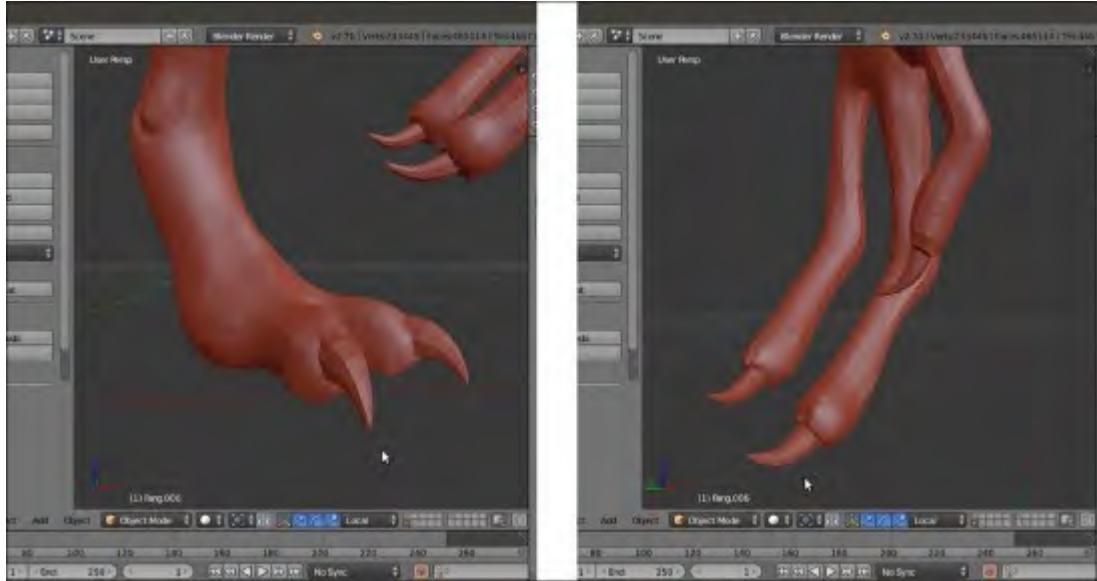
Positioning the eye spheres

22. Press **Shift + A** and add a **Cube** primitive. Go into **Edit Mode** and scale it a lot smaller; use the side template as a reference to modify by scaling, extruding, and tweaking the scaled **Cube's** vertices in order to build a low resolution **fang**. Go out of **Edit Mode** and go to the **Object Modifiers** window to assign a **Subdivision Surface** modifier.
23. Duplicate the **fang** and, as always, following the side and front templates as a guide, build all the necessary **teeth** for the **Gidiosaurus**.
24. Select all of them and press **Ctrl + J** to join them into one single object; press **Ctrl + A** to apply **Rotation & Scale**; then, do the same as in steps 19 and 20.
25. Go to the **Outliner** and rename the new object **Fangs**.



Making the teeth

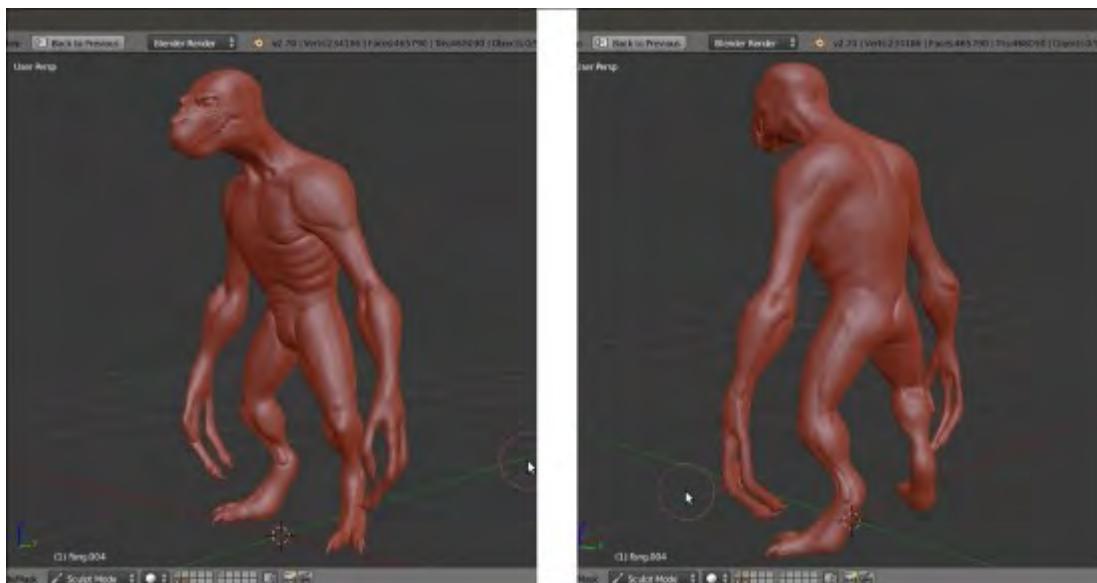
26. Add a new **Cube** and repeat the process to model the **talons** of the **hands** and **feet**:



Making the talons

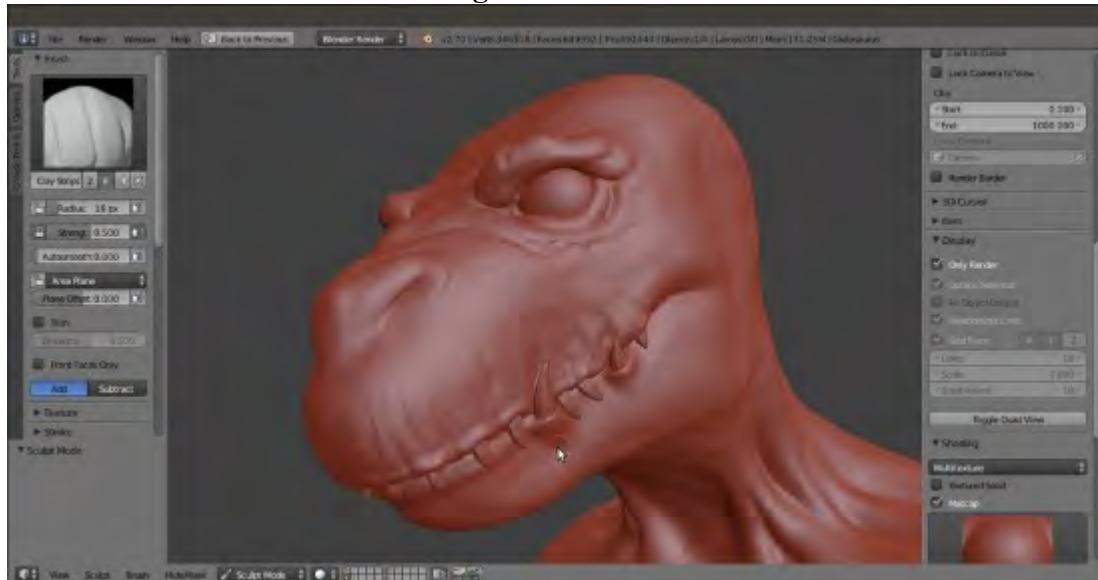
Note that the **Eyes**, **Fangs**, and **Talons** objects are not going to be sculpted, and therefore they are kept as separate objects. Later, we'll start to retopologize the sculpted body of the creature, while the **eyes** will be modeled and detailed in the traditional polygonal way; **fangs** and **talons** are good enough as they are.

27. Reselect the **Gidiosaurus** object and go back into **Sculpt Mode** to keep on refining the creature's shape more and more; don't be afraid to exaggerate the features, we can always smooth them later.



The almost completed sculpted mesh

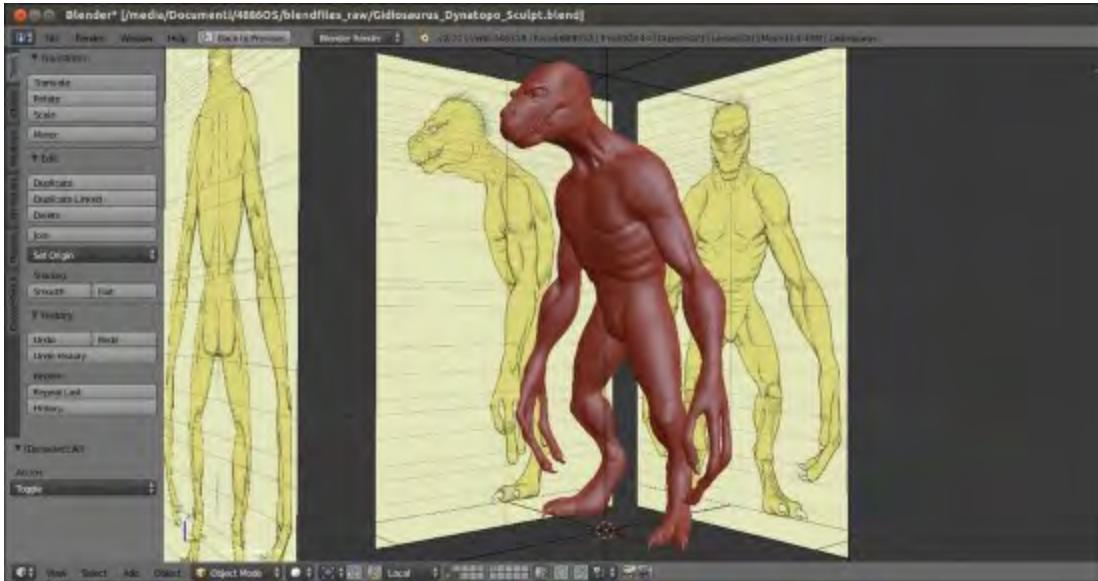
28. Adjust the shape of the **eyebrows** to perfectly fit the **Eyes** object; then, work more on the **mouth rim** to accommodate the **fangs**.



Refining the eyebrows and the mouth rim

29. When you think you have arrived at a good enough point, just go out of **Sculpt Mode** and *remember to save the file!*

Just a quick note: we don't actually need to go out of **Sculpt Mode** to save the file, it's possible to save it periodically (press *Ctrl + S* or *Ctrl + W* to save the file over itself, and *Ctrl + Shift + S* to *save as*) without needing to exit the sculpting session each time.



The completed sculpted mesh compared with the reference templates

So, here we are; the character's sculpting is basically done. We can work on it a lot more, tweaking the shapes further and adding details such as **scales**, **wrinkles**, and **veins**, but for this exercise's sake (and for this recipe), this is enough.

There's more...

A nice aspect of the **Dynamic topology** feature is the possibility to actually join different objects into a single mesh; for example, with our **Gidiosaurus**, we can join the **teeth** and the **talons** to the sculpted base mesh and then keep on sculpting the resulting object as a whole.

Actually, there are two ways to do this: simply by joining the objects and by the **Boolean** modifier.

To join the objects in the usual way, we can do the following:

1. Go out of **Sculpt Mode**.
2. Select the first object (that is, the **teeth**), press **Shift** + select the second object (the **talons**), and lastly press **Shift** + select the sculpted base mesh so that it's the active object (the final composited object will retain the active object's characteristics).
3. Press **Ctrl + J** and it is done!

This is the way you join objects in Blender in general, and it can actually work quite well. There is only one problem: there will always be a visible seam between the different objects, and although in the case of **teeth** or **talons** this will not be a problem, in other cases it should be avoided. Let's say you are working on a separated **head** and later you want to join it to a **body**; in this case, you don't want a visible seam between the **head** and **neck**, obviously!

So, the option is to use the **Boolean** modifier:

1. Go out of **Sculpt Mode**.
2. Select the character's base mesh and go to the **Object Modifiers** window under the main **Properties** panel; assign a **Boolean** modifier.
3. Click on the **Object** field of the modifier to select the object you want to join (let's say, the **Talons** object) and then click on the **Operation** button to the left to select **Union**.
4. Click on the **Apply** button to apply the modifier.
5. Hide, move onto a different layer, or delete the original object you joined (the **talons**).

Unlike the previous method, with Booleans, it will be possible to sculpt and smooth the joining of the different objects without leaving visible seams.

Chapter 3. Polygonal Modeling of the Character's Accessories

In this chapter, we will cover the following recipes:

- Preparing the scene for polygonal modeling
- Modeling the eye
- Modeling the armor plates
- Using the Mesh to Curve technique to add details

Introduction

In the previous two chapters, we did the following:

- Quickly modeled a simple base mesh, as close as possible to the shape of the reference templates
- Sculpted this base mesh, refining the shapes and adding details to some extent

We have also quickly modeled very simple teeth and talons, and placed bare UV Spheres as placeholders for the eyes.

It's now time to start some polygonal modeling to complete the eyes, but especially to build the armor that our character is wearing.

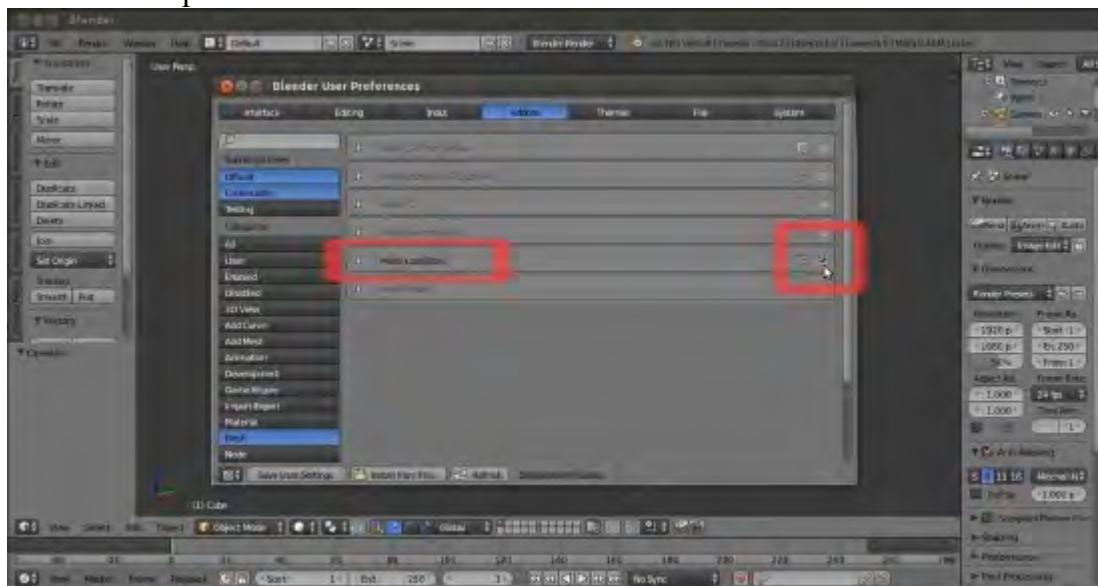
Preparing the scene for polygonal modeling

Coming from a sculpting session, our .blend file must first be prepared for the polygonal modeling, verifying that the required add-ons are enabled and all the character's parts are easily visible and recognizable; for this, even though the topic of **Materials** is complex and there will be an entire chapter dedicated to it later in this book, we are going to assign basic materials to these parts so that they have different colors in the 3D viewport.

Getting ready

First, we are going to look for the **LoopTools** add-on, an incredibly useful script by Bartius Crouch that extends the Blender modeling capabilities (and that also has other functionalities, as we'll see in the next chapter about retopology); this add-on is provided with the official Blender release, but still must be enabled. To do this, follow these steps:

1. Start Blender and call the **Blender User Preferences** panel (*Ctrl + Alt + U*); go to the **Addons** tab.
2. Under the **Categories** item on the left-hand side of the panel, click on **Mesh**.
3. Check the empty little checkbox on the right-hand side of the **Mesh: LoopTools** add-on to enable it.
4. Click on the **Save User Settings** button at the bottom-left of the panel to save your preferences and close the panel:



The Blender User Preferences panel

5. Open the `Gidiosaurus_Dynatopo_Sculpt.blend` file.

How to do it...

Now, we can start with the scene setup:

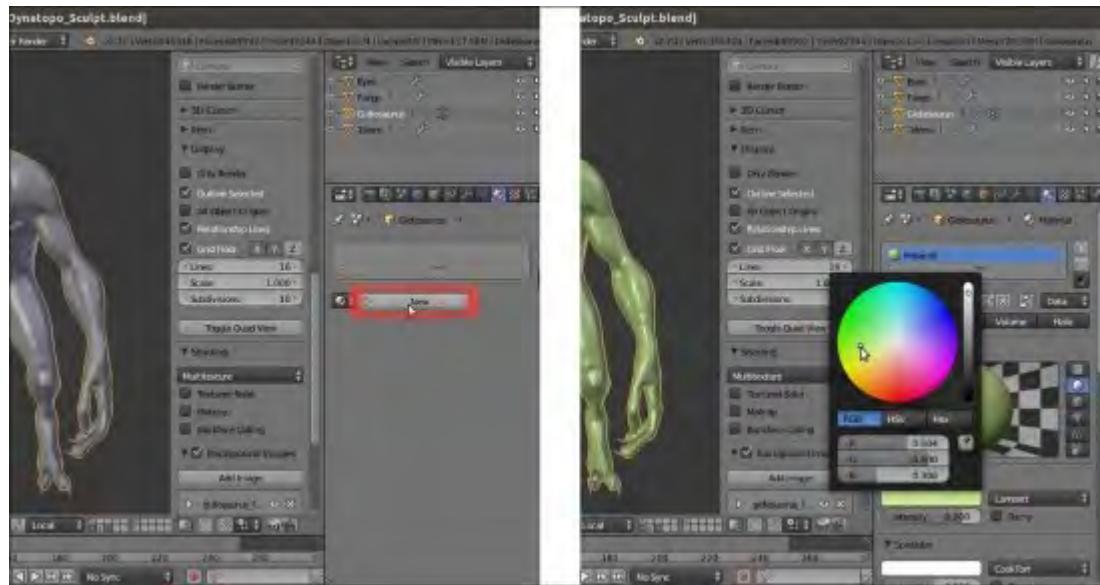
1. Click on the **11th** scene layer button (the first one in the second row of the first-left layer block of **Visible Layers** in the toolbar of the 3D window) to make it the only one visible (or else, just put the mouse pointer on the 3D viewport and press the *Alt + 11* keys; the *Alt* button is to allow for double digits).
2. Press *Shift* + left-click on the **13th** button to multiactivate it (or use the *Shift + Alt + 13* shortcut).
3. Go to the **Outliner** and click on the little grayed arrow icons on the side of the **Eyes**, **Fangs** and **Talons** items to make them selectable again.
4. If not already present, show the **Properties** 3D window sidepanel (*N* key) and go to the **Shading** subpanel; uncheck the **Matcap** item:



Disabling the Matcap item

5. Select the **Gidiosaurus** mesh; go to the **Material** window under the main **Properties** panel to the right and click on the **New** button to assign a material (note that, at least at the moment, we are using the default **Blender Internal** engine); click on the **Diffuse** button and change the color to **RGB 0.604, 0.800, 0.306** (a greenish hue, but in this case you can obviously choose any color you wish). Double left-click on the material name inside the data block slot to rename it as **Body**.
6. Select the **Eyes** object and again in the **Material** window under the main **Properties** panel to the right, click on the **New** button to assign a new material; click on the **Diffuse** button and this time change the color to **RGB 0.800, 0.466, 0.000**. Rename the material as **Eyes**.
7. Select the **Fangs** object and repeat the process; change the diffuse color to **RGB 0.800, 0.697, 0.415**. Rename the material as **Enamel**.

- Select the **Talons** object and go to the **Material** window under the **Properties** panel to the right; click on the little arrows on the left-side of the **New** button and from the pop-up menu, select the **Enamel** material:



Assigning a material and choosing a color

- Go to the **UV/Image_Editor** window on the left-hand side of the screen and press *Shift* + left-click on the **X** icon on the right-hand side of the data block name to get rid of the **gidiosaurus_tquarters.png** image. Then, click on the **Open** button, browse to the **templates** folder, and load the **gidiosaurus_armor1.png** image.
- Save the file as **Gidiosaurus_modeling.blend**.



The armoured character's image loaded in the UV/Image Editor for reference

How it works...

We have deselected the **Matcap** view, assigning also differently colored basic materials to the four parts making up the character's mesh (**body**, **eyes**, **fangs**, and **talons**) to have a clearer way of differentiating the different pieces of the mesh. Then, we have replaced the template we used as reference for the sculpting of the **Gidiosaurus** body with a new one showing the armor as well (in the **templates** folder there are actually two slightly different versions of the armor; we chose the first one).

We have also activated the **13th** scene layer to be ready for the modeling of the **armor** (in the **11th** we have the character's mesh and in the **12th** we have the **fangs**, **talons**, and **eyes**).

Note that, in this cookbook, I will always specify scene layers to indicate the **20** 3D layers accessible from the buttons on the viewport toolbar and distinguish them from other types of layer systems present in Blender, such as for the **bones** or the **Grease Pencil** tool and so on.

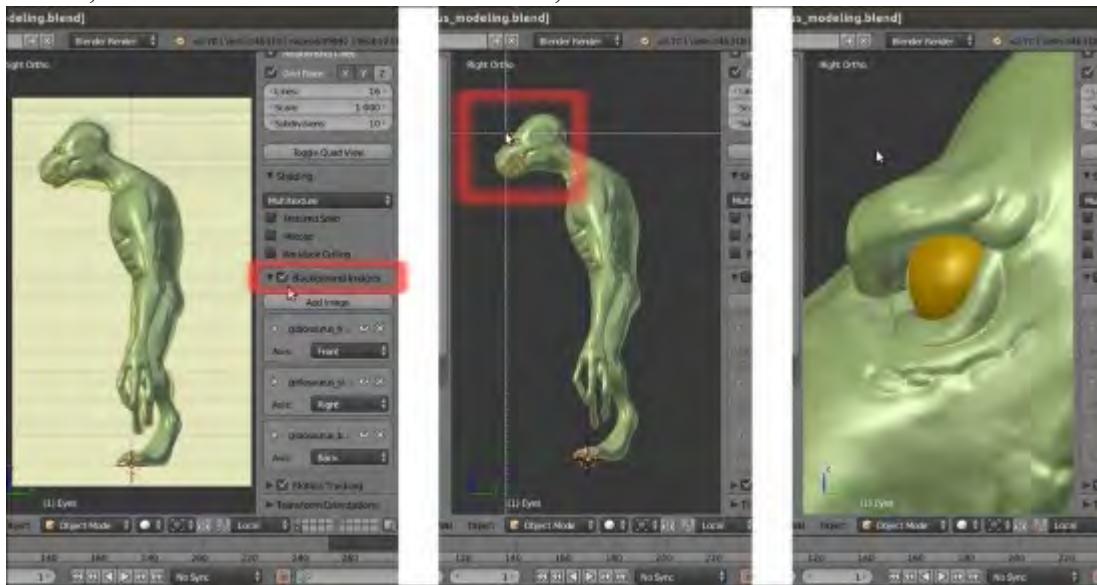
Modeling the eye

It's now time to start to define the creature's **eyes**. We already had **UV Sphere** placeholders, but we're going to refine this mesh to deliver a more convincing eye. By the way, keep in mind that a good portion of the expressiveness of the eye will be due to the use of appropriate textures; for more information, see [Chapter 12, Creating the Materials in Cycles](#), and [Chapter 13, Creating the Materials in Blender Internal](#).

Getting ready

Following the previous recipe, there is nothing particular to be prepared before starting, except for the following:

1. Go to the **Properties** 3D view sidepanel (*N* key if not already present) and uncheck the **Background Images** item.
2. Press *3* on the numpad to go in **Side** view and zoom to the **UV Sphere** location, by pressing *Shift + B* and drawing a box around the point you want to zoom at; as you release the mouse button, the selected area will be zoomed in;



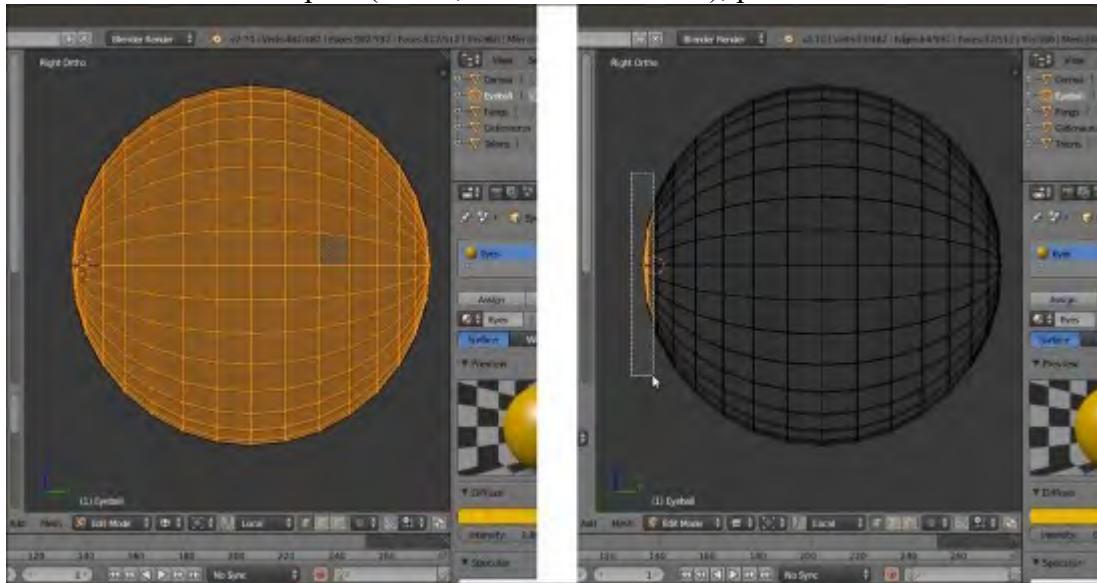
Disabling the background images and zooming to the eyes area

3. Go to the **Outliner** and click on the eye icon on the right-hand side of the **Gidiosaurus** item to hide it; or else, select the mesh in the 3D viewport and press the *H* key. Alternatively, you can also press the slash (/) key in the numpad to go in **Local** view, a particular view mode where only the selected objects are still visible (press the slash (/) again to go back to the normal view mode).

How to do it...

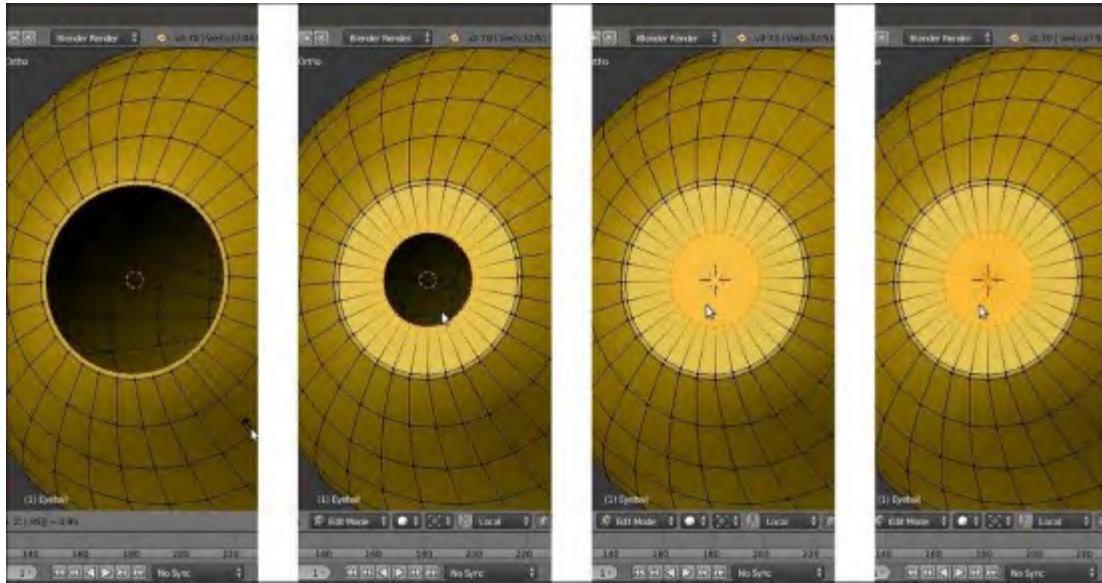
Without further ado, let us begin to build the eye:

1. Press **Z** to go in the **Wireframe** viewport shading mode.
2. In the **Outliner**, select the **Eyes** item (or else, if you wish, in the 3D viewport, select the **UV Sphere** object) and rename it as **Cornea**.
3. Press **Shift + D** and then immediately press the **Esc** key or right-click to cancel the *Grab/Translate* function, obtaining a duplicated object that now shows as **Cornea.001**; in the **Outliner**, rename the new object as **Eyeball**.
4. Press **Tab** to go in **Edit Mode**; if necessary, press **A** to select all the vertices and scale them to **0.990 (S | .99 | Enter)**.
5. Press **A** to deselect all the vertices. Then, box-select (**B** key) the pole vertex and the first row of vertices at the left-side pole (that is, in total **33** vertices); press **X** to delete them:



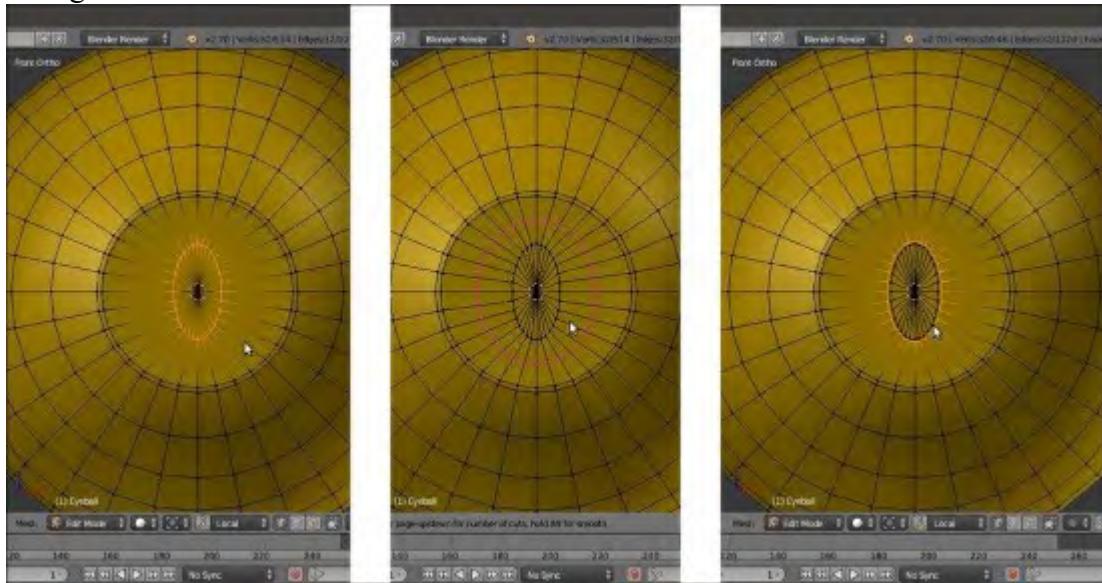
Box-selecting the vertices at the UV Sphere pole

6. Reselect all the remaining vertices; then, press the period (.) key on the numpad to center the view on the selection.
7. Go to the **Outliner** and click on the eye icon on the left-hand side of the **Cornea** item to hide it.
8. Rotate the view to align it with the hole in the **UV Sphere** and, if necessary, press the **5** key on the numpad to go in **Ortho** mode.
9. Press **Z** to go in the **Solid** viewport shading mode and press **A** to deselect everything.
10. Select the first row of vertices around the hole (**Alt + right-click** on the edge-loop). Press **E** to extrude them and then **S** to scale them; keep **Ctrl + Shift** pressed and scale to **0.9500** (or else, press **S | .95 | Enter**).
11. Press **E** and **S** again to extrude and scale the vertices to **0.500**.
12. Press **F** to fill the selection and **Alt + P** to poke the created N-gon face (that is, to automatically subdivide the single N-gon face into triangular faces connected to a central vertex).



Extruding and closing the eye

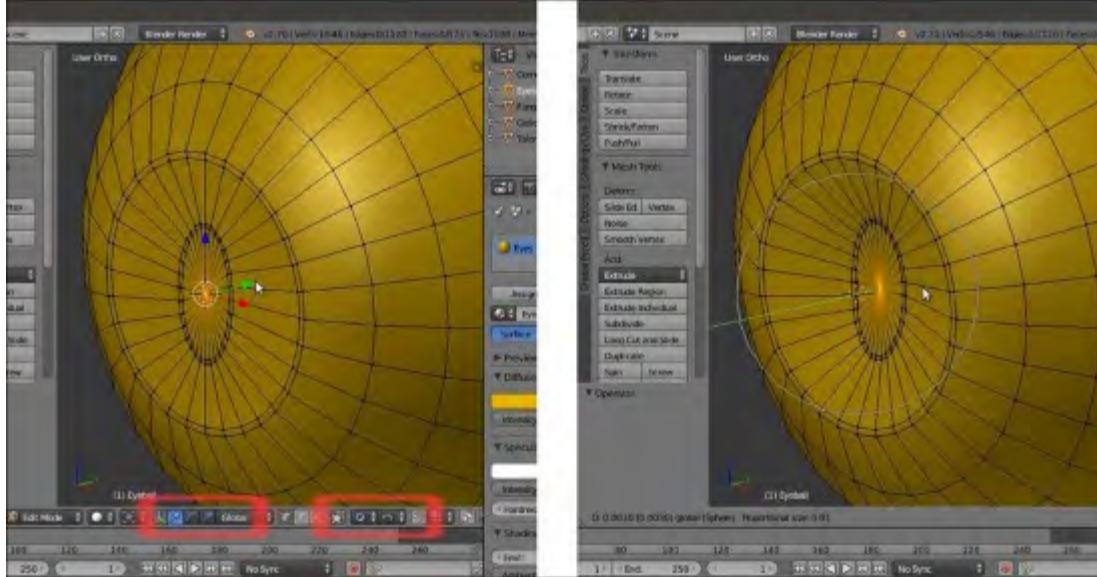
13. Press **I** on the numpad to go in **Front** view. Scale the selected vertices to **0.500** on the **x** axis (**S | X | .5 | Enter**).
14. Press **Ctrl + R** and add an edge-loop outside of the iris; keep **Ctrl** pressed and move the mouse to edge-slide it to **-0.900**.



Making the pupil

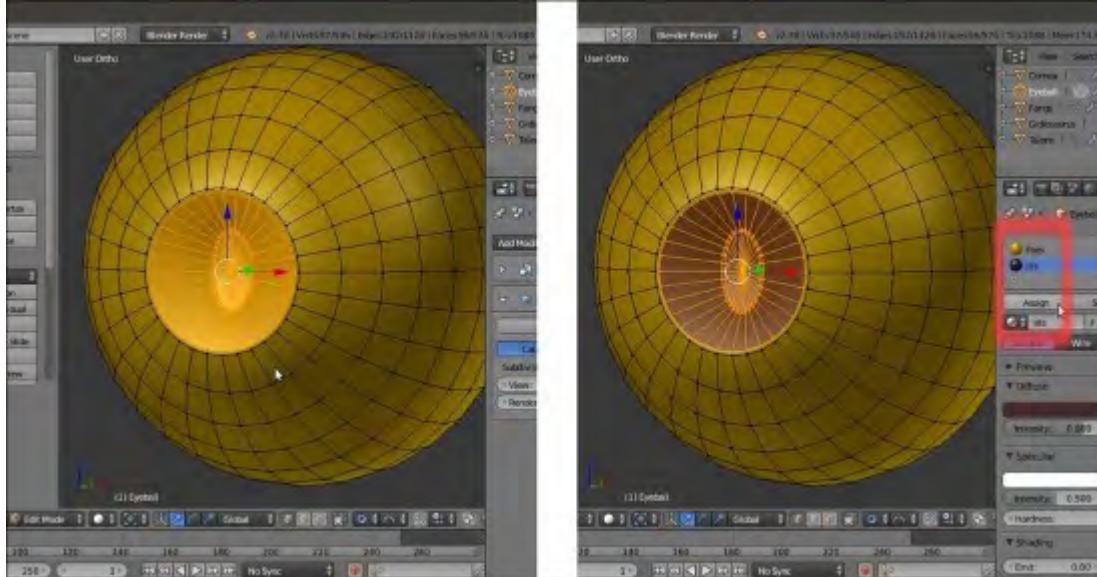
15. In the toolbar of the 3D window, enable the **PET** (the **Proportional Editing tool**); set it to **Connected** and the **Proportional Editing Falloff** option to **Sphere**.

16. Enable the widget, set it to *Translate* (the second icon from the left, the one with the arrow), set **Transform Orientation** to **Global**, and select the central vertex of the pole. By using the widget, move it on the y (green) axis to **0.0030** (click on the green arrow and hold *Shift* for a finer control as you move the mouse on the y axis), while with the middle mouse wheel, set the **Proportional size** value of the PET to a quite small radius, or **0.01** to be precise:



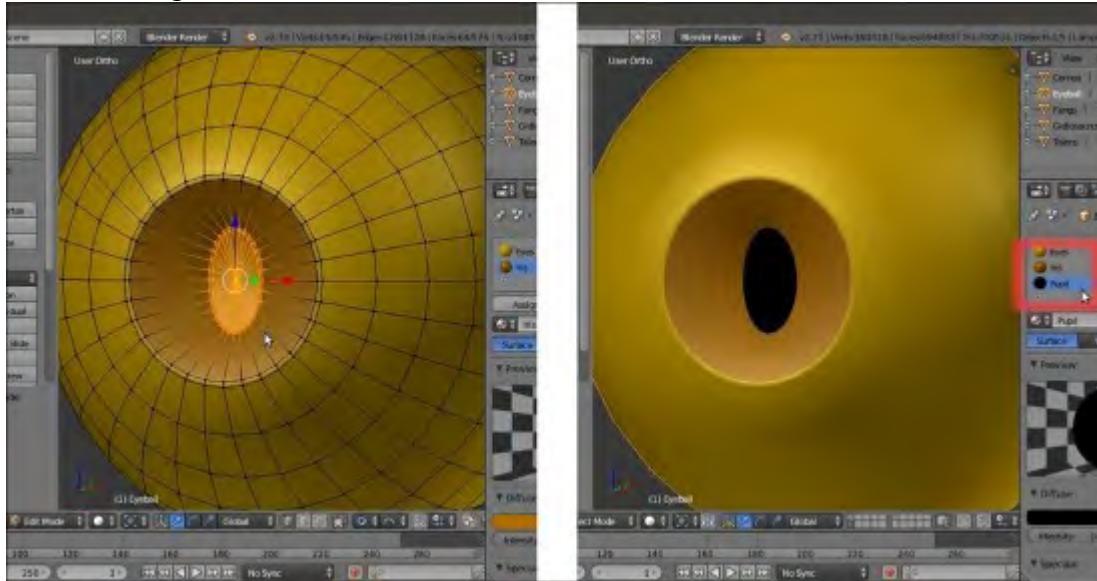
Creating the iris concave shape

17. Press *Ctrl* and the + key on the numpad 3 times, in order to grow the selection starting from the single selected vertex at the center of the iris.
 18. Go to the **Material** window, create a new material, and rename it as **Iris**; change its diffuse color to something like **RGB 0.061, 0.025, 0.028** and then click on the **Assign** button:



Assigning a material to the iris

19. Press **Ctrl** and the **-** key on the numpad just **1** time, in order to reduce the selection to the pupil. Go to the **Material** window, create a new material, and rename it as **Pupil**; change its diffuse color to plain black and then click on the **Assign** button.
20. Press **Tab** to go out of **Edit Mode**.



The almost completed eye

21. Go to the **Object Modifiers** window under the main **Properties** panel on the right-hand side of the UI and assign a **Subdivision Surface** modifier; check the **Optimal Display** item.
22. In the **Outliner**, unhide the **Cornea** object and assign a **Subdivision Surface** modifier as well; check the **Optimal Display** item and then hide it again (you can also use the **H** and **Alt + H** keys to do this).
23. Select the **Eyeball** object and go to the **Material** window; select the **Pupil** material and go to the **Specular** subpanel to set the **Intensity** value to **0.000**. Set the **Specular Shader Model** option of both the **Eyes** and **Iris** materials to **WardIso** and the **Slope** value to **0.070**. Set the **Iris** material's **Emit** value (under the **Shading** subpanel) to **0.050**.
24. In the **Outliner**, select the **Cornea** object and in the **Material** window, click on the little icon reporting **2** on the right-hand side of the material name (it's the display of the number of users for that material). The name **Eyes** automatically changes to **Eyes . 001**: rename it **Cornea**; then, go to the **Transparency** subpanel and enable it. Set the **Fresnel** value to **1.400** and the **Blend** factor to **2.000**. Go to the **Options** subpanel further down and uncheck the **Traceable** item.
25. Unhide the **Gidiosaurus** mesh (**Alt + H**) and enable the **6th** scene layer (the one with the **Camera** and the **Lamp**). Select the **Lamp** and in the **Object Data** window, change the type to **Sun** and then rotate it to: **X = 55.788948°**, **Y = 16.162031°**, and **Z = 19.84318°**; you can press **N** and then type these values in the slots of the **Rotation** panel at the top of the **Properties 3D** window sidepanel.

26. Press **N** to hide again the **Properties** 3D window sidepanel and in the toolbar of the 3D window, go to the **Viewport Shading** button and select **Rendered** (or directly press the *Shift + Z* shortcut) to have a nice preview of the effect:



The Rendered preview of our character so far

27. Save the file.

How it works...

Actually, the **eyes** of the character are composed of two distinct objects: the **Eyeball** and the **Cornea** object.

The **Cornea** object is the transparent layer covering the **Eyeball** object, and by clicking on the eye icon in the **Outliner**, it has been made invisible in the 3D viewport but still renderable. With the **Cornea** object visible in the 3D views, **irises** and **pupils** would have been hidden behind, making the work of animating the **eyes** quite hard; animators always need to know what the character is looking at.

Both the **Cornea** and **Eyeball** objects, at the moment, are mirrored to the right by the **Mirror** modifier; this will be changed when we skin the mesh to the **Armature**.

If you can't find the **Rendered** view in the **Viewport Shading** mode button on the 3D viewport's toolbar, you may want to make sure you have the latest version of Blender; only versions after **2.6** have this feature for the Blender Render engine.

Modeling the armor plates

In the previous recipe, we modeled the character's **eye** and we had already modeled the **teeth** in [Chapter 2, Sculpting the Character's Base Mesh](#), because we needed them, at that moment, to go on with the sculpting; they had been made with simple **Cube** primitives quickly scaled and tweaked in **Edit Mode**.

It is now time to model the **armor** for our warrior. Let's begin by creating the hard metal plates. We are going to use an approach similar to the modeling of the **fangs**, which is by starting with a **Cube** primitive and subdividing it to have more geometry to be edited in the proper shape, and we'll also use the **LoopTools** add-on to simplify some processes.

Getting ready

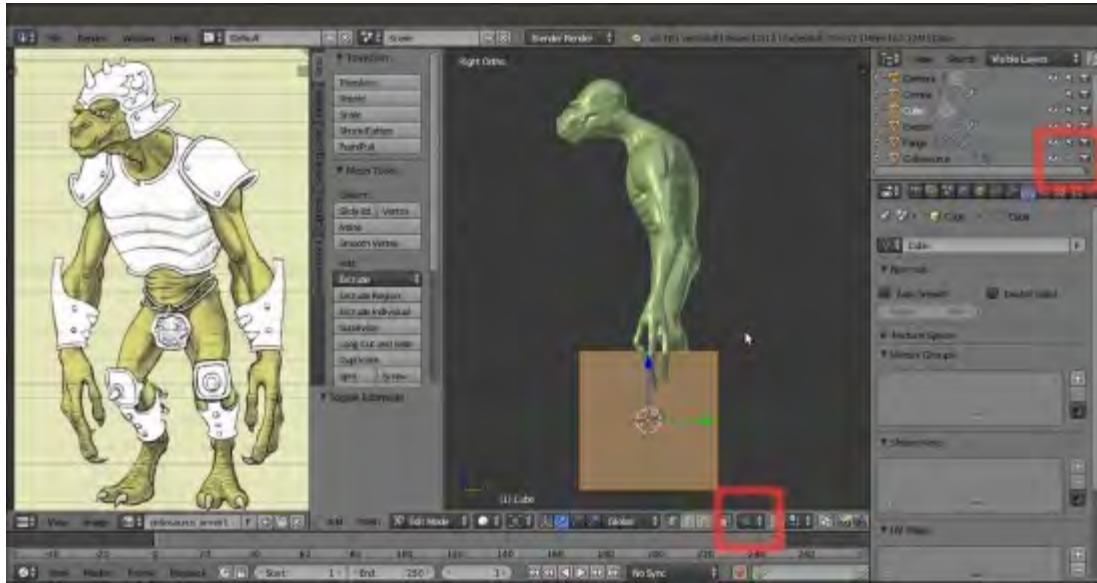
We will carry on with the `Gidiosaurus_modeling.blend` file:

1. Press **3** on the numpad to go in **Side** view.
2. By scrolling the middle mouse wheel, zoom back to frame the **Gidiosaurus** mesh in the 3D window.
3. In the **Outliner**, click on the arrow icon on the right-hand side of the **Gidiosaurus** item to make it unselectable.

How to do it...

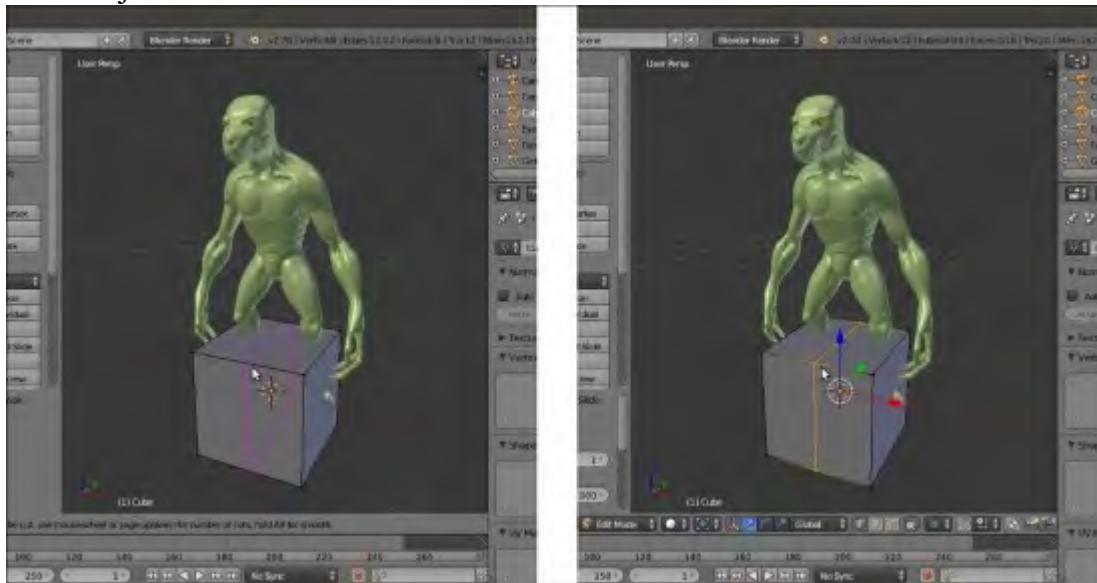
Now, we can start to build the **armor**; let's go with the **chest** piece:

1. Note that the **3D Cursor** is in the middle of the scene, at the character's pivot location (**Shift + S** | *Cursor to Selected* or also *Cursor to Active*, just in case).
2. Press **O** to disable the **Proportional Editing** tool; go to the 3D viewport toolbar to verify that the tool button is grayed.
3. Press **Shift + A** and add a **Cube** primitive to the scene.
4. Press **Tab** to go in **Edit Mode** and scale all the vertices to **0.500** (or press **S** | **.5** | *Enter*).



Adding the Cube primitive to the scene

5. Press *Ctrl + R* to add a loop along the y axis and then left-click twice to confirm it at the middle of the object:



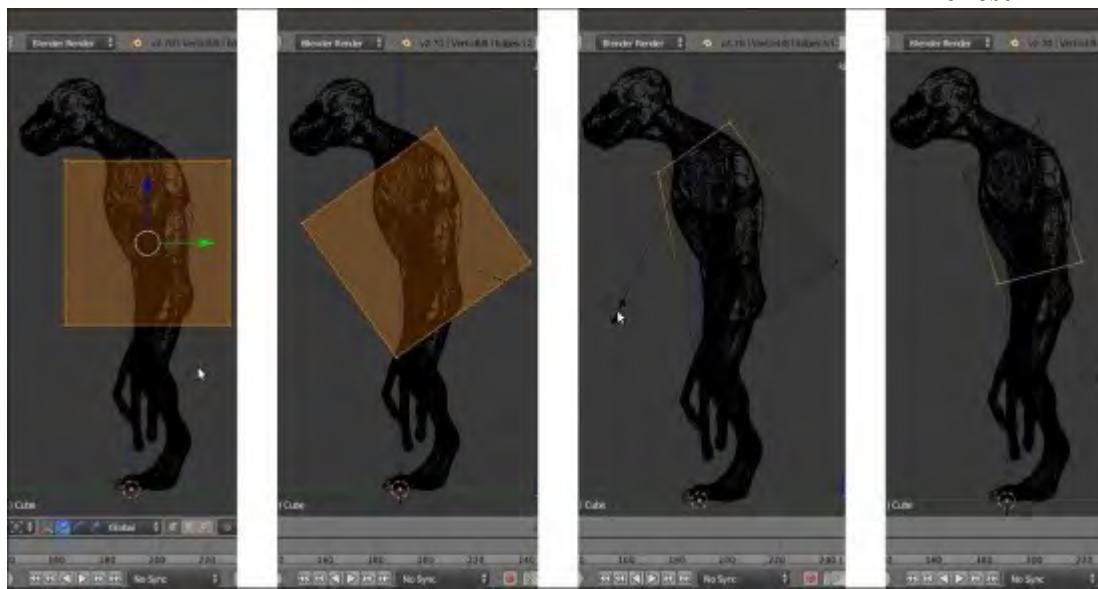
Adding a central vertical edge-loop to the Cube

6. Select the right-side vertices of the **Cube** and delete them; then, assign a **Mirror** modifier and check the **Clipping** item:



The Cube with the Mirror modifier

7. Go again in **Side** view and press Z to go in the **Wireframe** viewport shading mode; select all the vertices and move them upward.
8. Rotate the vertices to reflect the angle of the character's **chest**.
9. Select the upper vertices and scale and rotate them to fit the creature's **neck** area.
10. Select the bottom vertices and scale and rotate them to fit the base of the **chest**:



Starting to model the armor from the Cube primitive

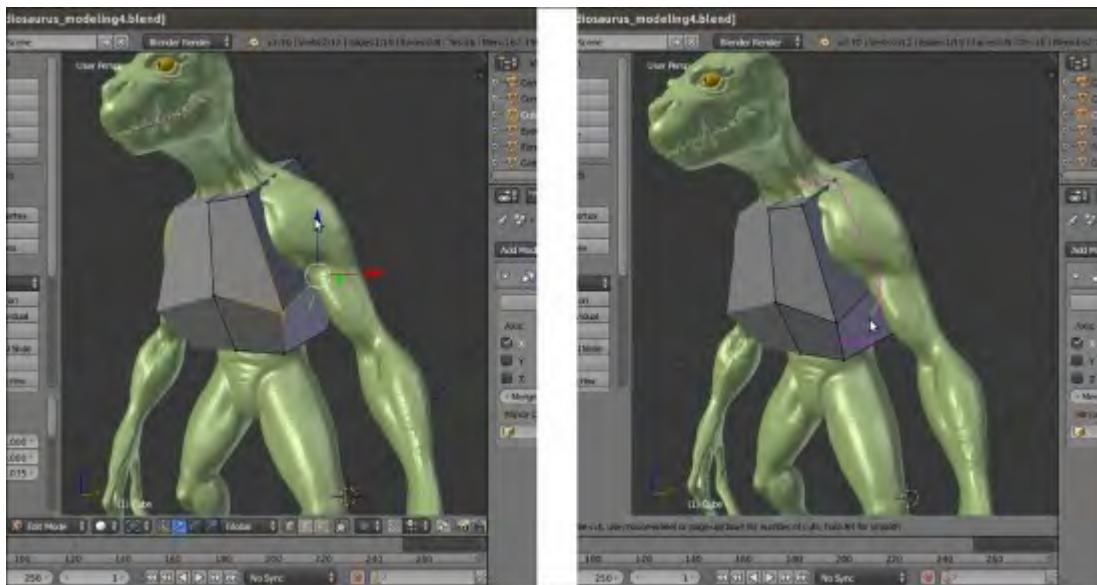
11. Press **Ctrl + R** to add a new horizontal edge-loop at the middle of the **Cube**; scale it bigger to fit the shape of the creature's **chest**.

12. While still in **Side** view, grab and move the vertices to conform them to the **chest** shape.
13. Press *I* to go in **Front** view and again move the vertices to adjust them consistently to the character's **chest** shape:



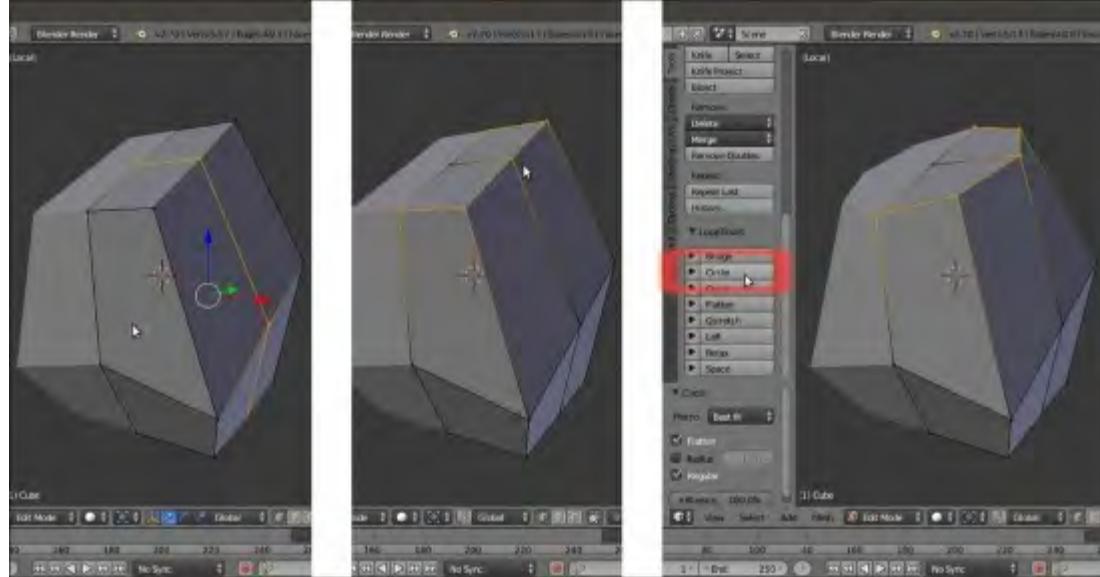
Adding more geometry and shape to the Cube

14. Select the **2** middle outer vertices and move them down, in order to place the edge connecting them just below the character's **armpit**.
15. Press *Ctrl + R* to add a loop along the *x* axis; click twice to confirm it at the middle of the lateral side:



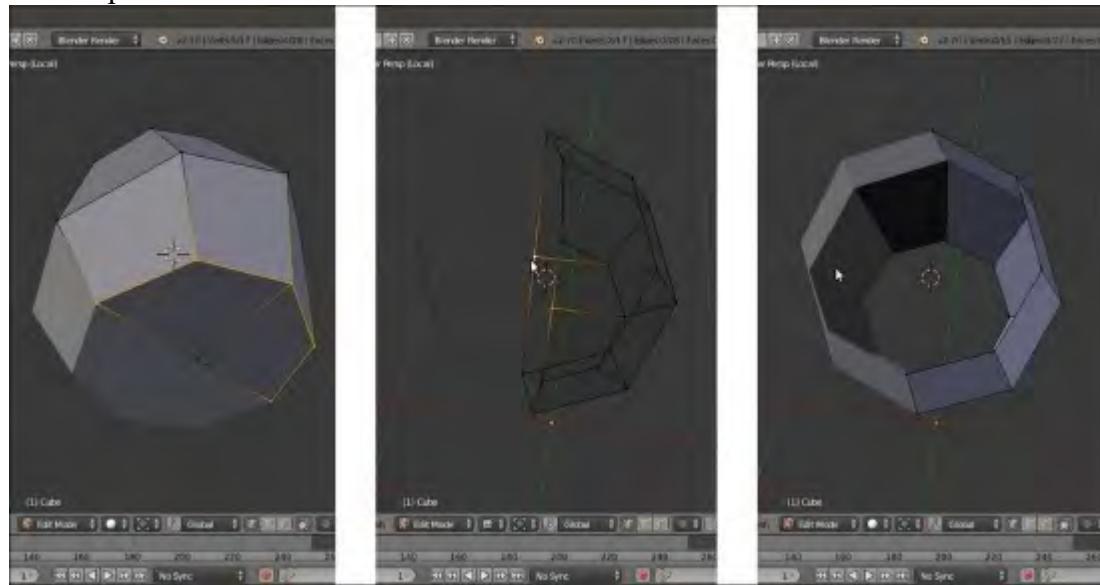
Adding more geometry again

16. Press the slash key (/) on the numpad to go in **Local** view with the selected object (in this case, even if still in **Edit Mode**, it is the **Cube**) and select the upper outer edge-loop.
17. Go to the **Tool Shelf** panel and scroll down the **Tools** tab to find the **LoopTools** subpanel (the **LoopTools** items are available also in the **Specials** menu that we can call by pressing the W key in **Edit Mode**); click on the **Circle** button to make the selection on a circular path:



Using the LoopTools add-on

18. Do the same also with the middle and the bottom edge-loop; then, select the central upper and bottom pole's vertices and delete them:



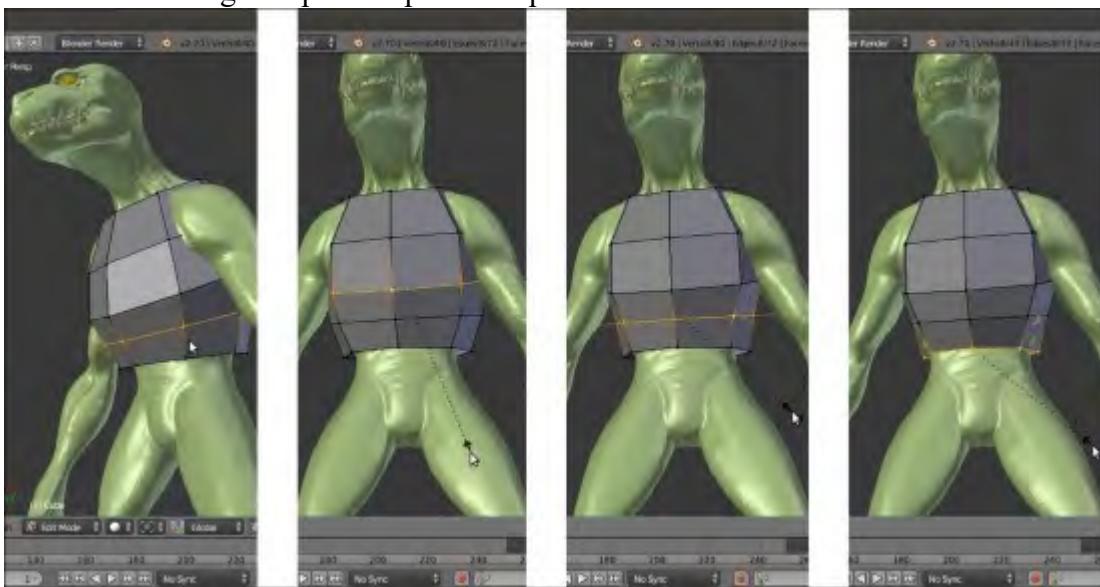
Going on with the modeling

19. Press the slash key (/) on the numpad to go out of **Local** view.
20. Press **Tab** to go out of **Edit Mode** and go to the **Object Modifiers** window under the main **Properties** panel; click on the **Apply** button to apply the **Mirror** modifier.
21. Go back in **Edit Mode** and press **Ctrl + R** to add a horizontal edge-loop to the upper half of the mesh.
22. Scale the new edge-loop to **1.100**:



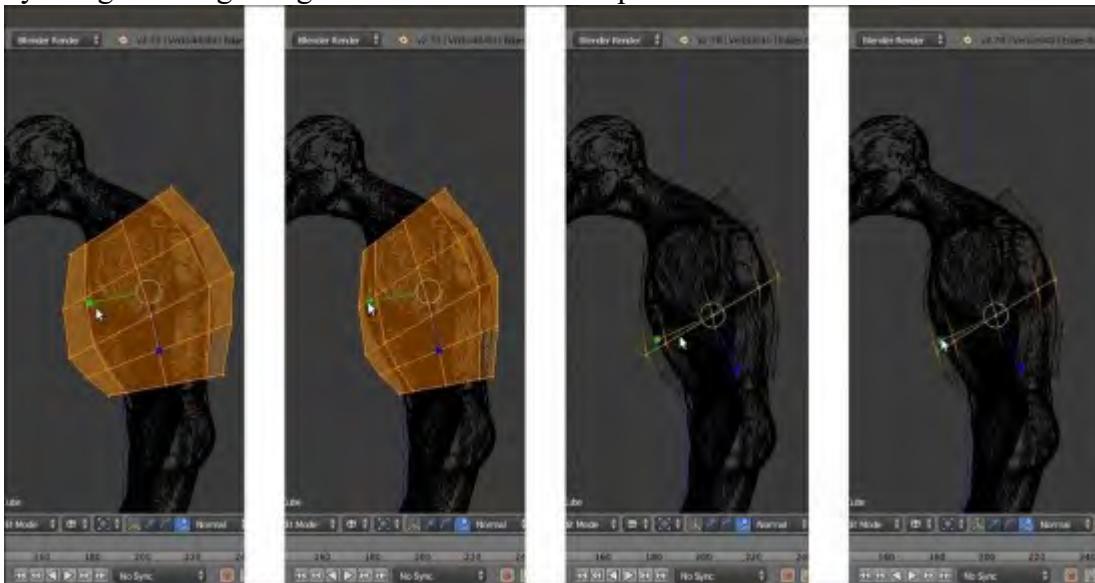
Adapting the shape of the armor to the chest by adding more geometry as edge-loops

23. Add a new horizontal edge-loop also to the lower half of the mesh.
24. Select the middle edge-loop and scale it smaller on the *x* axis, to **0.900**.
25. Select the bottom edge-loop and scale it smaller on the *x* axis as well.
26. Select the last edge-loop and repeat the operation.



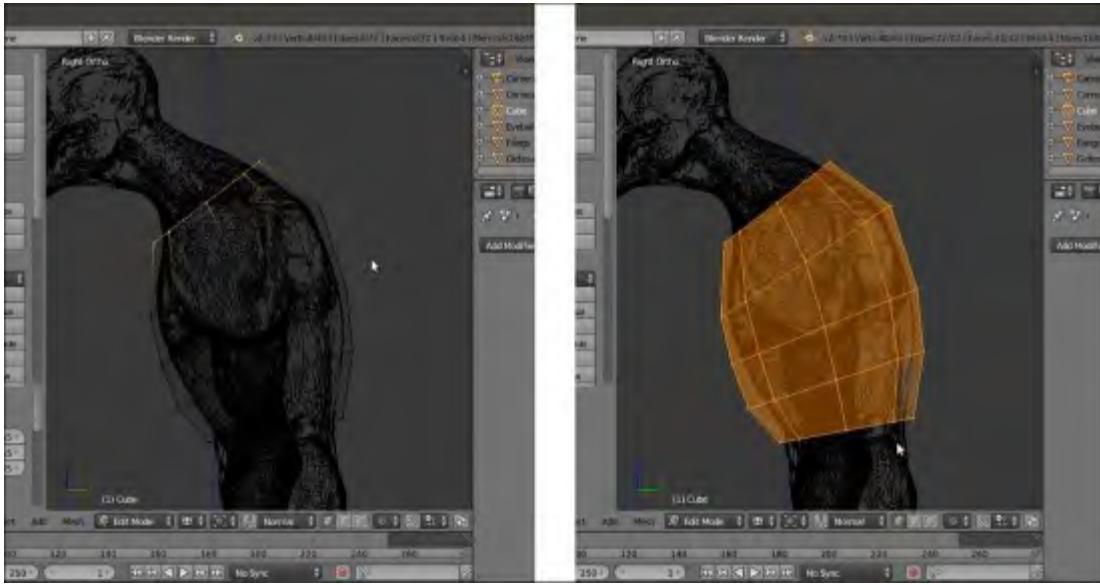
Going on with the modeling by adding edge-loops

27. Press 3 on the numpad to go in **Side** view and Z to go in the **Wireframe** viewport shading mode.
28. If not already, enable the widget in the toolbar of the 3D window; set the **Transformation manipulators** to **scaling** (the last icon to the right) and the **Transform Orientation** option to **Normal**.
29. Select all the vertices and by moving the green scaling manipulator of the widget, scale smaller all the edge-loops on the normal y axis; small enough to almost reach the character's **back** and **chest** surfaces.
30. Deselect everything and then select the middle edge-loop (press *Alt* + right-click); scale it again by using the widget to get close to the **torso** shape:



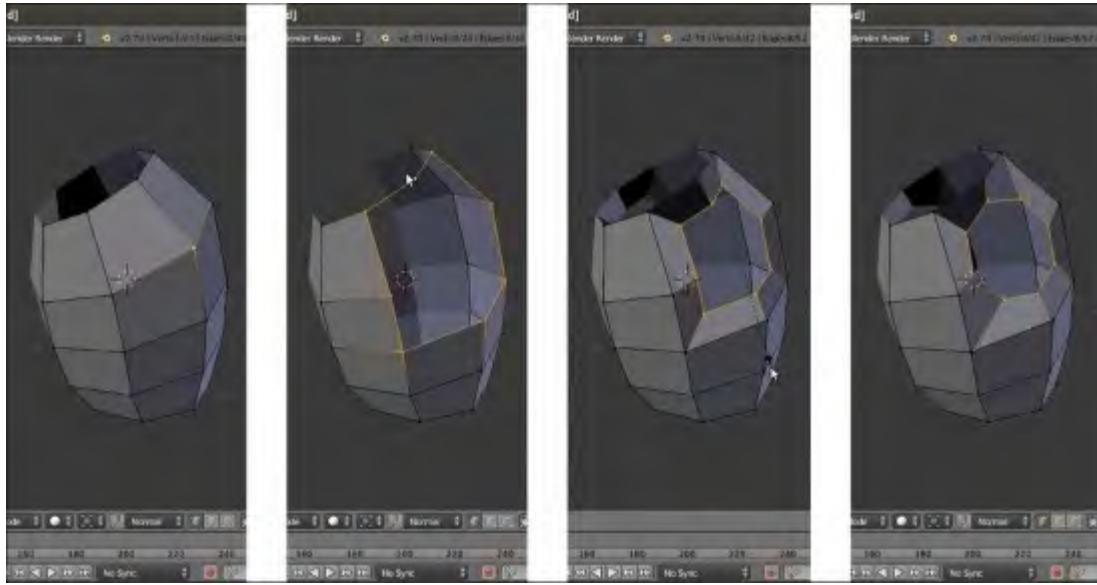
Adjusting the chest armor depth

31. Do the same with the other edge-loops by selecting them individually, rotating and scaling them, and also by moving the vertices.



Refining the lateral profile of the armor

32. Press the slash (/) key on the numpad to go again in **Local** view.
33. Select the right-side vertices of the **Cube** and delete them.
34. Go to the **Object Modifiers** panel and assign a new **Mirror** modifier; as usual, check the **Clipping** item.
35. Select the central vertex on the upper-side part and delete it.
36. Select the resulting loop of edges around the resulting hole (you can press *Alt* + right-click and then *Shift* + right-click to add the remaining unselected top vertex to the selection; it doesn't get selected with the edge-loop because there are no faces connecting it to the other vertices, but only edges).
37. Press *E* and then *S* to extrude new faces and scale them (about **0.600**).
38. Select the new edge-loop and go to the **LoopTools** panel under the **Tools** tab of the **Tool Shelf** panel; click on the **Circle** button to make it rounded:



Adding the arm's holes

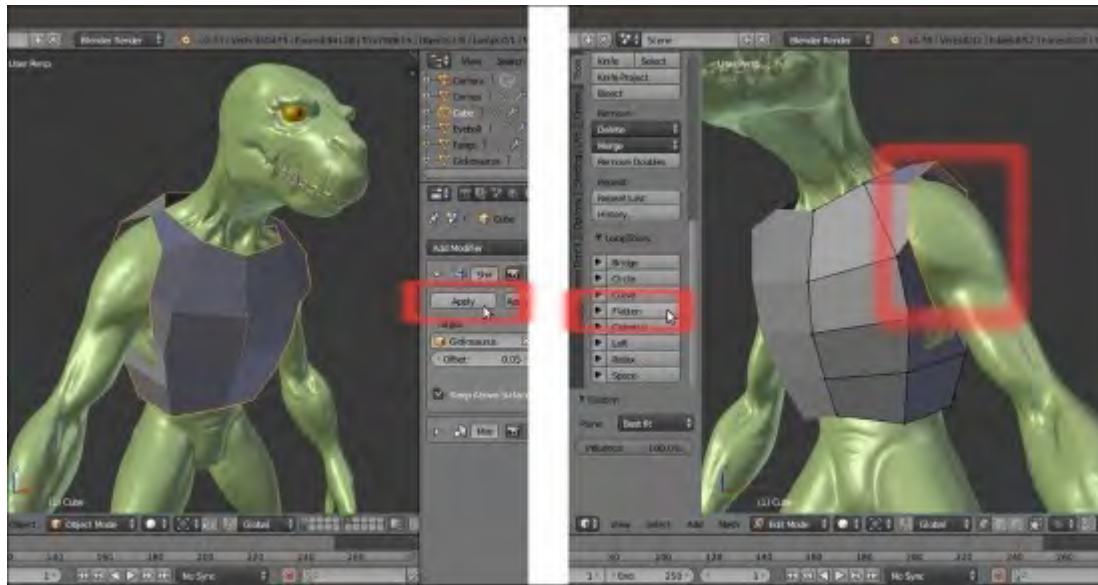
39. Go in **Front** view and move the edge-loop outward.
40. Go out of **Edit Mode** and press the slash (/) key on the numpad to go out of **Local** view.
41. Press *I* on the numpad to go again in **Front** view. Go to the **Object Modifiers** panel and assign a **Shrinkwrap** modifier to the **Cube**; check the **Keep Above Surface** item and in the **Target** field, select the **Gidiosaurus** name:



Assigning the Shrinkwrap modifier to the chest armor

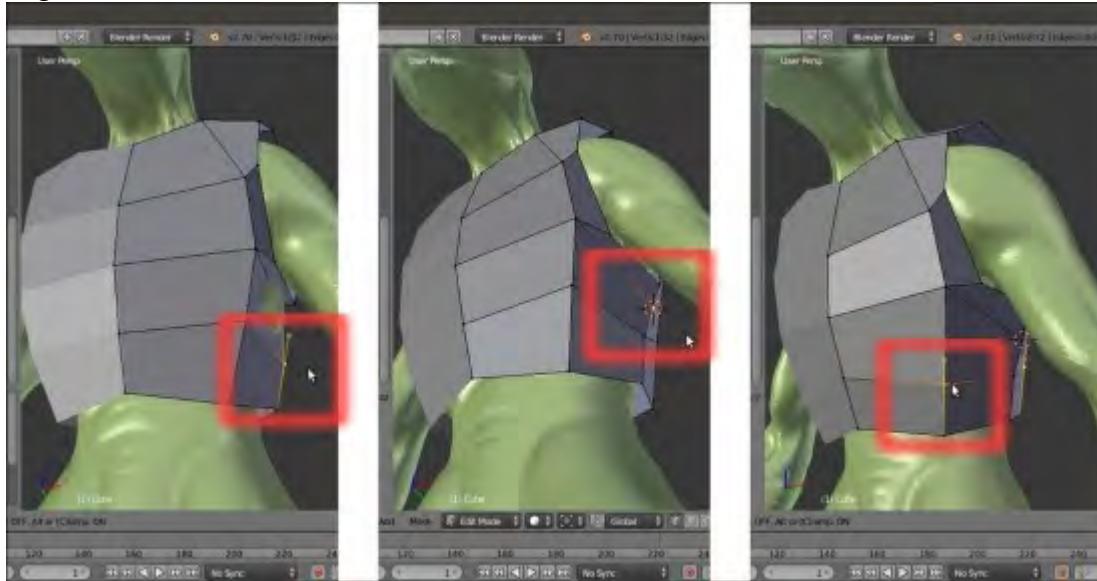
42. Set the **Offset** value to **0.05**.
43. Move the **Shrinkwrap** modifier to the top of the modifier stack and click on the **Apply** button.

44. Go in **Edit Mode** and select the **shoulder** edge-loop; go to the **LoopTools** panel and click on the **Flatten** button:



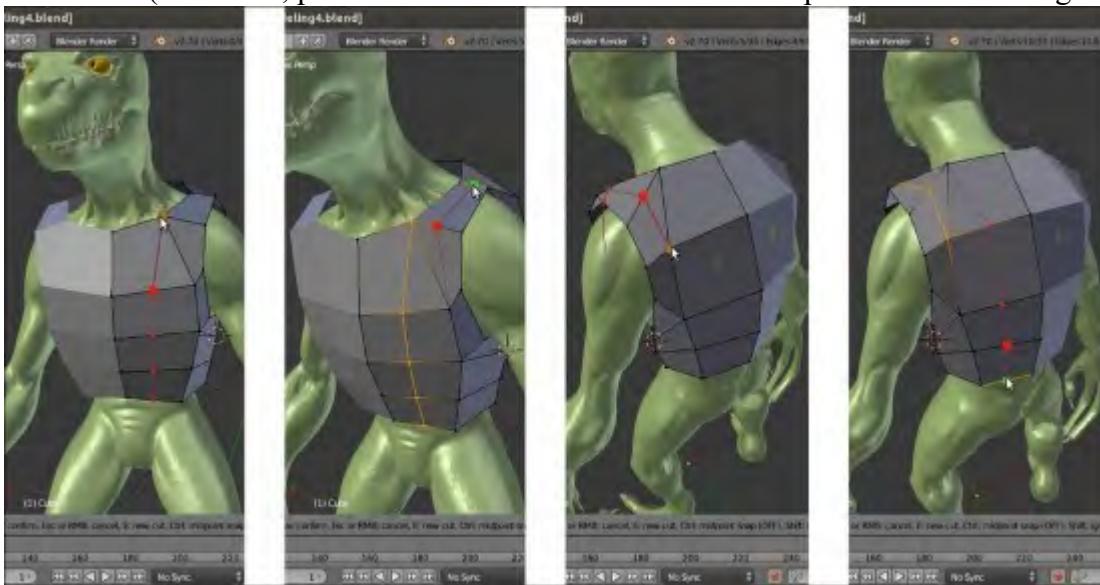
Refining the modeling through the LoopTools add-on

45. Fix, below the **armpit**, the lateral vertices that are curved inwards, by using the **Alt + S** shortcut to move them outward along their normal, the **3D Cursor** and the **Snap** pop-up menu (**Shift + S**) to place them midway from other vertices, and the **Shift + V** shortcut to slide them along the edges:



Tweaking vertices

46. Press the **K** key to activate the **Knife Topology Tool**; by keeping the **Ctrl** key pressed to constrain the cuts to the middle of the edges, cut a new edge-loop as shown in the following screenshot (each time, press *Enter* to confirm the cut and then pass to the following one):



Using the Knife tool

47. Press **Alt + J** to join the already selected triangular faces into quads:

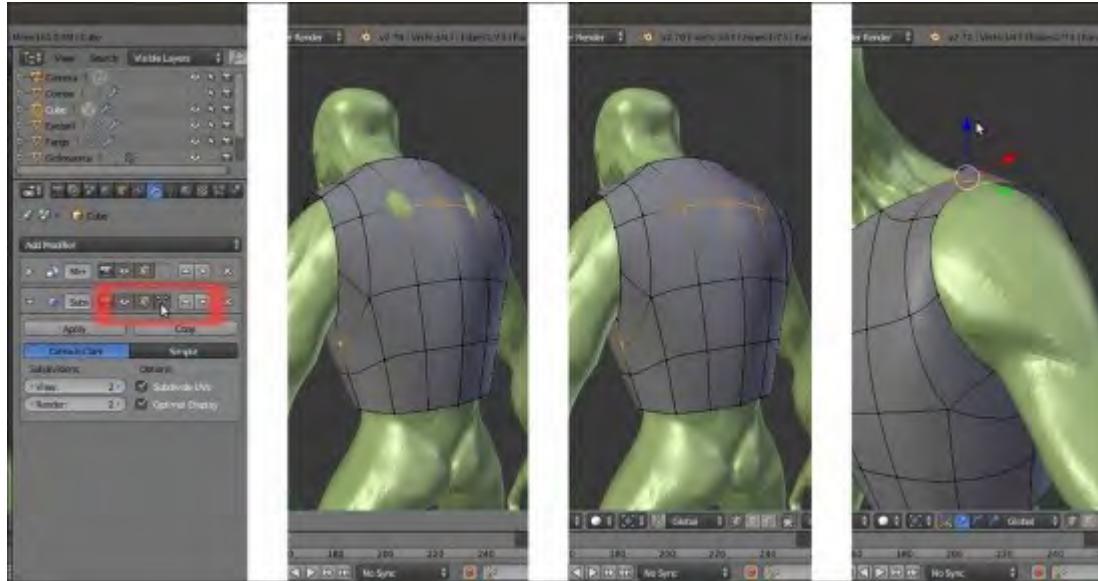


Joining two triangular faces into one quad face

48. Select all the vertices and press **Ctrl + N** to recalculate the normals.
 49. Deselect all the vertices and go to the **Object Modifiers** window; assign a **Subdivision Surface** modifier, set the **Subdivisions** level for **View** to 2, and check the **Optimal Display** item. Click

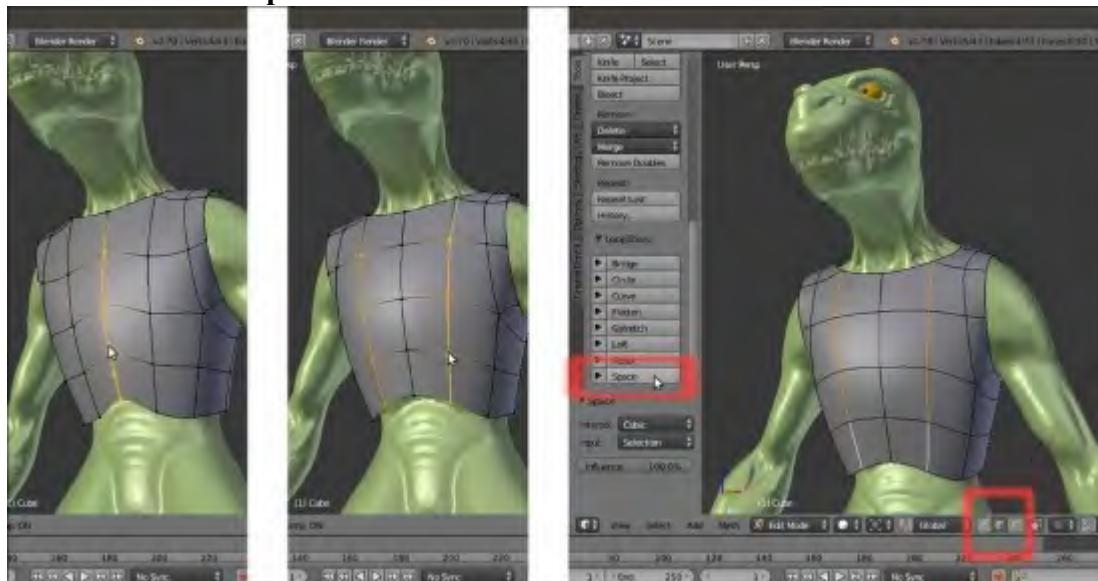
on the *Adjust edit cage to modifier result* icon, the last one to the right with the editing triangle, in order to see the effect of the modifier in **Edit Mode**.

50. Go out of **Edit Mode** and then go to the **Tools** tab under the **Tool Shelf**; under the **Edit** subpanel, select the **Smooth** shading.
51. Go back in **Edit Mode** and select the vertices (in our case, mainly on the side and back) corresponding to areas where the sculpted mesh is overlapping the **armor**. Press *Alt + S* to scale their position along their normals and so fix the overlapping; then, select the upper vertices of the **shoulder** and move them closer to the character's **shoulder** surface:



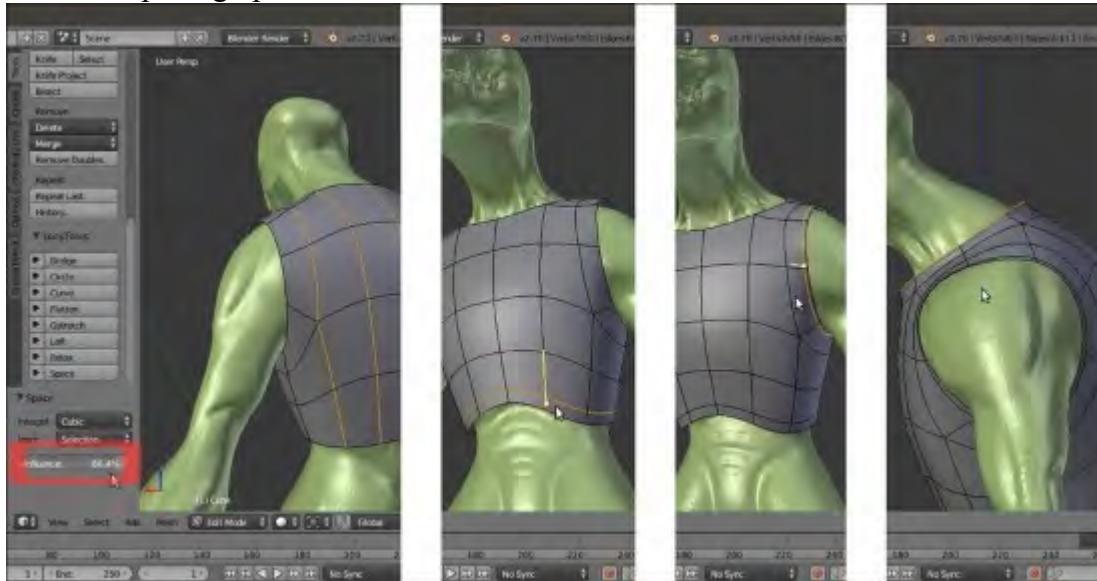
Tweaking vertices with the visible cage of the Subdivision Surface modifier

52. Repeat the operations of the previous step on all the vertices that need it; select the vertices on the **belly** and press *Shift + V* to move them upward, but along the edges to model the arc shape at the bottom of the **plate**:



Sliding the vertices and adjusting the polygonal flow through the LoopTools add-on

53. Select the edges of the front and back and click on the **Space** button in the **LoopTools** add-on panel; if needed, tweak the value of the **Influence** slider at the bottom of the **Tools** tab to set the amount for the operation.
54. Add edge-loops at the bottom of the **armor** and at the **shoulder** opening to create a rim; extrude the **neck** opening upwards to create a kind of short collar:



Extruding geometry

55. Add edge-loops on the front of the **chest** plate as shown in the following screenshot (*Ctrl + R* and then slide it to **0.500**) and then select the front vertices of the alternate edge-loops and in **Side** view, move them forward.
56. Select the last bottom edge-loop and scale it bigger (to **1.100**):



Adding edge-loops to add detailing to the armor

57. Move the front vertices of the **breast** and **belly** downward, using the image loaded in **UV/ Image Editor** as reference. Add more edge-loops to add definition to the front of the **chest** plate (in the following screenshot, the three added edge-loops are selected at the same time only to highlight them; in Blender, they must be added one at a time). Then, smooth the resulting oddly spaced back vertices by using the **Space** button of the **LoopTools** add-on:



Making the edges' length even through the LoopTools add-on

58. In the **Outliner**, rename the **Cube** item as **Breastplate** (by either double-left-clicking or by pressing **Ctrl + left-click** on the item).

59. Then, go to the **Material** window under the main **Properties** panel and assign a new material to the **Breastplate** object; rename the material as **Armor_dark**. Set the diffuse color to **RGB 0.605, 0.596, 0.686** and the **Diffuse Shader Model** option to **Oren-Nayar**; set the specular color to **RGB 0.599, 0.857, 1.000** and the **Specular Shader Model** option to **WardIso**; set the **Intensity** value to **0.164** and the **Slope** value to **0.100**. Under the **Shading** subpanel, check the **Cubic Interpolation** item.
60. Go to the **Object Modifiers** window and assign a **Solidify** modifier; move it up in the stack, before the **Subdivision Surface** modifier. Set the **Thickness** value to **0.0150** and check the **Even Thickness** item.
61. Set **Viewport Shading** to **Rendered** to have a quick preview (be sure to have the proper scene layers activated, that is, the **6th** for the lighting and the **11th** and **13th** for the **character** and **armor**). Then, go to the **World** window under the main **Properties** panel and activate the **Indirect Lighting** tab; then, click on the **Approximate** button under the **Gather** subpanel. For the moment, leave the rest as it is:



The Rendered result so far, with some World Lighting setting

62. Save the file.

How it works...

This is the usual polygonal modeling process that is common to most aspects of 3D packages. Starting from a **Cube** primitive, we moved and arranged the vertices to model the **chest** armor plate, extruding and also adding new edge-loops by using the **Knife Topology Tool** and the **Ctrl + R** shortcut.

We used the **Mirror** modifier to work only on half of the mesh and to have the other half automatically updated. In some cases, we had to temporarily apply the **Mirror** modifier to better scale the edges as complete circles (otherwise, they would have been half circles with odd scaling pivot points); then, we had to delete the vertices from one side and assign the **Mirror** modifier again.

At a certain point, as the **armor's** shape got more defined, we started to tweak the vertices in **Edit Mode**, but with the **Subdivision Surface** modifier applied to the editing cage in order to have the right feedback while conforming the **armor's** shape to the **character's** shape.

We also used a few of the options available in the **LoopTools** add-on that has been revealed to be an incredibly handy aid in the modeling process.

See also

- <https://sites.google.com/site/bartiuscrouch/looptools>
- <http://www.blender.org/manual/modeling/index.html>

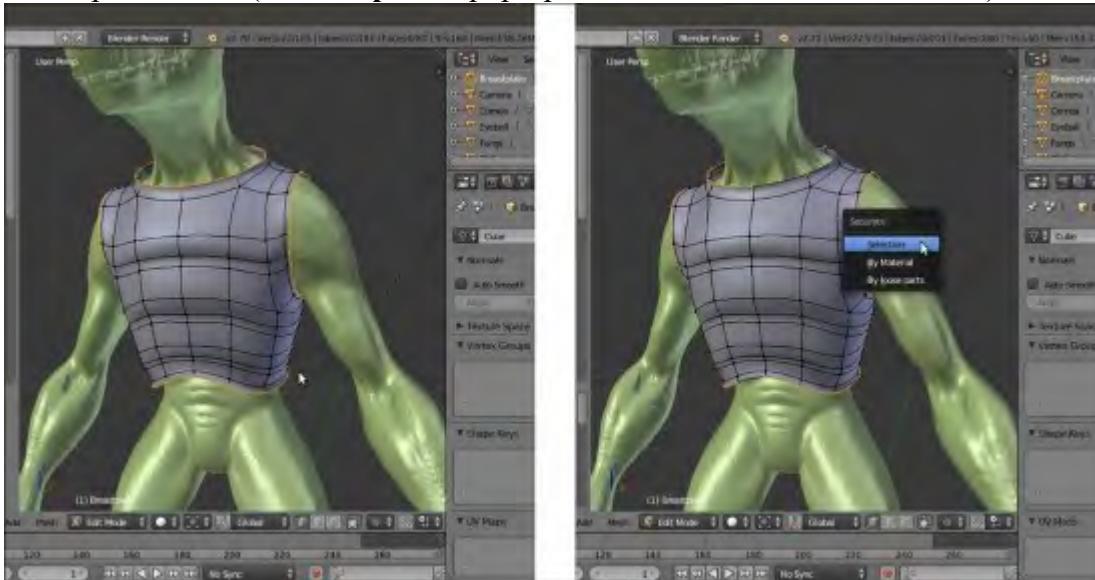
Using the Mesh to Curve technique to add details

In the previous recipe, we modeled the basic bulk of the **Breastplate**. We are now going to see a simple but effective technique to add detailing to the borders of the **armor** plate.

How to do it...

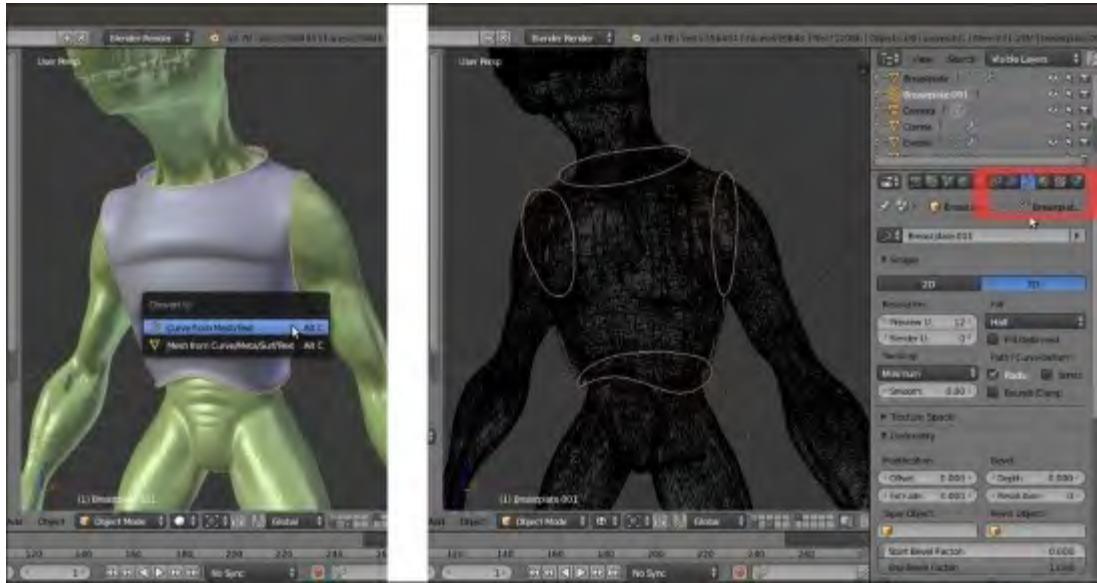
Assuming we have gone out of **Edit Mode** and then saved the file, reopen the `Gidiosaurus_modeling.blend` file and proceed with the following:

1. Go back in **Edit Mode** and select the edge-loop around the **neck** (*Alt* + right-click), the edge-loop around the **shoulder** hole (*Alt* + *Shift* + right-click), and the last one at the base of the **Breastplate** (*Alt* + *Shift* + right-click again).
2. Press *Shift* + *D* and soon after, the right-mouse button to duplicate without moving them; press *P* to separate them (in the **Separate** pop-up menu, choose the **Selection** item):



Separating geometry by selection

3. Go out of **Edit Mode** to select the **Breastplate.001** object (the duplicated edge-loops).
4. Press *Alt* + *C* and in the **Convert to** pop-up menu, select the first item: **Curve from Mesh/Text**.
5. The mesh edge-loops actually get converted into **Curve** objects, as you can see in the **Object Data** window under the main **Properties** panel on the right-hand side of the UI:



Converting the geometry in Curves

6. In the **Object Data** window, under the **Geometry** tab, set the **Extrude** value to **0.002** and the **Depth** value to **0.010**; then, under the **Shape** tab, set the **Fill** mode to **Full**:



"Modeling" the Curves by the settings

7. Press **Alt + C** and this time, in the **Convert to** pop-up menu, select the second item: **Mesh from Curve/Meta/Surf/Text**.
8. Press **Tab** to go in **Edit Mode**, press **A** to select all the vertices, and in the **Mesh Tools** tab under the **Tools** tab in the **Tool Shelf** panel, click on the **Remove Doubles** button (note that in the top

- main header, a message appears: **Removed 2240 vertices**; so always remember to remove the doubles after a conversion!).
9. Go out of **Edit Mode** and click on the **Smooth** button in the **Edit** subpanel; in the **Outliner**, rename it as **Breastplate_decorations**.
 10. Assign a **Subdivision Surface** modifier, with the **Subdivision** level as **2** and **Optimal Display** enabled.
 11. Go to the **Material** window and assign a new material; rename it as **Armor_light** and copy all the settings and options from the **Armor_dark** material, except for the diffuse and the specular colors—set them to **RGB 1.000** (pure white; a faster way is to assign the **Armor_dark** material, make it a single user, change the colors to white, and rename the material as **Armor_light**).



Assigning a new material

12. As always, remember to save the file.

How it works...

Even if at first sight this seems a complex process, actually it's one of the easiest and fastest ways to model a mesh. We have just duplicated the edge-loops that are located where we had the intention of adding the modeled borders. With a simple shortcut, we have converted them to a **curve** object that can be beveled both by other curve objects or simply by values to be inserted in the fields under the **Geometry** tab. Then, once we obtained the shape we wanted, we converted the curve back to a mesh object.

We could have kept the armor decorations as **curves**, but by converting them to meshes, we have the opportunity to unwrap them for the mapping of the textures according to the rest of the **armor**.

Note that the **Preview U** value under the **Resolution** item in the **Shape** subpanel for the **curve** objects should be kept low if you don't want a resulting mesh with a lot of vertices; you can set it quite lower

than the default **12**. Just experiment before the final conversion, while keeping in mind that once converted to mesh, the **decorations** will probably be smoothed by a **Subdivision Surface** modifier with the rest of the **armor**; in any case, the obtained **decorations** mesh can also be simplified at a successive stage.

In this chapter, we saw the process that can be used to model the **armor** meshes. We will not demonstrate the rest of the **armor** modeling, as the same techniques can be used over again. However, feel free to model the rest of the **armor** on your own or have a look at the provided `Gidiosaurus_modeling_02.blend` file:



The completed armor as it appears in the rendering

Chapter 4. Re-topology of the High Resolution Sculpted Character's Mesh

In this chapter, we will cover the following recipes:

- Using the Grease Pencil tool to plan the edge-loops flow
- Using the Snap tool to re-topologize the mesh
- Using the Shrinkwrap modifier to re-topologize the mesh
- Using the LoopTools add-on to re-topologize the mesh
- Concluding the re-topologized mesh

Introduction

The re-topology of a mesh, as the name itself explains, is simply the reconstruction of that mesh with a different topology; usually, the re-topology is used to obtain a low resolution mesh from a high resolution one.

In our case, this is obviously needed because we are later going to rig and animate our **Gidiosaurus**, and these tasks would be almost impossible with a mesh as dense as the high resolution sculpted one; we not only need to reconstruct the shape of the mesh with a lower number of vertices, but also with the edge-loops properly placed and flowing for the best render and deformation of the character's features.

In Blender, we have several tools to accomplish this task, both hardcoded into the software or as add-ons to be enabled, and in this chapter, we are going to see them.

Using the Grease Pencil tool to plan the edge-loops flow

It would be perfectly possible to start immediately to re-topologize the high resolution mesh, at least for an expert modeler; by the way, it's usually a good practice to have a guide to be followed in the process, to solve a priori any issue (or at least most of them) that we would come across.

So, let's start this chapter by planning what the right topology can be for a low resolution mesh of our **Gidiosaurus** character; we are going to use the **Grease Pencil** tool to draw the paths of the edge-loops and polygons flow, straight onto the sculpted mesh.

Getting ready

First, let's prepare the screen:

1. Open the `Gidiosaurus_modeling_02.blend` file.
2. Go to the **UV/Image Editor** window to the left and *Shift* + left-click on the **X** icon on the toolbar to get rid of the template image (to be more technically precise, to unlink the template image data block; the *Shift* key is to set the users to **0** and definitely eliminate the image from the file).
3. Put the mouse pointer on the border between the two windows and right-click; in the little **Area Options** pop-up panel, left-click on the **Join Area** item and then slightly move the mouse pointer to the left and left-click again, to join the two windows and obtain a single big 3D viewport window:



Joining the two windows into one

4. Click on the **11th** scene layer button to show only the sculpted **Gidiosaurus** mesh and parts such as the **teeth**, **eyes**, and so on.

5. Go to the **Outliner** and click on the icons showing an eye image placed to the right side of the **Eyeballs**, **Fangs**, and **Talons** items to hide them.
6. Press the **N** key to make the **Properties 3D** view sidepanel appear to the right of the 3D window and scroll it to find the **Grease Pencil** subpanel (already enabled by default); go to the **Tool Shelf** panel to the left of the 3D window and click on the **Grease Pencil** tab:



The Grease Pencil panels and the screen layout in current state

7. Check to enable the **Continuous Drawing** item just below the four buttons at the top of the **Grease Pencil** tab on the **Tool Shelf**.
8. Go to the **Grease Pencil** subpanel under the **Properties 3D** view sidepanel to the right and click on the **New** button; then, click on the **+** icon button to the left side to add a new **Grease Pencil** layer, which is by default labeled **GP_Layer**; set the **Stroke color** to **RGB 1.000, 0.000, 0.350** and **Thickness** of the strokes to **4 pixels**.
9. Double-click on the **GP_Layer** name to rename it as **Head**.
10. Go to the **Tool Shelf** and, under **Stroke Placement**, click on the **Surface** button:



Starting to use the Grease Pencil tool

11. Save the file as `Gidiosaurus_retopology.blend`.

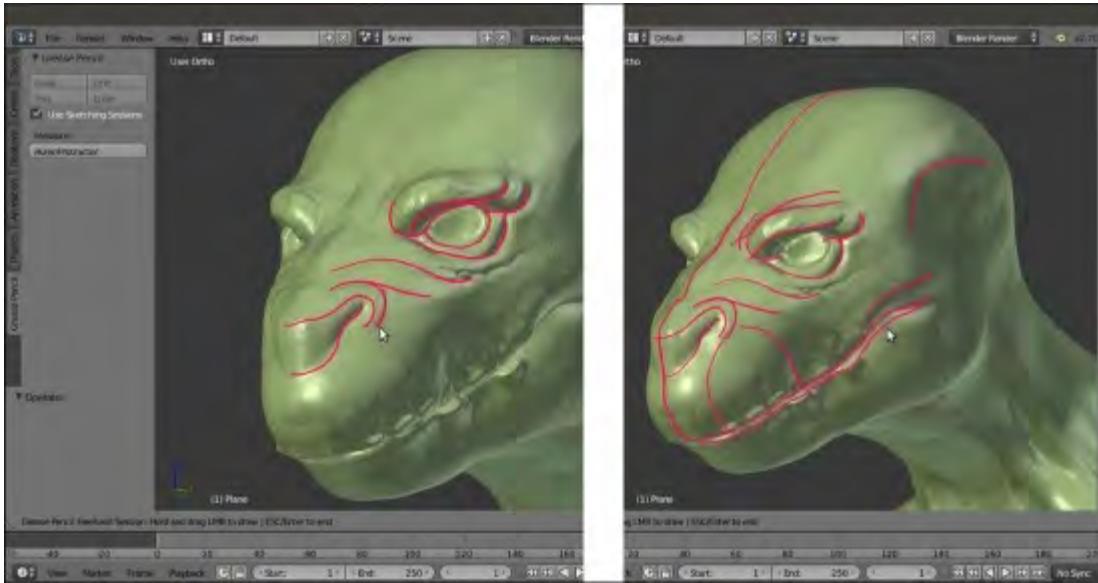
How to do it...

We are now going to start to draw on the character's **head**:

1. Press `Shift + B` and draw a box around the head of the **Gidiosaurus** to zoom to it; then, press the `5` key on the numpad to go into **Ortho** view.
2. Click on the **Draw**, **Line** or **Poly** buttons at the top of the **Grease Pencil** tab in the **Tool Shelf**; alternatively, keep the **D** key pressed (along with left-click) to start to draw the first stroke on the mesh (`Ctrl + D` + left-click and `Ctrl + D` + right-click, respectively for **Line** and **Poly**).

Because we enabled the **Continuous Drawing** item in the **Tool Shelf**, we can continue to draw without the need to reactivate the drawing mode at each stroke. To quit the sketching session (for example, to change the brush), we can press the `Esc` or the `Enter` keys, so *confirming* the sketching session itself at the same time; otherwise, without the **Continuous Drawing** item enabled, the sketching is confirmed right after each stroke.

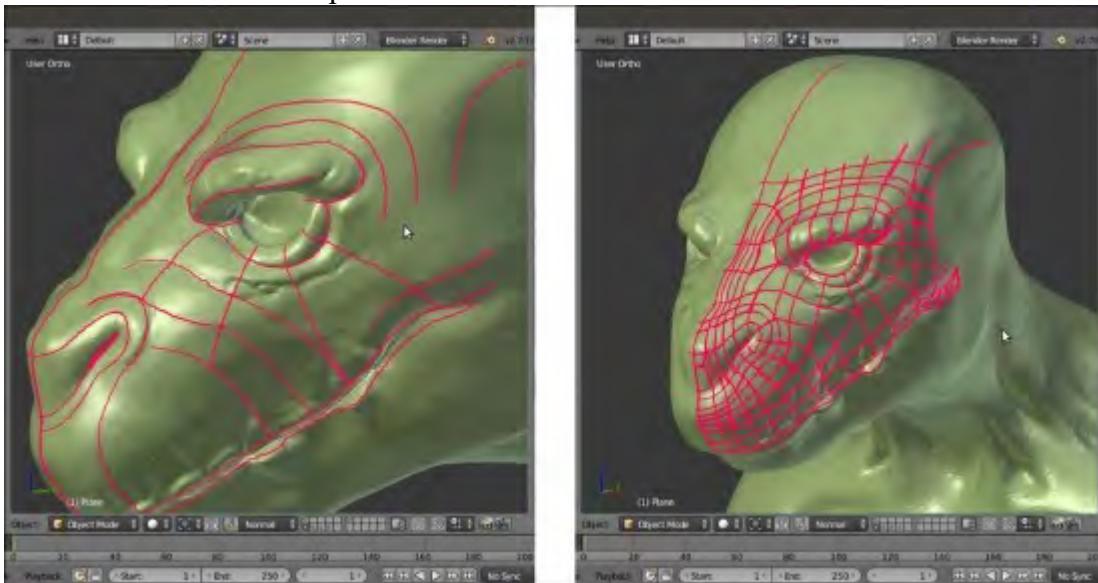
3. Start to draw (one half side of the mesh is enough) the strokes; try to make the strokes follow the main, basic, and more remarkable features of the sculpted mesh such as the main **skin folders** going from the **snout** to the **eye sockets** and the bags under the **eyes**, **nostril**, **mouth rim**, and so on.
4. Don't worry too much about the quality or the precision of the strokes; also, don't be afraid to erase (`D` + right-click or the **Erase** button) and/or correct the strokes, if necessary. The **Grease Pencil**, in this case, is just a tool to sketch directly on the mesh the guidelines we will later follow for the re-topology stage:



Drawing the head's main features topology

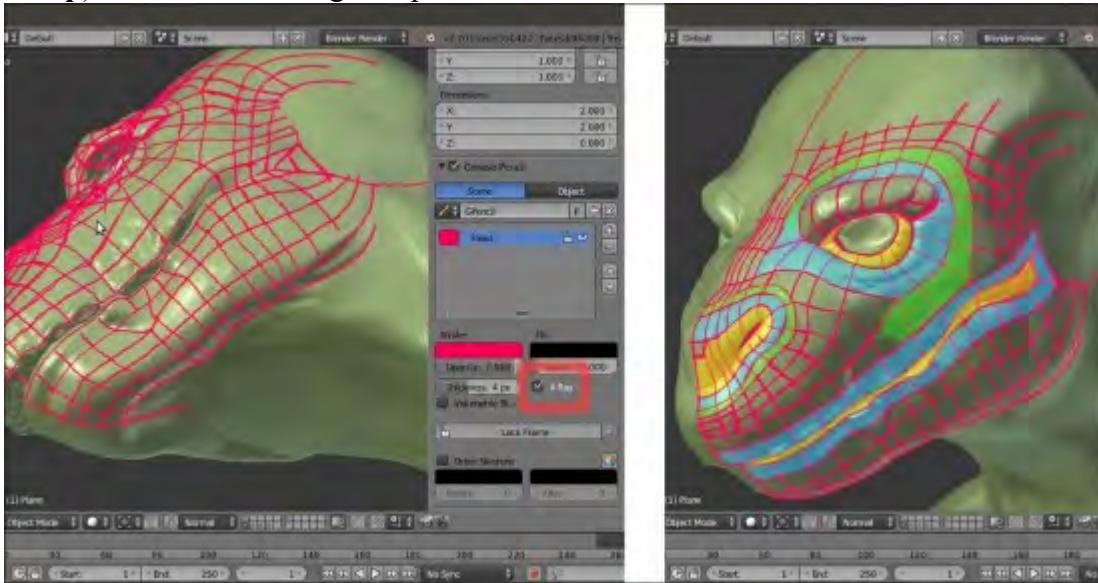
In the case of our **Gidiosaurus**, the topology for a correct deformation is similar to the topology we would use for a human face, but a lot simpler: we just need edge-loops around the **eyes** and in the **eyebrows** area, to give them mobility for expressiveness; a few edge-loops around the **mouth** that, however, in our case, remains quite rigid; and edge-loops following the folders on the top of the **snout**, which can also be important for the *growl* expression.

- Once the strokes for the main features have been posed, try to join them into a web of edges, as balanced and efficient as possible:



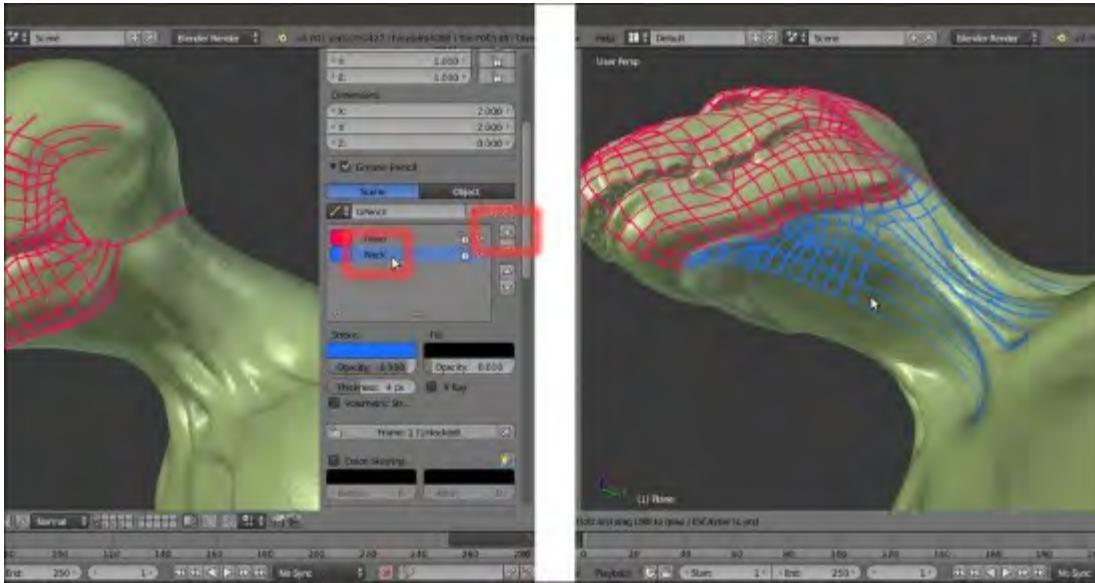
Connecting the strokes

6. At a certain point, when and if the overlapping of the strokes starts to become confusing, you can uncheck the **X Ray** item, which is located to the right side of the **Thickness** slot in the **Grease Pencil** layer subpanel, to disable the visibility of the strokes behind the mesh surface.
7. Forget about the edge-loops of stiff parts such as the cranium; it's enough to plan the position and the flow of the deforming ones. In the screenshot at the bottom right, I have highlighted (in Gimp) the main facial edge-loops for the **Gidiosaurus** with different colors:



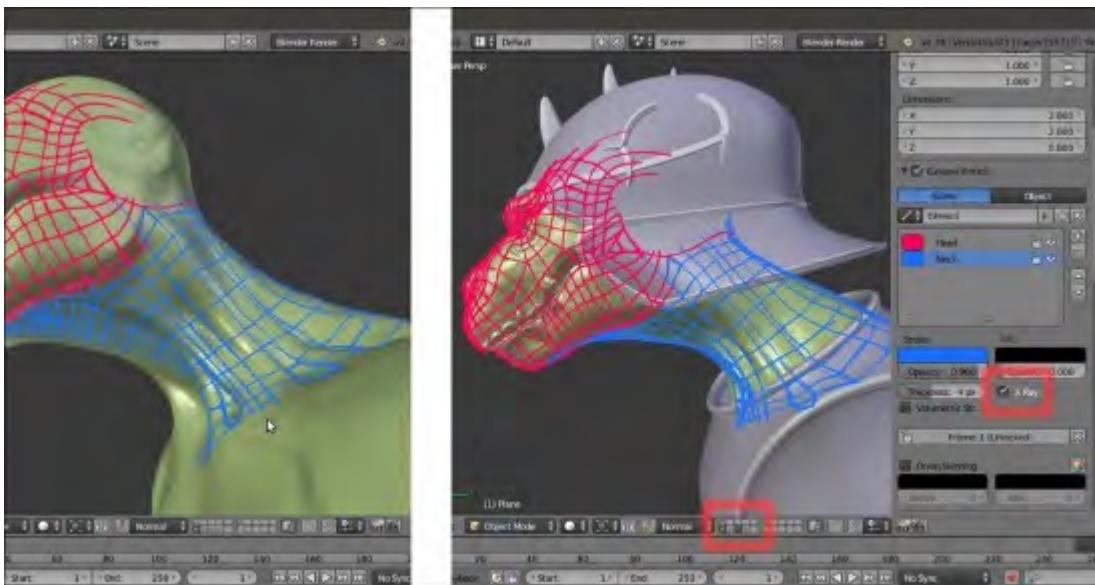
The X Ray button and the highlighted main edge-loops

8. When you think you are done with the **head**, click on the + icon button to add a new layer and rename it as **Neck**. Set the values the same you did for the **Head** layer, just change the color of the strokes; I set mine to **RGB 0.106, 0.435, 0.993**, but whatever color you choose, be sure that it stands out in the viewport against the mesh color.
9. In the case of the **neck**, the important thing is to find the correct joining with the **head's** edge-loops under the lower **jaw**, as you can see in the bottom-right screenshot:



The Neck layer

10. Continue to stroke on the **neck** by drawing parallel horizontal loops along its length and use the vertical strokes to outline the **neck's** muscles (don't look for a **sternocleidomastoid** muscle here; the **Gidiosaurus**, anatomy, although similar in some ways, is not human at all!).
11. Remember that because our character is wearing an **armor**, it is not necessary to re-topologize the whole body, but only the exposed parts; so we can stop the planning just a little beyond the plates outside edges. To verify the correct extension of the strokes, just be sure to have the **X Ray** item enabled in the **Grease Pencil** layers and also the **13th** scene layer enabled to show the **armor**:



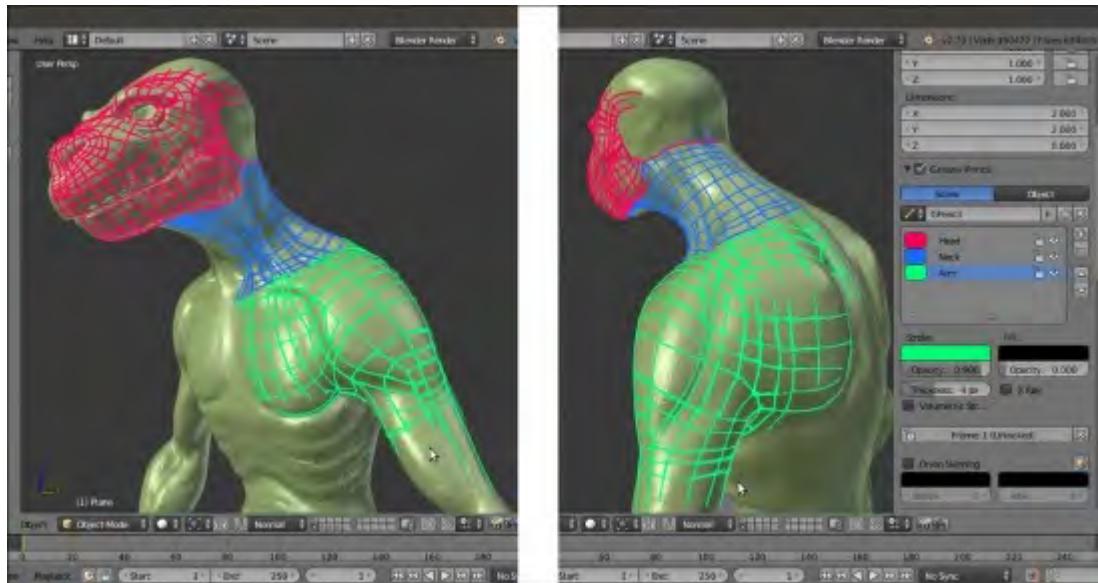
Verifying the extension of the strokes under the armor

12. Click again on the + icon button in the **Grease Pencil** subpanel under the **Properties** 3D view sidepanel and rename the new layer as **Arm**. Set the values the same as you did for the **Head** and **Neck** layers, but change the color once more (**R 0.000, G 1.000, B 0.476**); this time, we have to plan the joining of the cylindrical shape of the **arm** with the **shoulder** and the **collar bones** areas:



Sketching the guidelines on the arm

13. As before, also in this case, it is not necessary to go beyond the boundaries of the **armor chest plate**, but including also the muscles of the **chest** and **back** in the topology planning can give a more natural result:



The completed guidelines for the shoulder and the arm joining

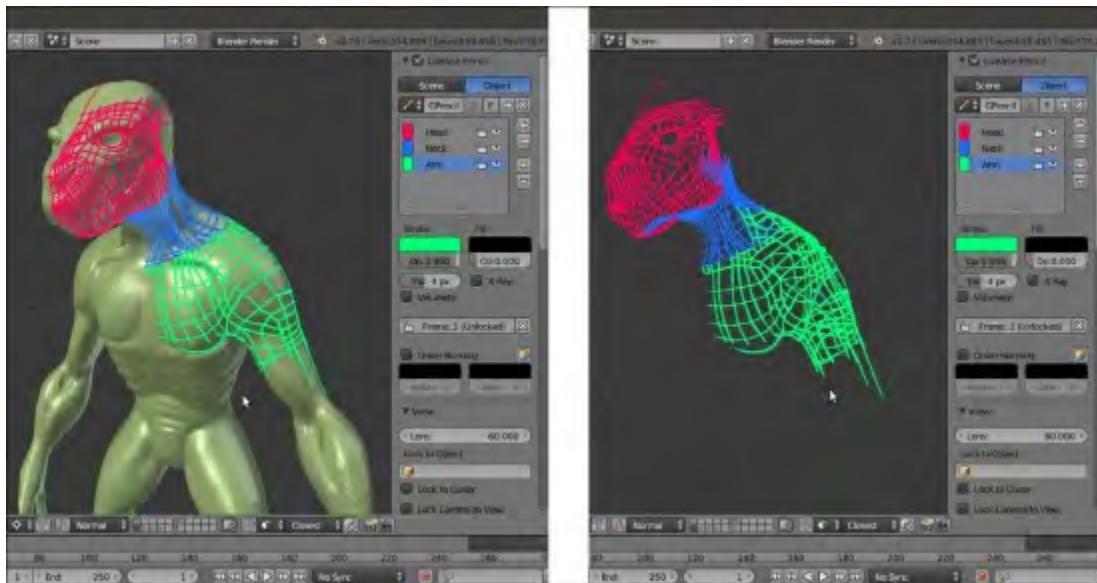
14. When you are done, save the file.

At this point, we can stop with the **Grease Pencil** sketching of the topology; the remaining parts of the exposed body are a lot simpler and will be quickly resolved in the successive recipe of this chapter.

There's more...

We can load any already existing **Grease Pencil** layer data blocks even into an empty scene, by clicking on the little arrows on the left-hand side of the **Gpencil** slot (*Freehand annotation sketchbook*) at the top of the **Grease Pencil** subpanel on the **Properties** 3D view sidepanel, and indifferently for **Scene** or **Object**. Actually, the **Grease Pencil** tool can be used as a sketchbook tool, to write quick notes and/or corrections inside the **Node Editor** window or the **UV/Image Editor** window, and even as an animation tool, by drawing inside an empty scene or on the surface of other objects to be used as *templates*.

In the following screenshot, you can see the sketching sessions previously made on the **Gidiosaurus** object's surface, showing *a solo* and keeping the volumes of the character in the 3D space:



The Grease Pencil layers in the 3D space

See also

- http://www.blender.org/manual/grease_pencil/introduction.html

Using the Snap tool to re-topologize the mesh

In this recipe, we'll use the **Snap** tool to start to re-topologize the sculpted high resolution mesh.

Getting ready

First, let's prepare both, the mesh to be *traced*, which is the high resolution mesh, and the tool itself:

1. Go to the **Outliner** and click on the *Restrict viewport selection* icon, which is the arrow one, to the side of the **Gidiosaurus** item to make it not selectable.
2. Be sure that the **3D Cursor** is at the center of the scene (*Shift + C*) and add a **Plane** primitive.
3. Click on the *Snap during transform* button, the little icon with the magnet, on the 3D view toolbar, or else press *Shift + Tab* to activate the tool.
4. Click on the **Snap Element** button (*Type of element to snap to*) on the close right to select the **Face** item, or else press **Shift + Ctrl + Tab** to make the **Snap Element** pop-up menu appear in order to select the item from:



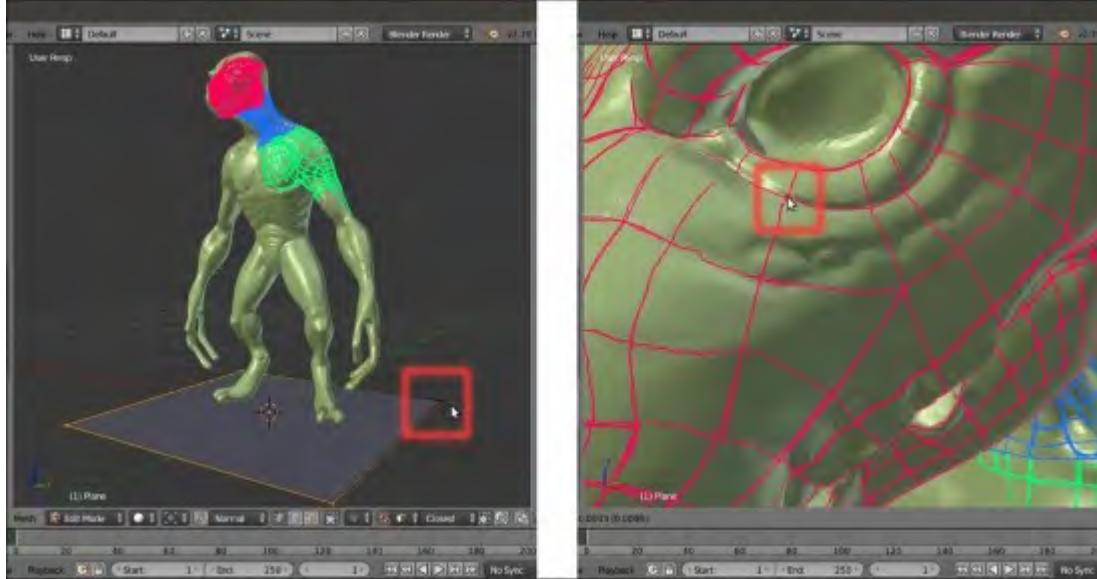
The *Snap Element* menu

How to do it...

Now, we are going to start the re-topology:

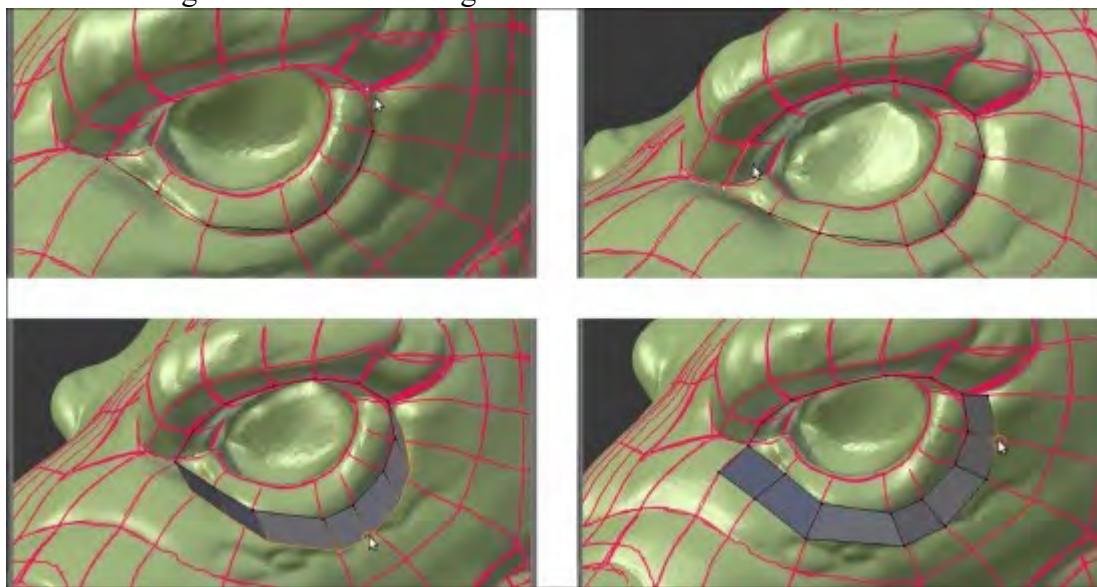
1. With the **Plane** object selected, press *Tab* to go into **Edit Mode**; with all the vertices already selected by default, by pressing *Shift* + right-click, deselect just **one** vertex (anyone of them, it doesn't matter which one).
2. Press **X** to delete the other **three** vertices that are still selected.
3. Select the single remaining vertex and move it onto the **head** of the sculpted mesh, close to the **left eye socket**; as the **Snap** tool is enabled, the vertex stays on the mesh surface.

4. Press the period (.) key on the numpad to zoom the 3D view centered on the selected vertex:



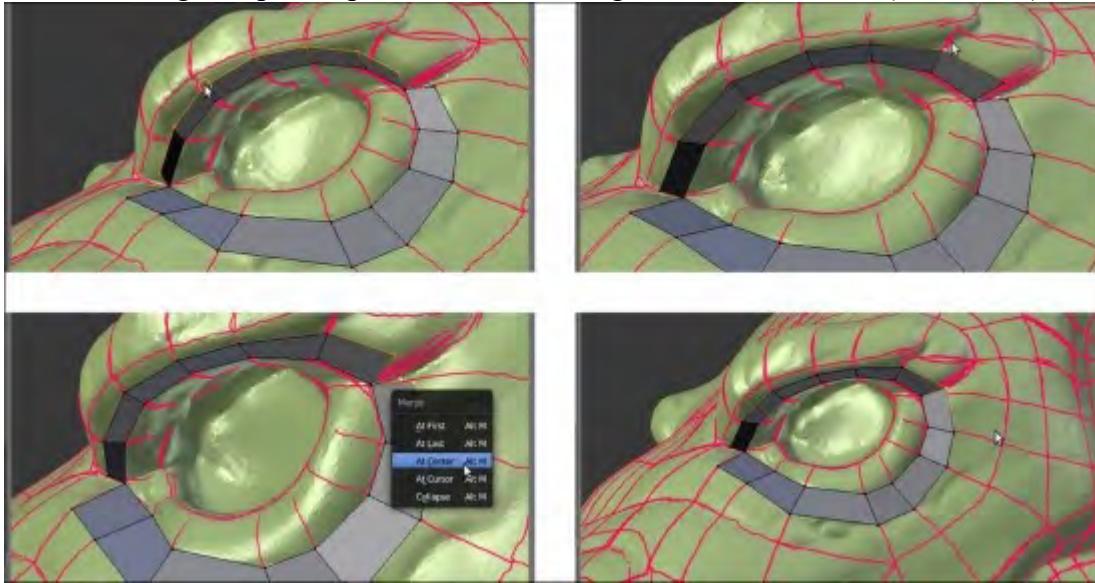
Starting the re-topology process

5. Go to the **Object Data** window and check the **X-Ray** item under the **Display** subpanel in the main **Properties** panel to the right of the screen.
6. Start to extrude the vertex, building an edge-loop around the **eye socket** and following the **Grease Pencil** guideline, both by pressing the *E* key or *Ctrl* + left-click to add vertices; if needed, press *G* to move them at the right location (that is, at the intersections of the guidelines).
7. When you have almost completed the edge-loop around the **eye socket**, select the last and the first vertices and press the *F* key to close it.
8. Select the bottom row of vertices of the edge-loop and extrude them; adjust the position of each vertex on the ground of the strokes guideline:



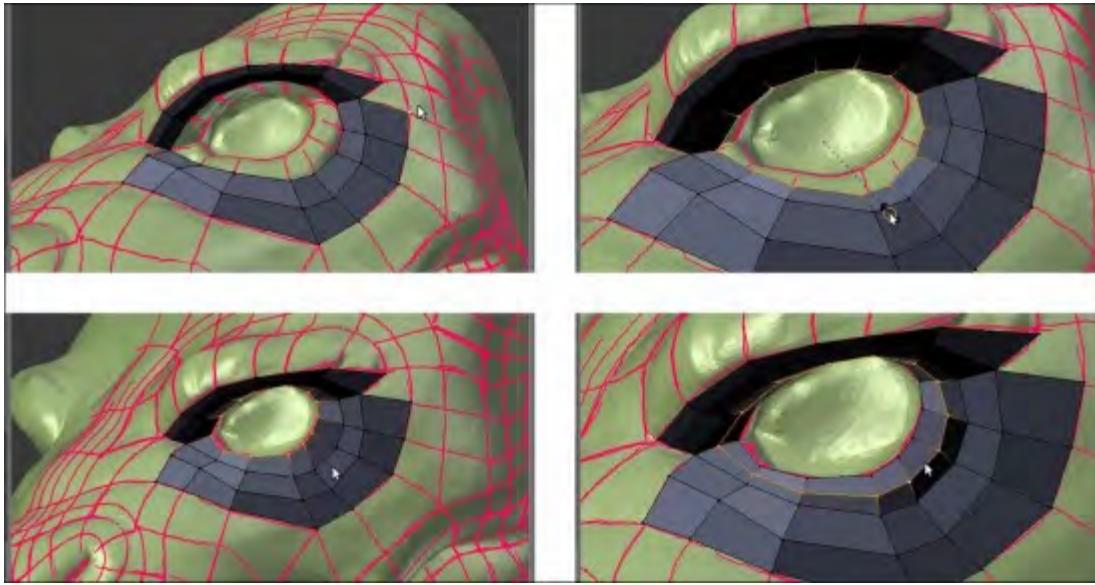
The first re-topology around the eye socket

9. Do the same with the upper row of vertices and then select the free vertices on the right-hand side of the edge-loops and press **Alt + M** to merge them at the center (**At Center**):



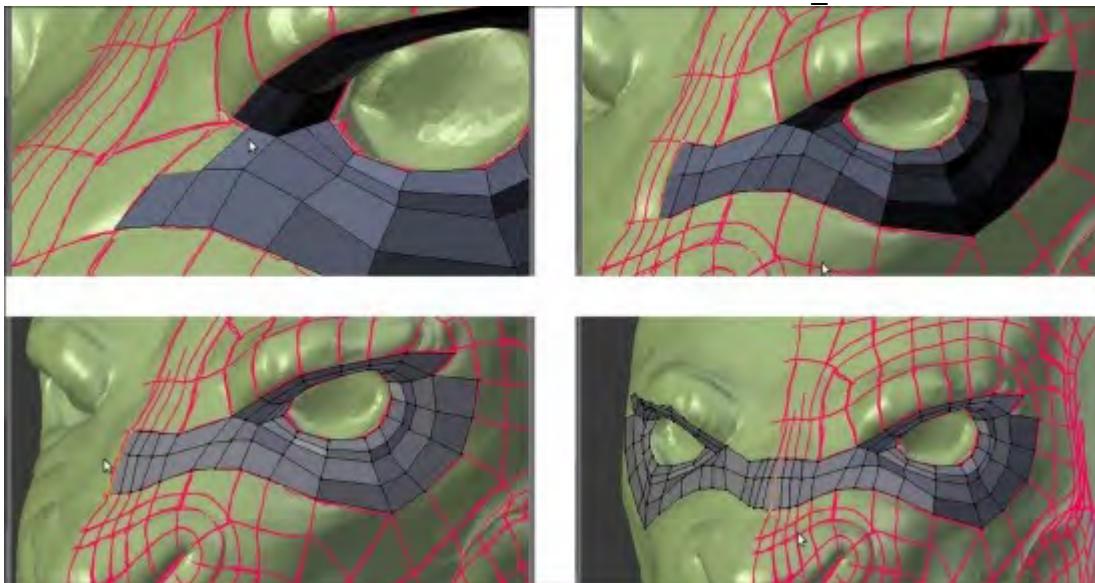
Building the eye edge-loop

10. While still in **Edit Mode**, select all the vertices and press **Ctrl + N** to recalculate the normals.
11. Keep on extruding the edge-loops to build the faces around the **eye socket**. Select the inner edge-loop and extrude it; then, scale it inside and adjust the vertices position as usual.
12. Cut a new edge-loop in the middle of the **eye socket** by pressing **Ctrl + R** and then select each vertex; press **G** and, immediately after, click with the left button of the mouse. This way the newly added vertex stays in place, but is snapped to the underlying surface (sadly, it doesn't work automatically as you cut or add vertices; they must be moved in some way to make the **Snap** tool work).



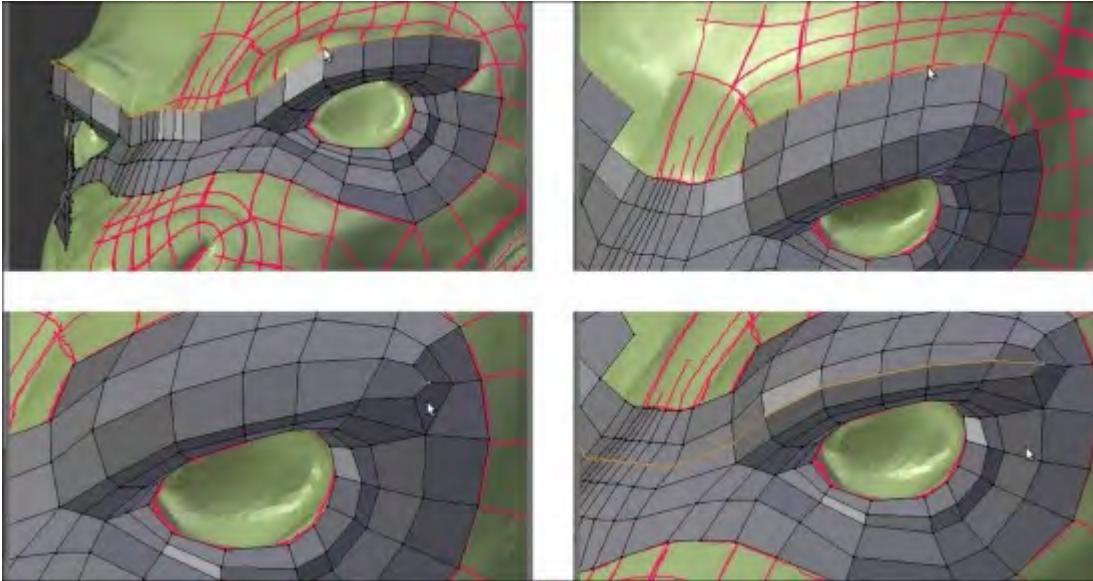
Adding geometry and snapping the vertices to the surface

13. Keep on adding geometry to the mesh, extruding or *Ctrl + left-clicking*, and switch between edges and vertices selection mode to make the workflow faster. Press 5 on the numpad to go into **Ortho** view when necessary. Following the strokes guideline, build faces going towards the median line of the object.
14. As you are arrived to the median line of the object, go to the **Object Modifiers** window under the **Properties** panel and assign a **Mirror** modifier.
15. Click on the *Adjust edit cage to modifier result* icon (the last one in the row to the side of the modifier's name), to activate the modifier during the editing, and check the **Clipping** item.
16. Adjust the vertices you just added to the median line of the mesh to stay on the y axis and recalculate the normals.
17. Go to the **Outliner** and rename the **Plane** item as **Gidiosaurus_lowres**.



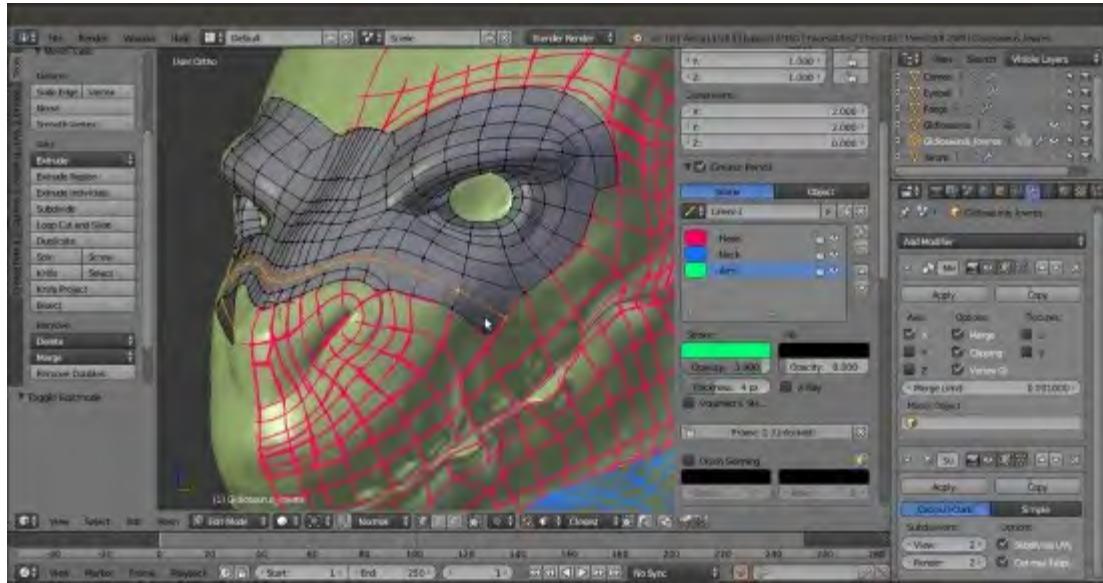
Going towards the median line

18. Build the remainder of the faces the same way, extruding edges or vertices, moving them to react to the **Snap** tool, and adding cuts and edge-loops where needed to keep all quads. **N-gons** faces can be split into quads by dividing an edge to add a vertex in the middle, selecting the new vertex and its opposite one and pressing the **J** key to connect them (see the two screenshots at the bottom row):



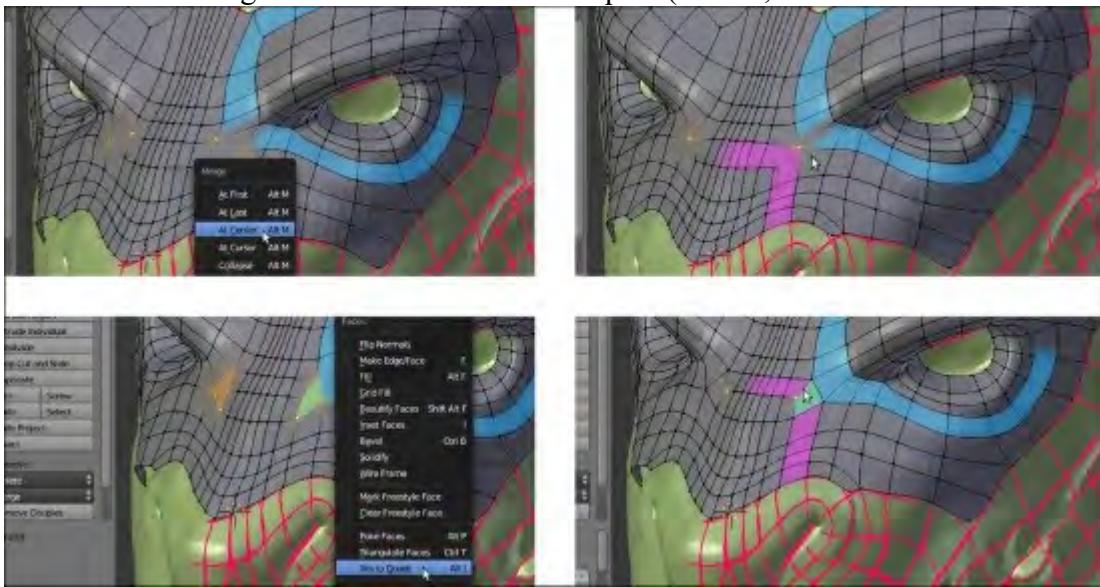
Building the eyebrows and dividing N-gons into two quad faces

19. Assign a **Subdivision Surface** modifier to the low resolution mesh and set **Subdivisions** to **2**. Check the **Optimal Display** item; if you want, click on the *Adjust edit cage to modifier result* icon, which is the last one in the row to the side of the modifier's name. To work with an already smoothed mesh (in the end, the mesh will be subdivided in any case) is a usual workflow; by the way, it depends on your preferences. If you prefer to work without the modifier, occasionally go out of **Edit Mode** to verify how the geometry behaves under the **Subdivision Surface** modifier.



The created geometry in the Subdivision Surface visualization mode

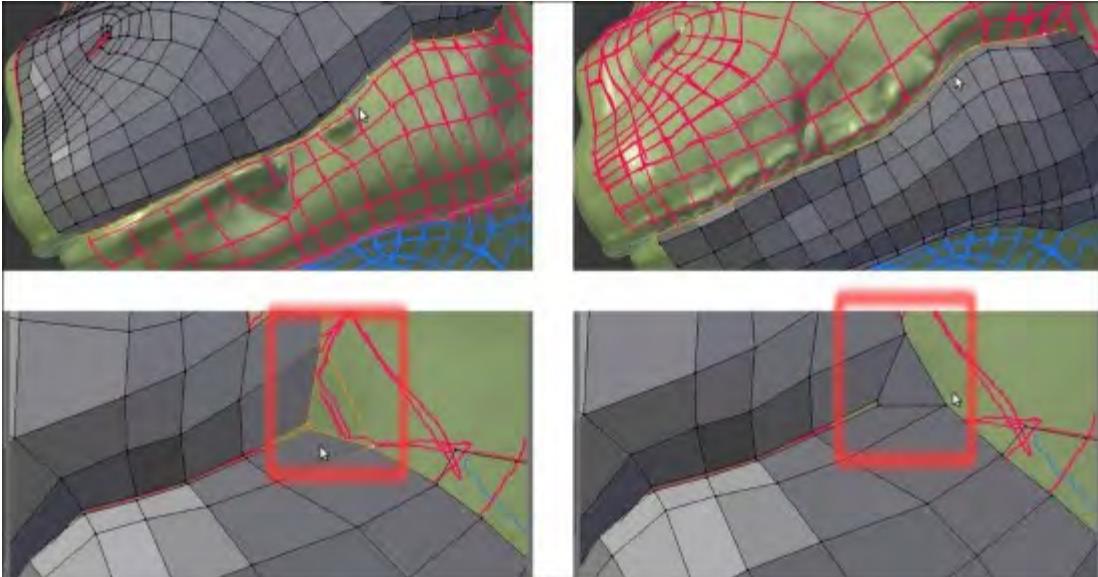
20. Around the eyebrows, it is important to have continuous edge-loops to allow for better mesh deformation; often, it is enough to merge (*Alt + M*) two vertices to obtain the right flow. Note that this creates a pole (check out the screenshot at the top right) that can later be eliminated by a cut and then merges the two tris faces into a quad (*Alt + J*; the two screenshots at the bottom):



Closing two edge-loops

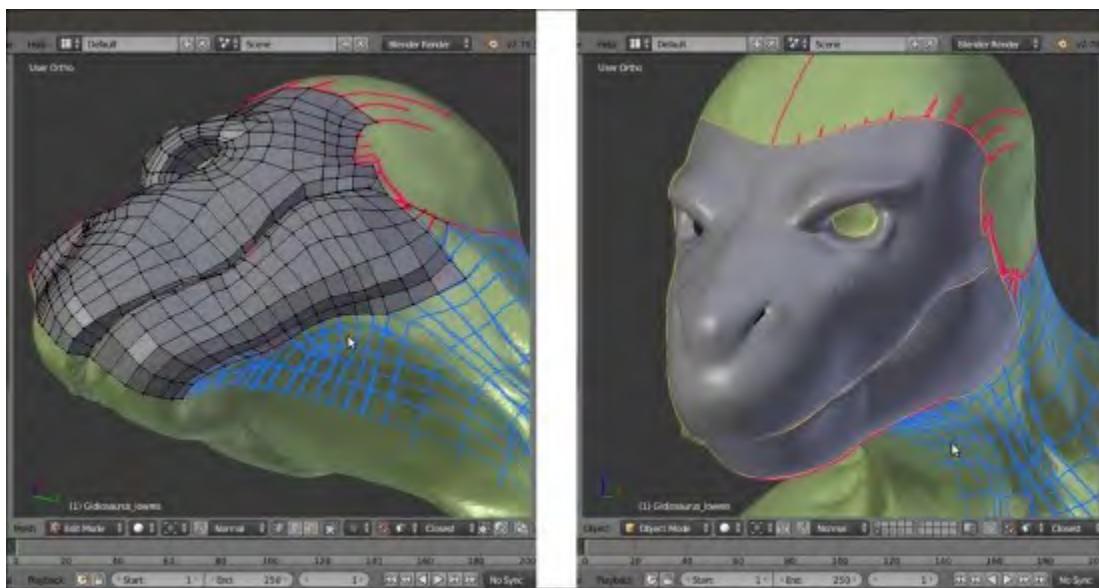
21. We have almost completed the **Gidiosaurus'** face. Select the vertices of the lower **jaw** and press the **H** key to hide them; select the **upper mouth rim** and extrude it and then adjust the vertices' position.

22. Deselect the vertices, press *Alt + H* to unhide the **mandible**, and press *Shift + H* to hide the unselected vertices (in this case, the **upper face**). Select the **mouth rim** of the **mandible** and extrude and then tweak the position of the vertices.
23. Connect the upper and the lower **jaws** by connecting the last vertices, as shown in the bottom-left of screenshot and then build a face. Tweak the vertices' position:



Connecting the jaw to the upper mouth

We can stop using the **Snap** tool at this point and continue with the re-topologizing by using different tools; we'll see this in the upcoming recipes.



The re-topologized face of the character

How it works...

The main requirement for a re-topology tool is the ability to trace the shape and volume of the high resolution mesh as easily as possible. In this recipe, we used the Blender **Snap** tool that, once set to **Face**, guarantees that every added vertex lies on the faces of any directly underlying object; this way, it is quite simple to concentrate on the flow of the polygons, while their vertices stay anchored to the mesh's surface.

To remark that the strokes are there only as a generic indication, note that in certain areas we are doubling the number of faces sketched with the **Grease Pencil** tool as well as to try to keep the density of the mesh as even as possible.

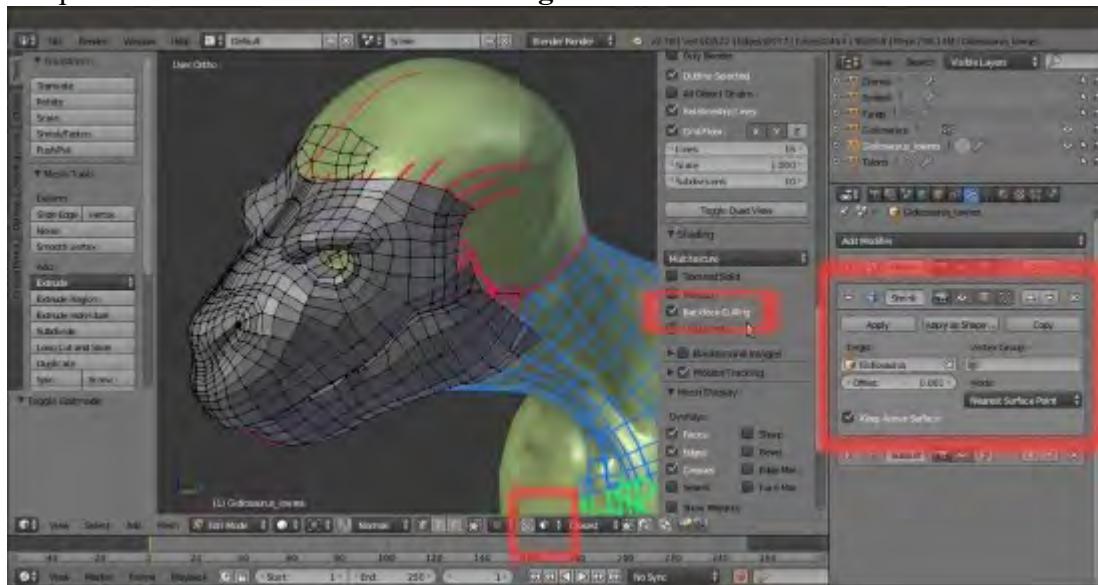
Using the Shrinkwrap modifier to re-topologize the mesh

Sometimes, the **Snap** tool is not enough or can be quite difficult to use because of a particular shape of the high resolution mesh; in these cases, the **Shrinkwrap** modifier can be very handy.

Getting ready

Basically, the usage of this method is all in the preparation of the modifier:

1. Assign the **Shrinkwrap** modifier to the **Gidiosaurus_lowres** mesh and, in the modifier stack, move it *before* the **Subdivision Surface** modifier.
2. Click on the **Target** field to select the **Gidiosaurus** mesh item and leave the **Mode** option to **Nearest Surface Point** (this seems to be the more efficient mode for this task; by the way, you can experiment with the other two modes that can reveal themselves useful in other situations).
3. Enable the *Display modifier in Edit mode* and *Adjust edit cage to modifier result* buttons (the penultimate one and the last one to the right, with the cube and four selected vertices image and with the upside-down triangle and three vertices image, respectively) and the **Keep Above Surface** item.
4. In **Edit Mode**, if it's necessary to make the low resolution mesh more easily visible against the high resolution one, change the **Offset** value to **0.001**.
5. Having the **X-Ray** item still active, go to the **Shading** subpanel under the **Properties 3D view** sidepanel and check the **Backface Culling** item:

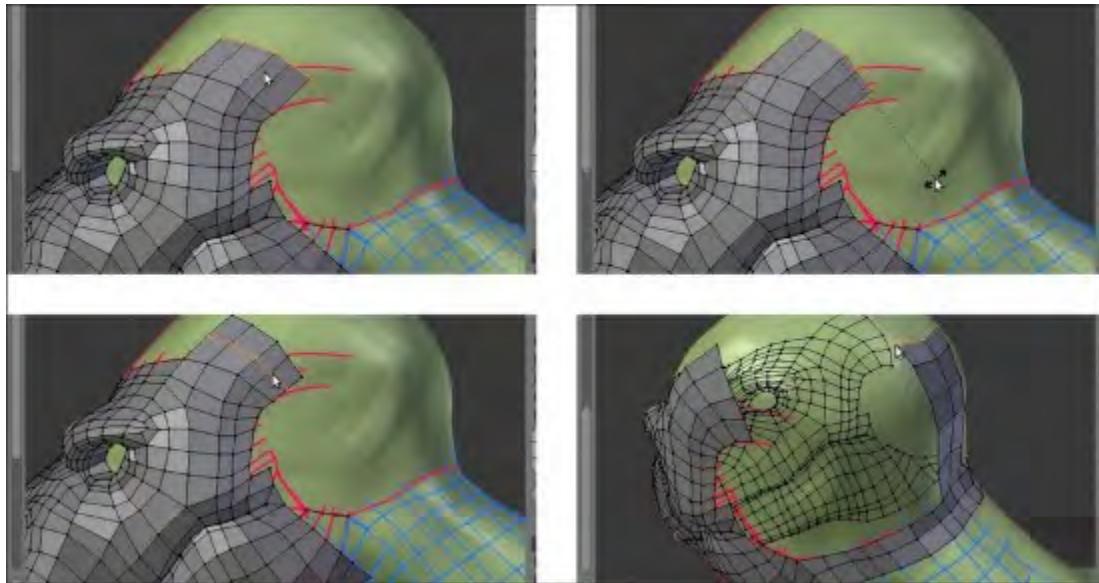


The **Shrinkwrap** modifier panel

How to do it...

In **Edit Mode**, select, extrude, and move the vertices as required! The **Shrinkwrap** modifier will take care of keeping the vertices adhering to the target mesh surface.

If you are having issues, such as vertices jumping everywhere as you try to move them, try to disable the **Snap** tool. This is not always the case, but sometimes the combination of both the tool and the modifier can give unexpected results; other times, it can be the opposite.



Extruding and cutting an edge-loop under the Shrinkwrap modifier

Remember that if you are using this method to re-topologize, at the end of the process, you *must apply the Shrinkwrap modifier*.

Also, save the file.

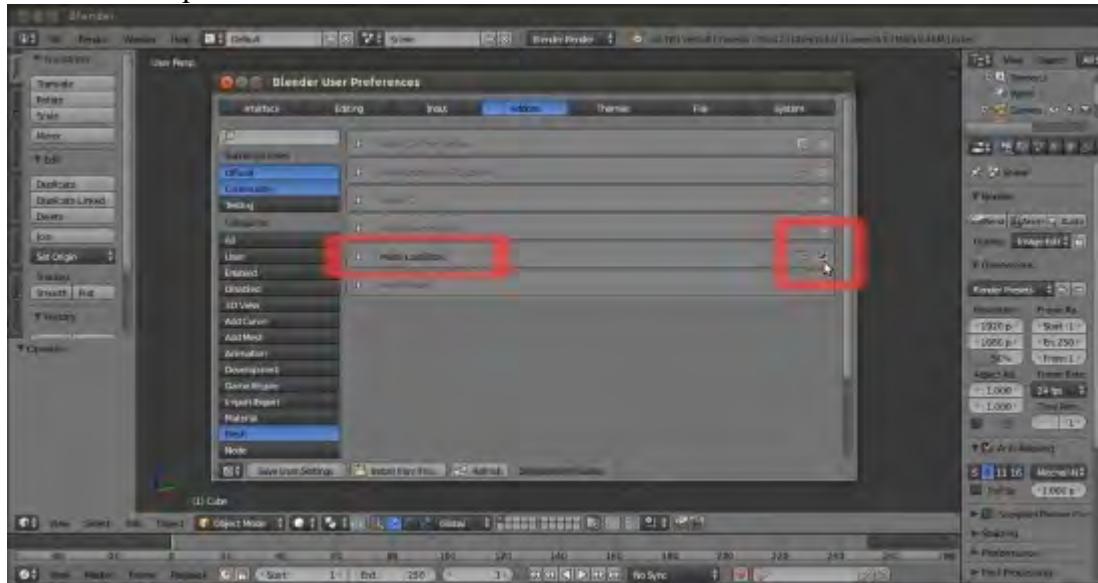
Using the LoopTools add-on to re-topologize the mesh

We have already seen the **LoopTools** add-on in [Chapter 3, Polygonal Modeling of the Character's Accessories](#). This incredibly useful Python script can even be used for the re-topology!

Getting ready

If the **LoopTools** add-on isn't enabled yet, perform the following steps:

1. Start Blender and call the **Blender User Preferences** panel (*Ctrl + Alt + U*); go to the **Addons** tab.
2. Under the **Categories** item on the left-hand side of the panel, click on **MESH**.
3. Check the empty little box to the right of the **MESH: LoopTools** add-on to enable it.
4. Click on the **Save User Settings** button at the bottom-left of the panel to save your preferences and close the panel:



The User Preferences panel and the LoopTools add-on enabled

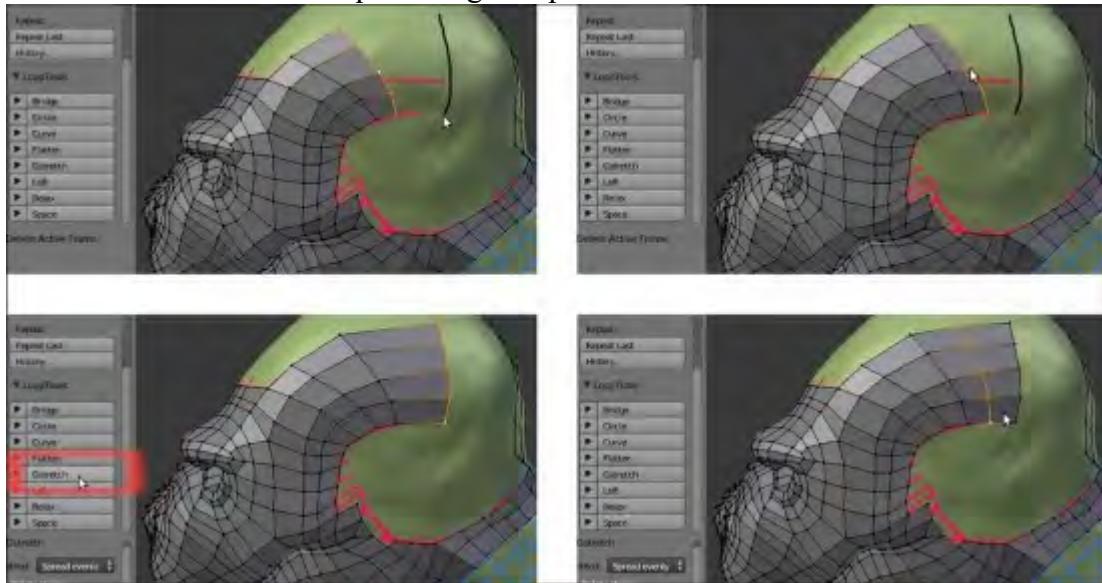
5. Load the `Gidiosaurus_retopology.blend` file.
6. Click on the *Snap during transform* button on the 3D view toolbar (or else, press *Shift + Tab*) to enable the **Snap** tool again.

How to do it...

In the **LoopTools** add-on, there are at least three tools that can be used for the re-topology: **Gstretch**, **Bridge**, and **Loft** (the last two seem to have almost the same effect so, at least for our present goal, we can consider them to be interchangeable).

Let's first see the **Gstretch** tool:

1. Go to the **Grease Pencil** subpanel under the **Properties** 3D view sidepanel to the right. Be sure that the **Grease Pencil** checkbox is checked and click on the + icon button to add a fourth layer after the **Arm** layer (actually, you can also delete the preexisting **GPencil** data block and start with a brand new one, or in any case disable the visibility of the other layers); leave the strokes color as it is by default—that is, pure black.
2. In **Edit Mode**, press **D** and sketch one edge-loop stroke.
3. Select the edges of the low resolution mesh and press **E** to extrude them and then right-click; click on the **Gstretch** button (or press **W** | **Specials** | **LoopTools** | **Gstretch**).
4. In the last operator panel at the bottom of the **Tool Shelf** (or else, press **F6** to make the pop-up window appear at the mouse cursor location), check the **Delete strokes** item.
5. Press **Ctrl + R** to cut the required edge-loops in the new faces:

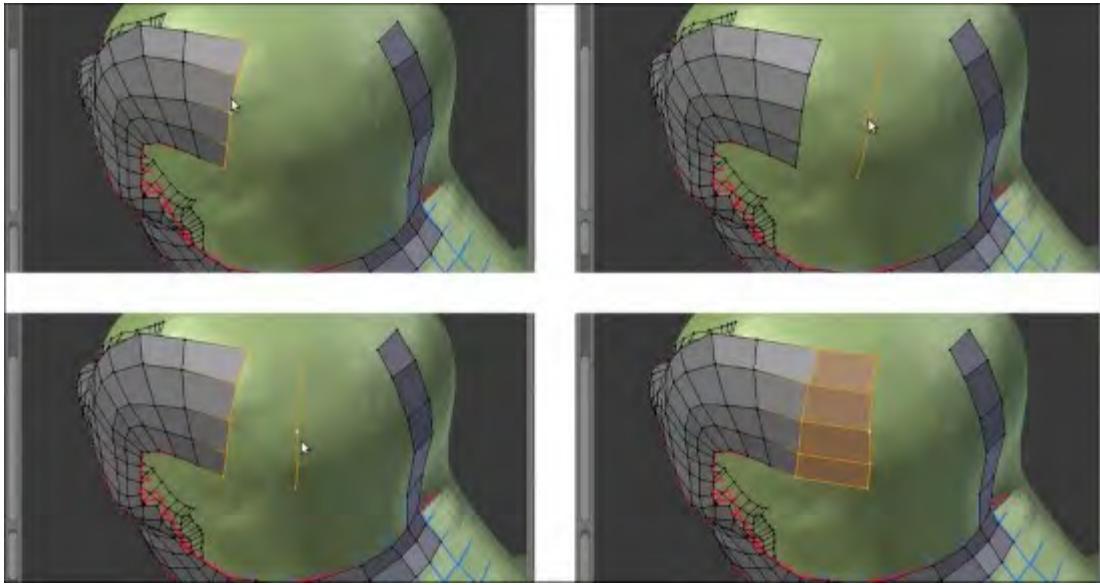


*Using the **Gstretch** tool in conjunction with the **Grease Pencil** tool*

Yes, it's that simple; it's enough to stroke the target position line and the new extruded vertices will be moved to that target position.

Also, now let's see the **Bridge** and the **Loft** tools:

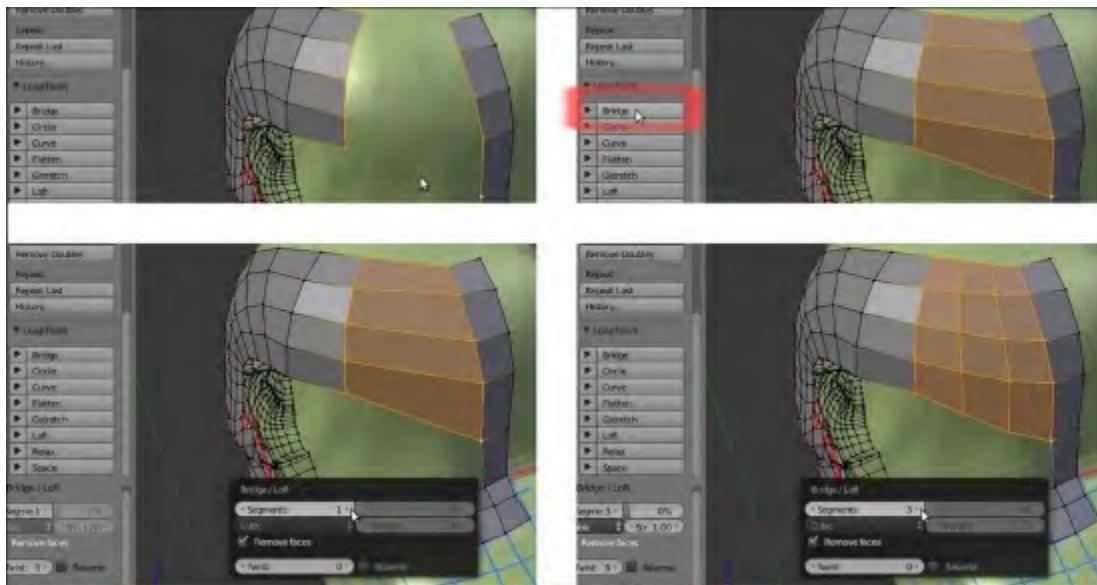
1. Select a group of edges and press **Shift + D** to duplicate them.
2. Move them into a new position and adjust the vertices as required.
3. Select both the new edges as the previous group.
4. Go to the **LoopTools** panel and click on the **Bridge** button (or again, through the **W** key to call the **Specials** menu).



Using the Bridge tool

5. If you need to add cuts, instead of the usual *Ctrl + R* shortcut, go to the last operator panel (*F6*) and change the value of the **Segments** slot to the number required.

It's not mandatory to duplicate new edges, it's enough to select the same number of vertices in the two edge-loops to be connected; here, after the **Bridge** tool operation, we have set the **Segments** value to **3**:

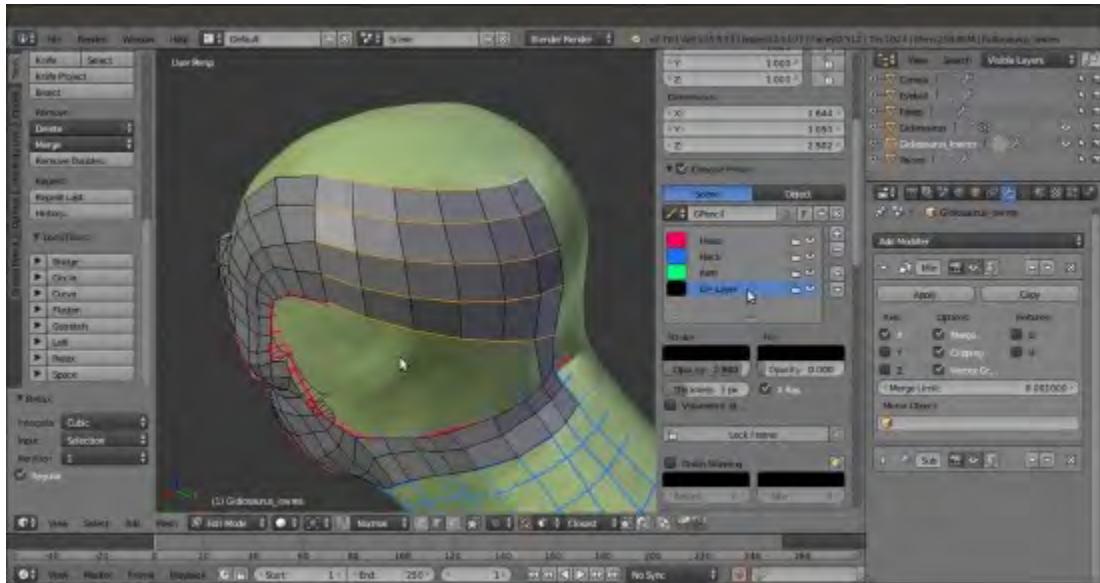


Adding cuts to the bridge operation

You can repeat the operation and the add-on will keep the last values you entered.

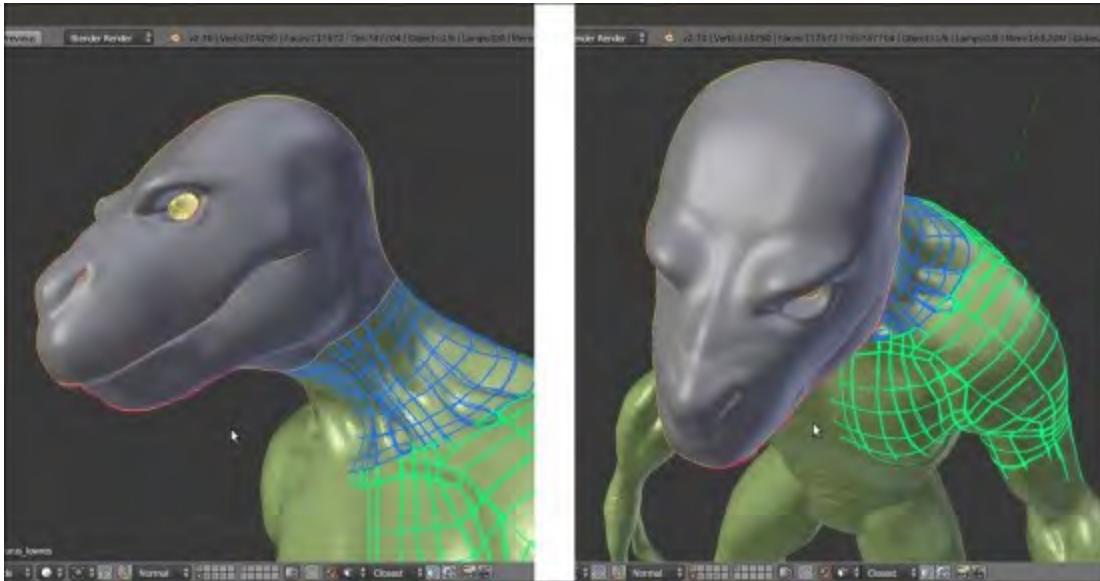
Repeat the steps, and this time click on the **Loft** button. The effect is almost the same, but if the new faces come out really messy, just click twice on the **Reverse** checkbox in the last operator panel; this should fix the issue.

You can then use all the other buttons to refine the added geometry; in the following screenshots, I tweaked the new geometry a little bit by selecting the horizontal edges and clicking on the **Space**, **Flatten**, and **Relax** buttons:



Completing the Gidiosaurus head

Using a mix of all the previous methods, in a short time, we have completed the head and the joining of the neck of our **Gidiosaurus_lowres** mesh; as you can see, particularly in the second screenshot at the bottom, the technique of following the main features and folders of the sculpted surface with the edge-loops can highlight the organic shapes even with a low resolution mesh:



The completed head

Don't forget to save the file and quit Blender.

Concluding the re-topologized mesh

The **Shrinkwrap** modifier method can be the way to quickly finish the rest of the re-topology of the **Gidiosaurus** sculpted mesh, by quickly re-topologizing the simpler cylindrical shapes and then completing the more difficult parts by hand.

Getting ready

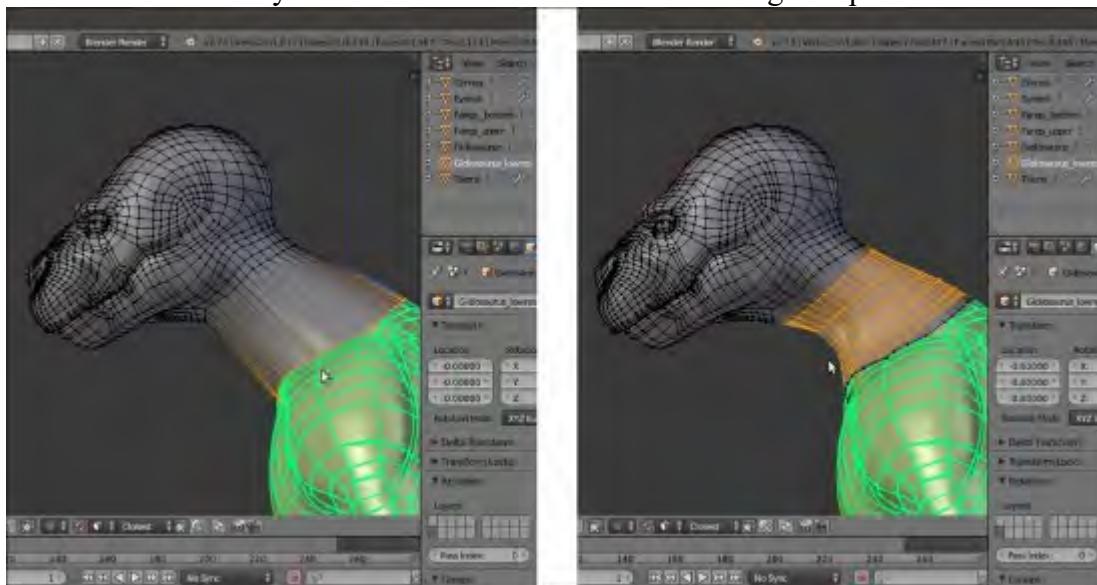
If necessary, repeat the steps to set up the **Shrinkwrap** modifier technique:

1. Assign the **Shrinkwrap** modifier to the **Gidiosaurus_lowres** mesh and in the modifier stack, move it *before* the **Subdivision Surface** modifier.
2. Click on the **Target** field to select the **Gidiosaurus** mesh item and leave **Mode** to **Nearest Surface Point**.
3. Enable the *Display modifier in Edit mode* and *Adjust edit cage to modifier result* buttons and the **Keep Above Surface** item.
4. In **Edit Mode**, to make the low resolution mesh more easily visible against the high resolution one, change the **Offset** value to **0.001**.
5. Having the **X-Ray** item still active, go to the **Shading** subpanel under the **Properties** 3D view sidepanel and check the **Backface Culling** item.
6. Then, to conclude the re-topology, we also need to enable the **Copy Attributes Menu** add-on; go to **Blender User Preferences | Addons | 3D View | 3D View: Copy Attributes Menu**.

How to do it...

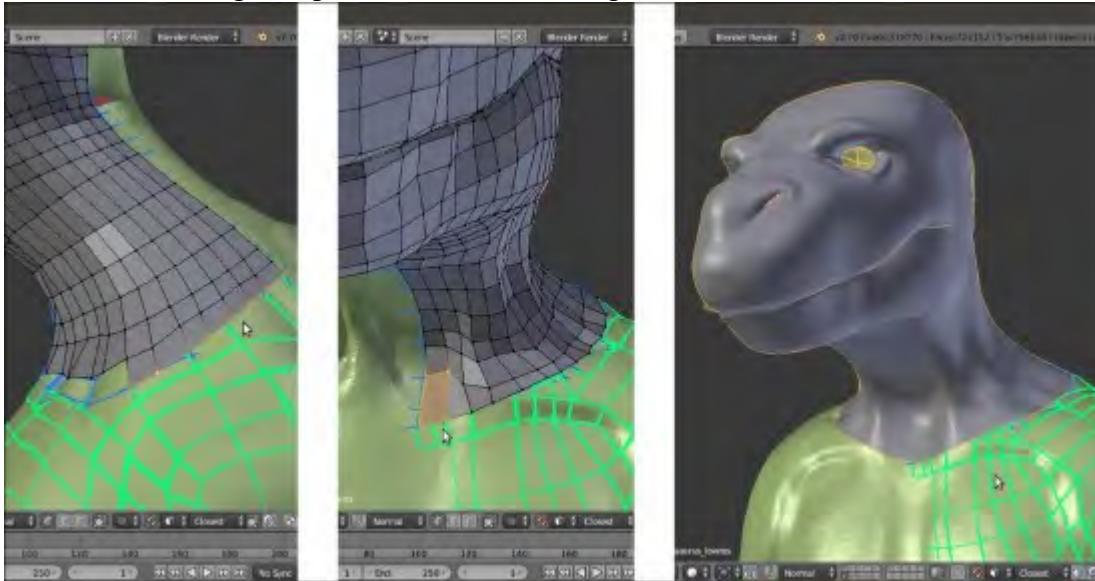
Let's go on by building the geometry of the **neck**:

1. While still in **Edit Mode**, just select the **head's** last edge loop on the **neck** and extrude it (**E** key) towards the **shoulders**.
2. Press the **Ctrl + R** keys and add at least **7** or **8** widthwise edge-loops:



Extruding the neck

- Also, with the aid of the **Snap** tool, tweak the position of the bottom row of vertices, extrude them to add an edge-loop of faces, and tweak again. Go out of **Edit Mode**.



The re-topology of the neck

We can use the same technique as in steps 1, 2, and 3 to quickly re-topologize the left **arm** and **leg** of the character. Instead of extruding the new geometry from the **Gidiosaurus_lowres** mesh, in this case, it's better to add a new simple primitive: a **Circle** or also a **Plane**; whatever the primitive, when you add it, be sure that the **3D Cursor** is at the character's origin point.

As you can see in the following screenshot, at first we just created the geometry only for the *main* cylindrical sections of the **limbs**:



Arms and legs re-topologized

Do the same for the **body**: just a couple of edge-loops placed at the **waist** to extrude the geometry from; remember that the **chest** is covered with the **armor breastplate**, so only the **exposed area** needs to be re-topologized.

One **Mirror** and one **Subdivision Surface** modifier has been assigned to the **three** objects (**head/neck**, **arm**, and **hips/leg**). Also, because of the **Mirror** modifier, the vertices of the half side of the **abdomen's** edge-loops have been deleted.

There's more...

After the main parts have been re-topologized, we can start to tweak the position of the vertices on the **arm** and **leg**, to better fit the flow and shapes of the muscles and tendons in the sculpted mesh.

Thanks to the aid of the **Shrinkwrap** modifier, we can do it quite freely; however, before we start with the tweaking, we require a little bit of preparation for a better visibility of the working objects, to affect and modify the new geometry (visible as a wireframe) and have the underlying sculpted mesh visible at the same time.

To do this, we have two ways:

The first way is as follows:

1. Go to the **Shrinkwrap** modifier panel and set the **Offset** value to **0.002**.
 2. Go to the **Object** window and disable the **X-Ray** item; in the **Maximum Draw Type** slot, under the **Display** subpanel, select **Wire**:



The mesh visualized in wireframe mode

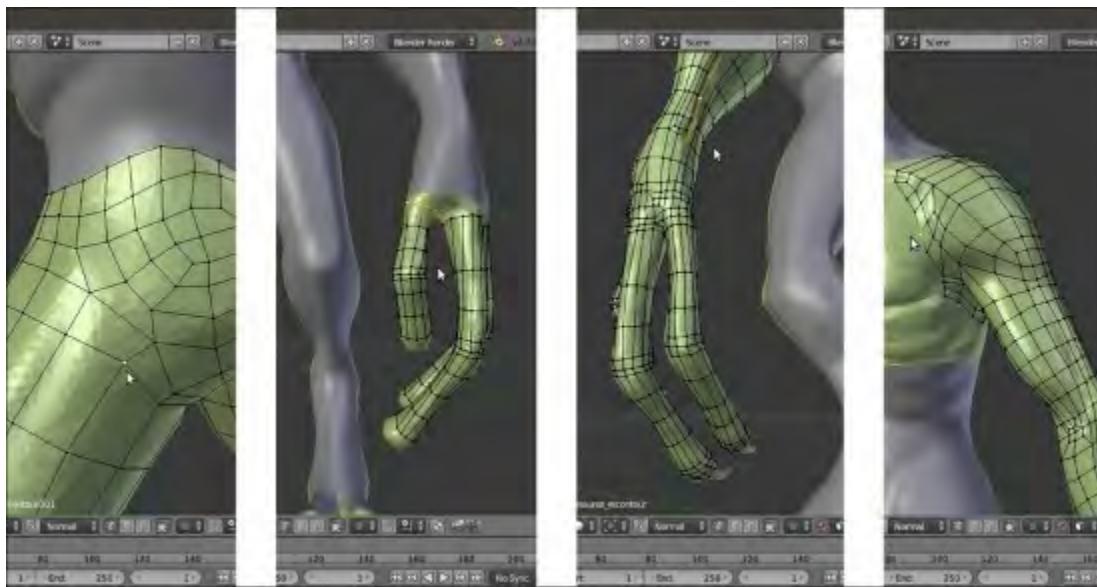
The second way is as follows:

1. Go to the **Shrinkwrap** modifier and set the **Offset** value back to **0.000**.
2. If this is the case, go to the **Object** window and, under the **Display** subpanel, enable the **X-Ray** item. In the **Maximum Draw Type** slot, under the **Display** subpanel, select **Textured**.
3. Go to the **Properties** 3D view sidepanel (press **N** if not already present); if necessary, enter **Edit Mode** and under the **Shading** subpanel, check the **Hidden Wire** item.
4. In both ways (I used the second one), if you want to enable the *Display modifier in Edit mode* and *Adjust edit cage to modifier result* buttons for the **Subdivision Surface** modifier to see its effect in **Edit Mode**, it is better to move the **Shrinkwrap** modifier *after* the **Subdivision Surface** modifier in the stack, to have a better looking result.



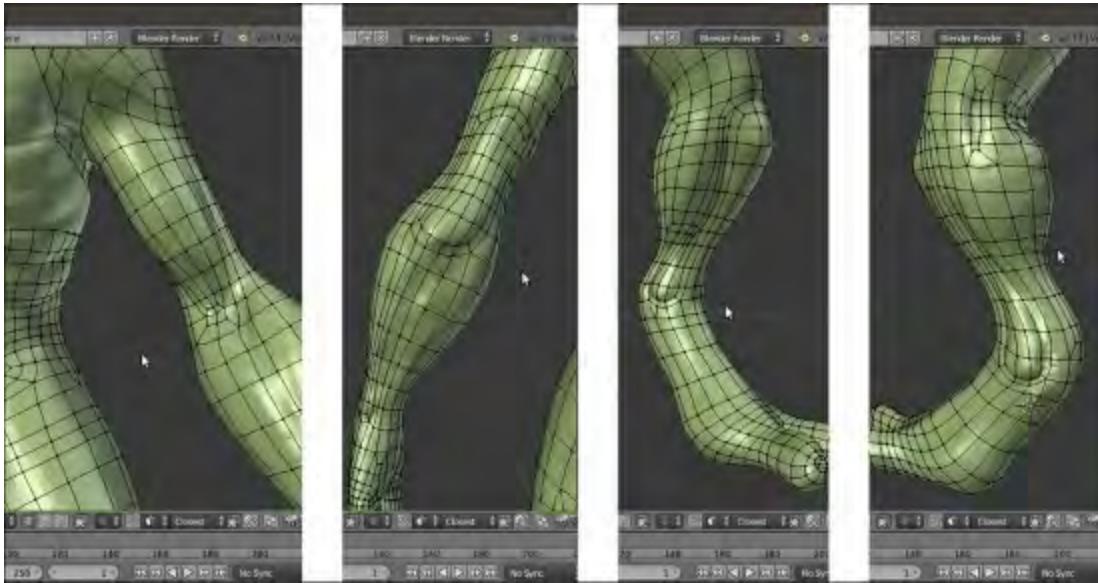
The second wireframe visualization method

We can now start to add the missing parts, by extruding and moving the vertices to better fit the sculpted features and also adding, if necessary, new edge-loops to better define these features:



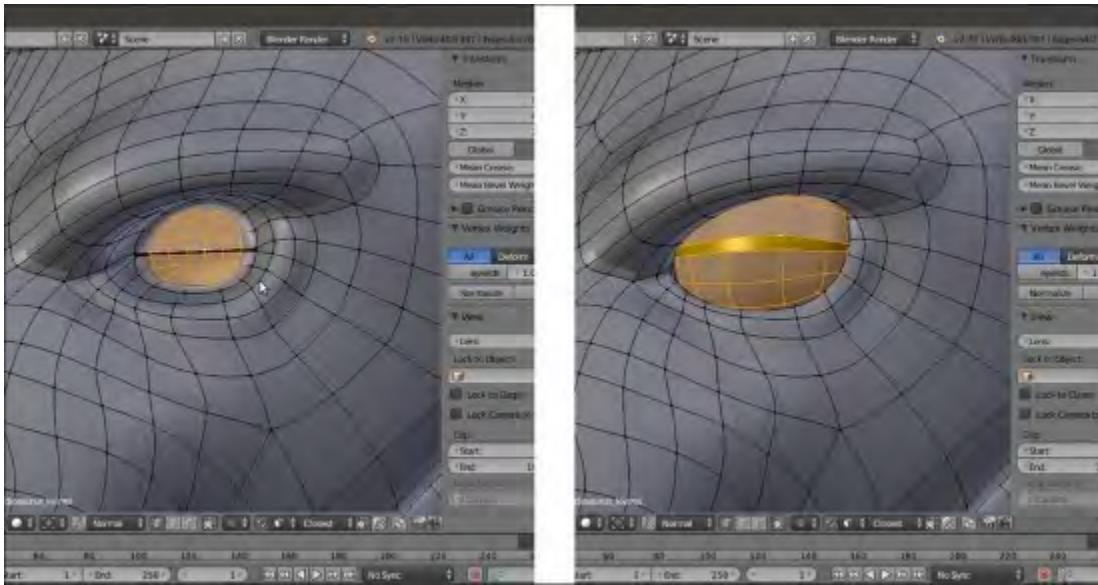
Refining and completing the remaining features

After the **wireframe** setup, it's easy to tweak the low resolution geometry to better fit the character's anatomy:



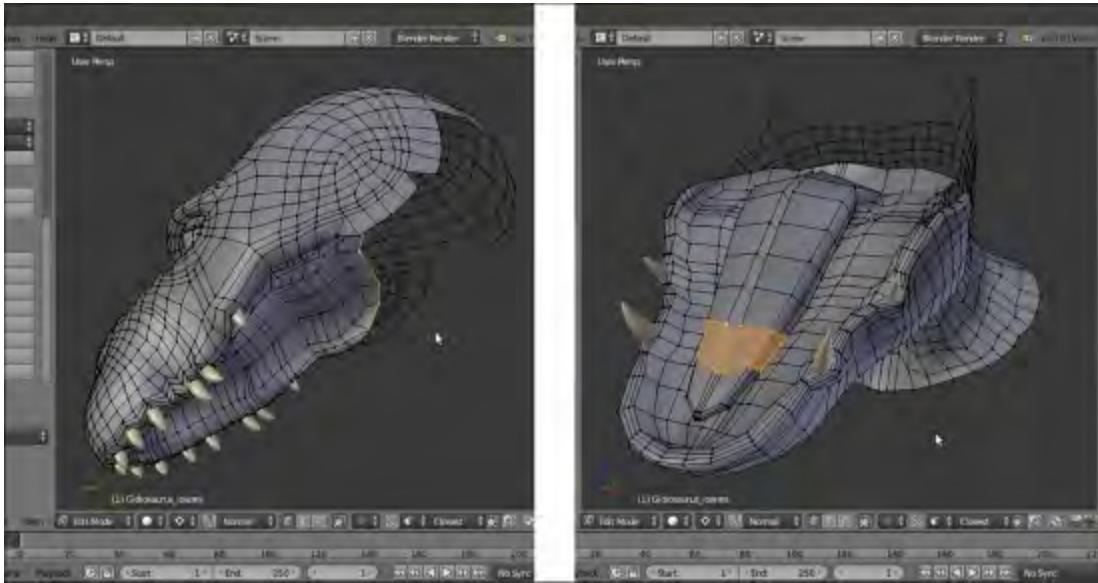
The character's anatomy

The still missing parts are modeled at this stage, such as the inside of the **nostrils** or the **eyelids**, again with the aid of the **Shrinkwrap** modifier; this time, targeted to the **Cornea** object to project the **eyelids** geometry onto it with an **Offset** value of **0.0035**:



The character's eyelids

Also, we built the **inner mouth** and the **tongue** of our character and refined the **dental alveoli**:



The character's alveoli and tongue

As in every project, we can go on with the refining, adding edge-loops, and so on, and this would seem a never-ending work; instead, at this point, we can consider the **Gidiosaurus** re-topology at the end, so it's time to apply the **Shrinkwrap** modifiers and, if this is the case, select the **Gidiosaurus** body's still separated objects and join them together to have a single mesh.

It's time to do the same with the **armor** that is still waiting on the **13th** scene layer:



The totally completed re-topologized character with the armor

How it works...

First, we have to quickly build the geometry using the **Shrinkwrap** modifier technique and then set the visibility of this geometry to wireframe (**Wire**), to make the underlying sculpted mesh visible.

The **Shrinkwrap** modifier, in the first case with the **Offset** value set high enough to allow the wireframe visibility over the sculpted surface, ensured that all the moved vertices and the new added geometry are automatically wrapped around the target mesh to preserve the volume.

At the end, we took back the **Offset** value to **0.000** anyway and we applied the **Shrinkwrap** modifier; then, we joined the re-topologized **arm** and **leg** objects together to the **Gidiosaurus_lowres** one.

As you have probably noticed, we haven't applied the **Mirror** modifiers yet. This is because it will still be useful in the next chapter.

Chapter 5. Unwrapping the Low Resolution Mesh

In this chapter, we will cover the following recipes:

- Preparing the low resolution mesh for unwrapping
- UV unwrapping the mesh
- Editing the UV islands
- Using the Smart UV Project tool
- Modifying the mesh and the UV islands
- Setting up additional UV layers
- Exporting the UV Map layout

Introduction

So, at this point, we have **sculpted** our high resolution character and through the **retopology** process, we have obtained a low resolution *copy*, which is easier to use for rigging, texturing, and animation.

There are several ways to apply textures to a mesh in Blender, as in any other 3D package. In our case, we are going to use **UV Mapping**, which is certainly one of the most commonly used and efficient methods for organic shapes.

Before the unwrapping process, the mesh must be prepared to make the task easier.

Preparing the low resolution mesh for unwrapping

In this recipe, we'll fix the last details such as the position of some of the character's parts (for instance, the **closed mouth**) and in general, anything that is needed to facilitate the unwrapping.

Getting ready

To be more precise, before the unwrapping, we must perform the following tasks in the right order:

1. Join the **teeth** and **talons** to the **body**.
2. Create the vertex group for the **mandible**.
3. Open the **mouth**.
4. Mark the seams to unwrap the **body**.

So, open the `Gidiosaurus_retopology.blend` file and deactivate the layer with the **armor** to hide it; select the **Gidiosaurus** object and save the file as `Gidiosaurus_unwrap.blend`.

How to do it...

The simplest of the *four* tasks just so happens to be the first, joining the **body** with the **teeth** and **talons**.

To join the body parts, follow these steps:

1. Select the **Talons** item in the **Outliner**, and then hold *Shift* and select the **Fangs_bottom**, **Fangs_upper**, and **Gidiosaurus_lowres** items.
2. Press *Ctrl + J* to join them.
3. Right away we will notice that, because the retopologized mesh didn't have any material assigned, the whole object gets the only material available, which is the **Enamel** material we had assigned to the **talons** and **teeth** earlier.
4. To fix this, assign a new material, or you can also assign the already existing **Body** material, to the retopologized mesh before the joining operation.
5. Alternatively, after the joining, click on the + icon to the side of the material names, and then select the **New** button in the **Material** window to create a new material. Now, enter **Edit Mode**, put the mouse pointer on the **Gidiosaurus** mesh, and press the *L* key to select all the **connected vertices**. Because the **talons** and **teeth** vertices are joined, *but not connected* to the **face** vertices, they don't get selected; for the same reason, you have to repeat the operation three times to select the **head**, **arm**, **hips**, and **leg** vertices:



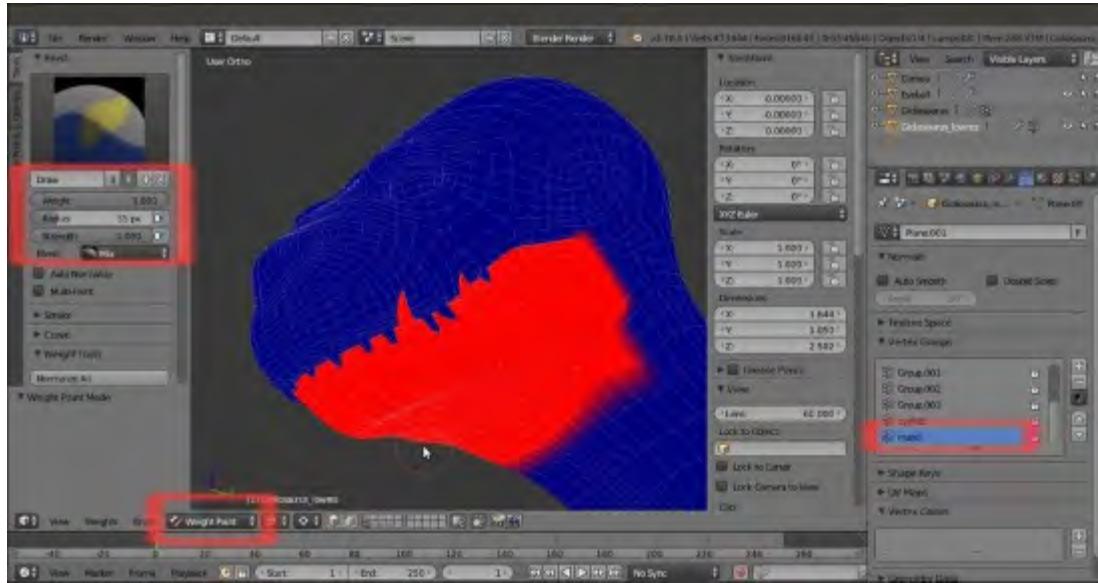
The head, arm and hip/leg vertices selected in Edit Mode

6. Click on the **Assign** button and go out of the **Edit Mode**. Now, edit the name and color of the new material or whatever, or else switch it with the **Body** one.

The second task is a bit more complex and is covered in more detail in [Chapter 7, Skinning the Low Resolution Mesh](#), which is about the **skinning** process. However, we need to explore this subject a little bit now, as it will help us operate on a small portion of the mesh easily.

To create a **vertex group** to open the **mouth**, follow these steps:

7. Go to the **Side** view and zoom in on the **head** of the character.
8. Go to the **Object Data** window; under the **Vertex Groups** subpanel, add a new group and rename it **mand** (short for **mandible**).
9. Press **Ctrl + Tab** to go into **Weight Paint** mode (or left-click on the mode button on the 3D window toolbar to switch from **Edit Mode** to **Weight Paint** mode); press **Z** to go into **Wireframe** viewport shading mode so that you can see the edges of the topology.
10. By using a combination of vertex selection mode, both in **Edit Mode** and by painting with **Weight** and **Strength** as **1.000** in the **Weight Paint** mode, assign vertices to the group of the **mandible** area and the part of the **neck**; obviously, you have to include the vertices of the inner bottom **jaw**, as well as the **tongue** and bottom **teeth**:



The visualization of the mand vertex group

11. Press **Ctrl + Tab** to exit the **Weight Paint** mode.

Note that a vertex group can be edited at a later time, so it will be easier to set the exact amount of weight on the vertices by looking at the **Lattice** modifier feedback, which is the next step.

So, to open the **mouth**, perform the following steps:

12. Add an **Empty** object to the center of the scene (**Shift + A | Empty | Plain Axes**).
13. Go to the **Side** view (press the **3** key on the numpad). Move the **Empty** to the position where the **mandible** should join the skull (to be precise, I placed it at this location: **X = 0.0000, Y = -0.3206, and Z = 2.2644**; go to the **Properties 3D** view sidepanel, and under the **Transform** subpanel, enter the values in the first three slots under the **Location** item).
14. To ensure that the **Empty** cannot be moved anymore, click on the lock icon on the right-hand side of its slot in the **Outliner** and also rename it to **Empty_rot_mand**:



The Empty_rot_mand in place

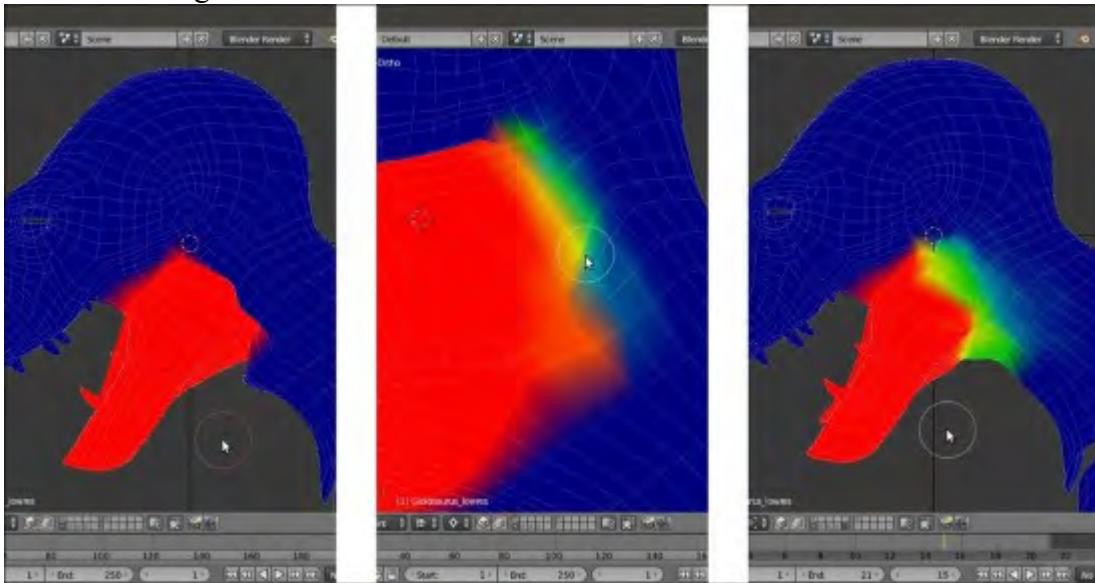
15. With the **Empty** still selected, press **Shift + S | Cursor to Selected**.
16. Add a **Lattice** object to the scene (**Shift + A | Lattice**), and in the **Object Data** window, set **Interpolation Type** for **U**, **V**, and **W** to **Linear**; select the **Gidiosaurus** object and go to the **Objects Modifier** window; assign a **Lattice** modifier. Move it before the **Subdivision Surface** modifier.
17. In the **Object** field, select the **Lattice** item; in the **Vertex Group** field, select the **mand** item.
18. In the **Side** view, select the **Lattice** object, go into **Edit Mode**, and select all the vertices and rotate them **35** degrees counterclockwise around the **x** axis:



Rotating the Lattice to open the mouth

As you can see, the **Lattice** only affects the vertices inside the **mand** vertex group; however, there is a clear indentation on the throat where the **mand** vertex group ends abruptly, so now we must blur this boundary to keep the smooth curved transition from the bottom **jaw** to the **neck**, and remove the abrupt edge.

19. Go back into the **Weight Paint** mode (*Ctrl + Tab*) and click on the **Brush** icon at the top of the **Tools** tab to switch the **Draw** brush with the **Blur** brush, and then start to blur the boundaries of the **mand** vertex group.
20. Sometimes, blurring the edge weights is not enough, so go back to the **Draw** brush, set the **Strength** to **0.500** (or whatever value you find works best), and paint on the vertices; then refine the transition again with the **Blur** brush:



Blurring and painting the weights

21. To make the job easier and faster, you can temporarily disable the **Lattice** modifier, as well as the **Subdivision Surface** modifier.
22. When you are done, go out of the **Weight Paint** mode, apply the **Lattice** modifier, and delete the **Lattice** object.
23. Make sure to keep the **Empty_rot_mand**, which will turn out to be useful when rigging the character. For now, just hide or move it onto a different layer.

At this point, we can obviously edit the **throat** area vertices as usual: relaxing and tweaking them and so on. Actually, this is the right moment to tweak all the vertices and any areas that couldn't be done before, such as the inside of the **mouth**, the **inner cheeks**, and so forth, because now we are going to do the last preparation task before the unwrapping.

To mark seams for the unwrapping of the body, we have to perform the following steps. Because our low resolution mesh is actually still only one half side, we don't need to place seams as median cuts, we only need to divide different areas (for example, the **inside of the**

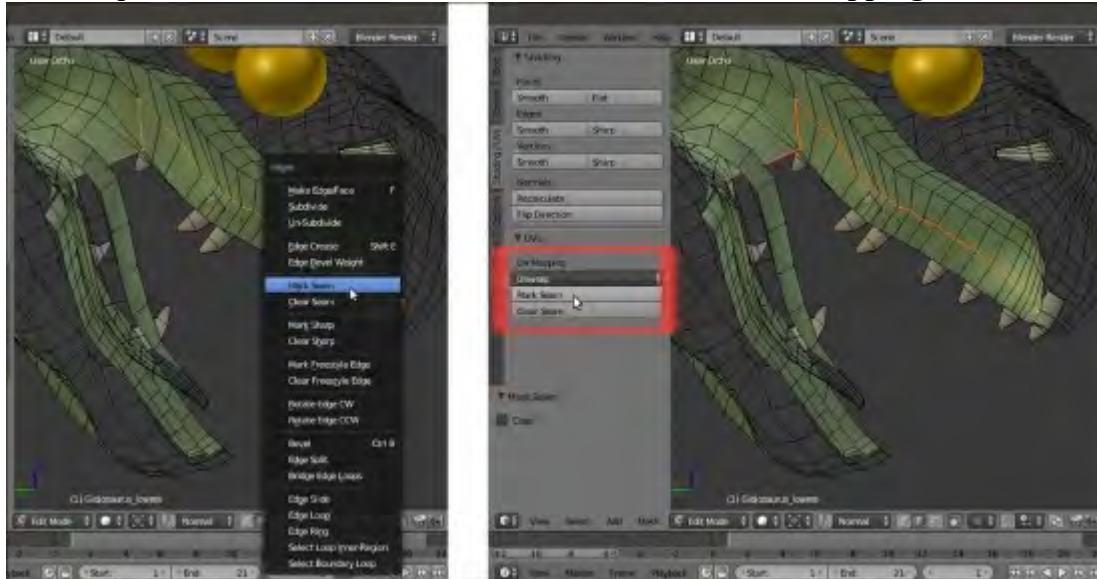
mouth from the **outside of the mouth**) and unroll cylindrical parts such as the **arms, fingers, and teeth**:

24. Go into **Edit Mode** and zoom in to the character's **head**; press *Ctrl + Tab* to call the **Mesh Select Mode** pop-up menu and select the **Edge** item, and then start to select the edge-loop inside the **mouth** (*Alt + right-click* to select an edge-loop); start from the bottom **jaw**, switching direction at the end of the **mouth rim** to go upward, and finish on the inside of the upper **jaw**:



The selected edge-loops inside the mouth

25. Press *Ctrl + E* to open the **Edges** pop-up menu and select the **Mark Seam** item. Alternatively, click on the **Shading / UVs** tab in the **Tool Shelf**, to the left-hand side of the screen, and in the **UVs** subpanel, click on the **Mark Seam** button under the **UV Mapping** item:



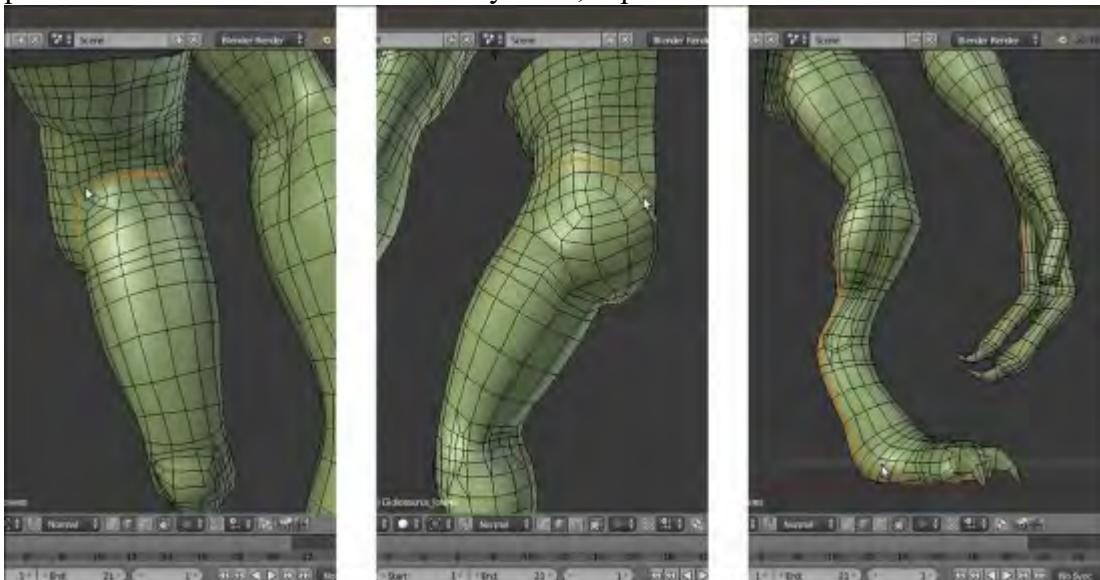
Marking the seams

26. Repeat the procedure for the **arm**; try to place the seams in the less visible areas:



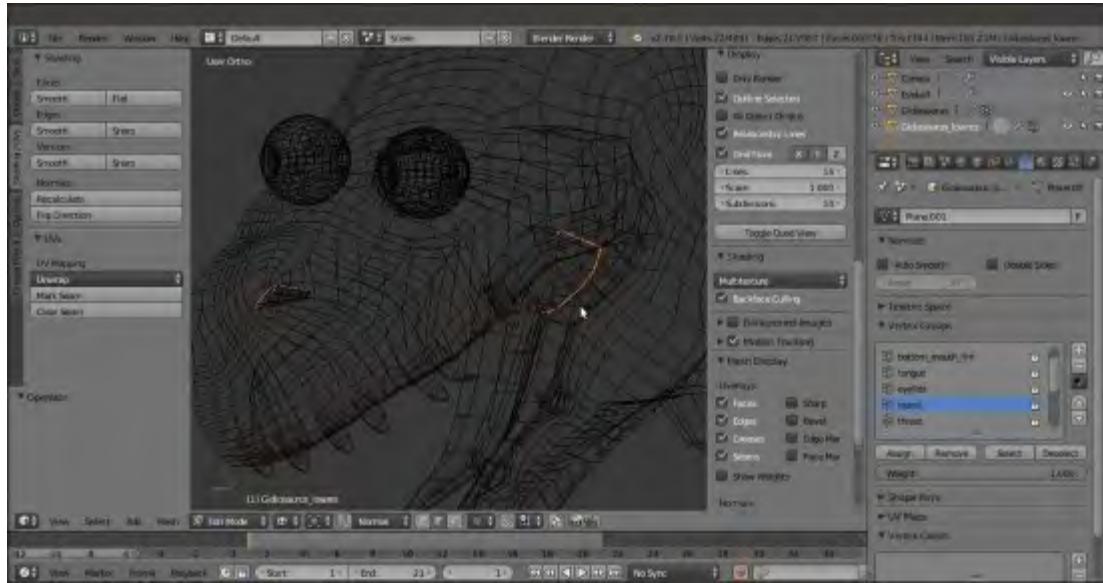
The seams on the arm

27. Do the same for the **pelvis** and **leg**; divide them into two parts with the seams and also try to place the seams inside the natural body folds, if possible:



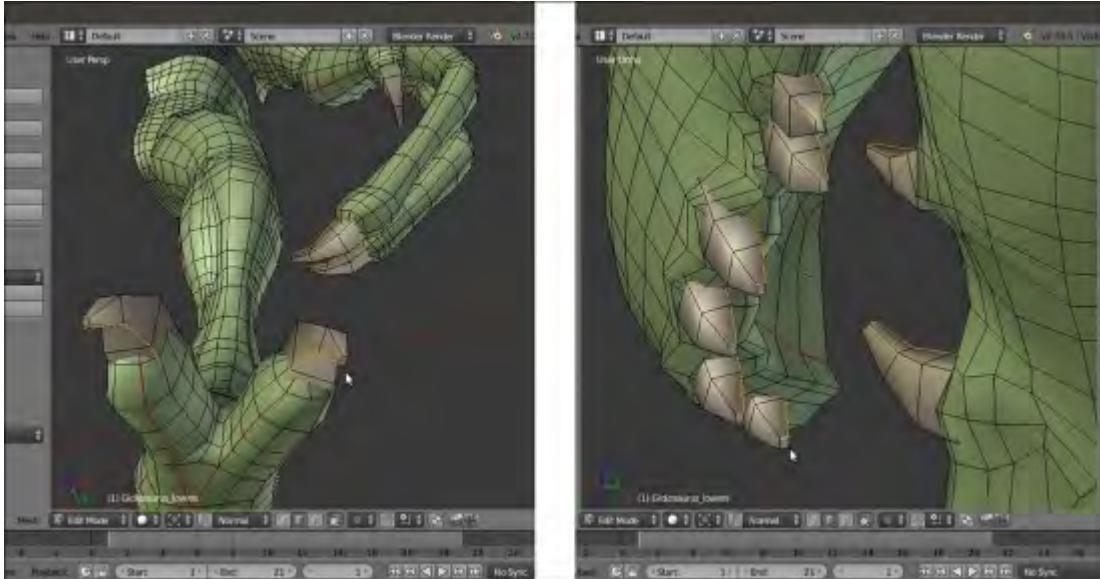
The seams on the pelvis/leg parts

28. It is important to try to place the seams to *divide* parts that would get unwrapped badly if treated as a single object; for example, the inner **nostril** and **tongue** from the **inner mouth**:



The seams inside the head

29. The final seams to add are for the teeth and talons, which would otherwise get badly unwrapped as squares:



The seams of the small parts

30. Save the file.

UV unwrapping the mesh

At this point, everything is ready for the unwrapping.

Getting ready

Put the mouse pointer on the bottom or on the top horizontal borders of the 3D window. As the mouse pointer changes to a double-arrow icon, right-click and in the **Area Options** pop-up menu select the **Split Area** item; then, left-click to obtain two windows and switch the left one to **UV/Image Editor**.

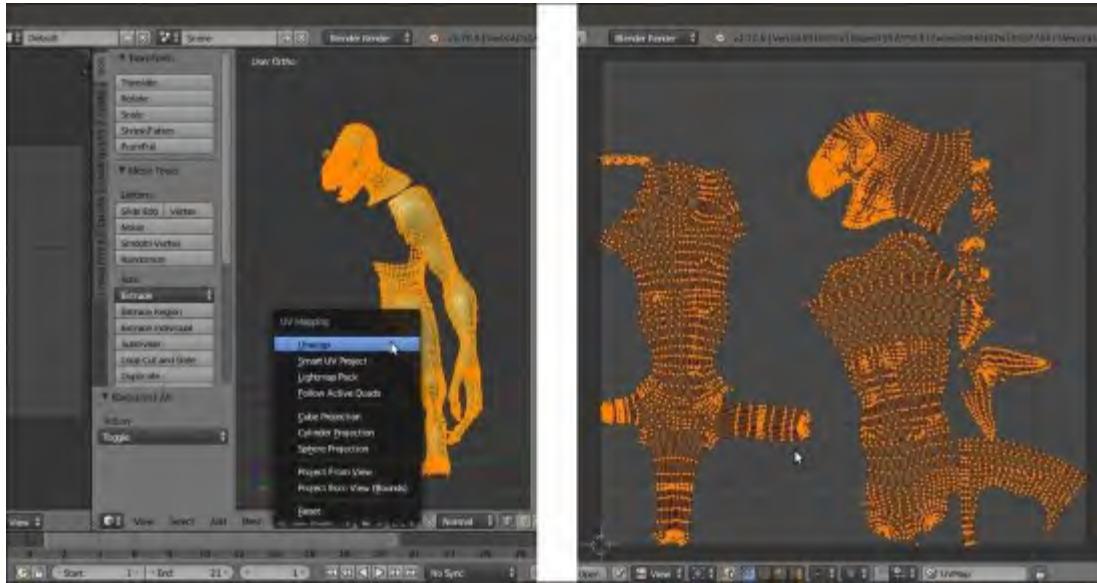


The two windows

How to do it...

To unwrap the mesh in Blender, several options are available; however, the one we are going to use now is the basic unwrap, the result of which we will edit and refine later:

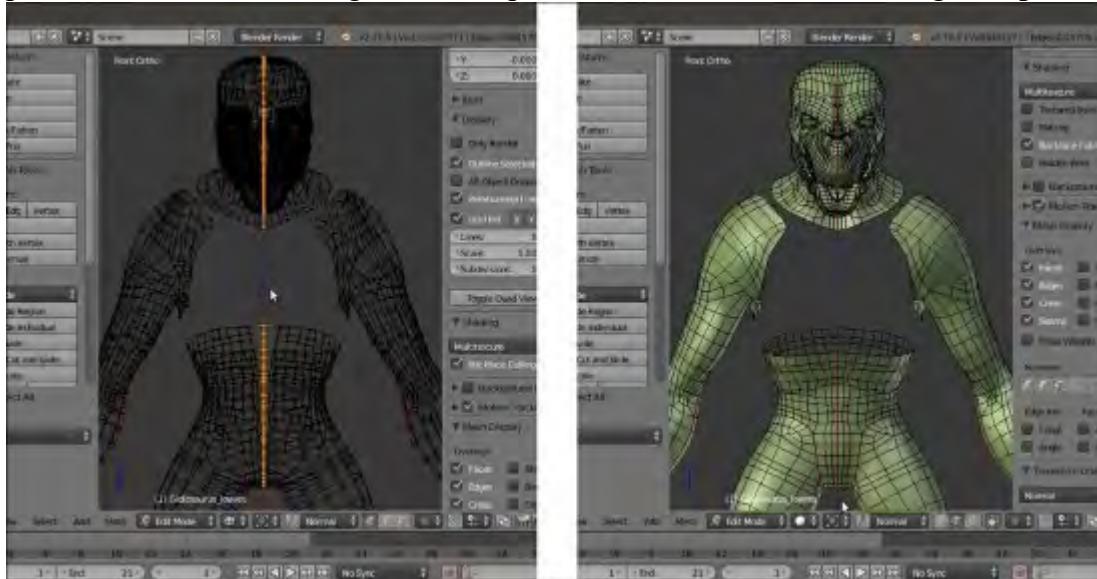
1. Ensure that the **UV/Image Editor** window is not set to **Render Result**, otherwise it won't display the **UV islands**.
2. Select the **Gidiosaurus_lowres** object and enter **Edit Mode**. Select all the vertices (A key) and press the **U** key; in the **UV Mapping** pop-up menu, select the first item, **Unwrap**:



Unwrapping the mesh

After a while, the **UV layer** of the unwrapped mesh appears in the **UV/Image Editor** window; as you can see, several things can be improved. Moreover, we are still using only half of a mesh.

3. Go out of the **Edit Mode** and go to the **Object Modifiers** window; apply the **Mirror** modifier.
4. Go back into **Edit Mode** and press **I** on the numpad to go to the **Front** view; press **Ctrl + R** and place a median seam through the **head** part of the mesh, as well as through the **pelvis** part:

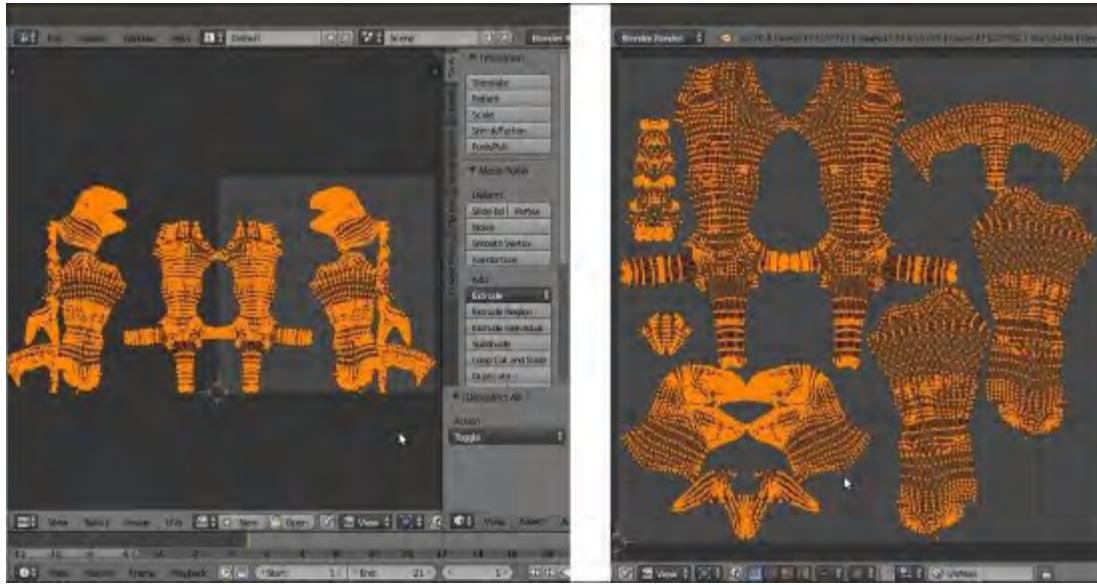


A new loop cut

5. Press *Ctrl + Tab* to switch to vertex selection and press *Z* to go into the **Wireframe** viewport shading mode, and then box-select the vertices on the left-hand side of the mesh (which is the side created by the **Mirror** modifier).
6. Go to the **UV/Image Editor** window; if not already selected, press *A* to select all the **UV islands** of the UV layer, and then press *Ctrl + M | X | Enter* to mirror these selected islands.
7. Press *G* to move them (temporarily) outside the default **U0/V0** tile space, as shown in the following screenshot:

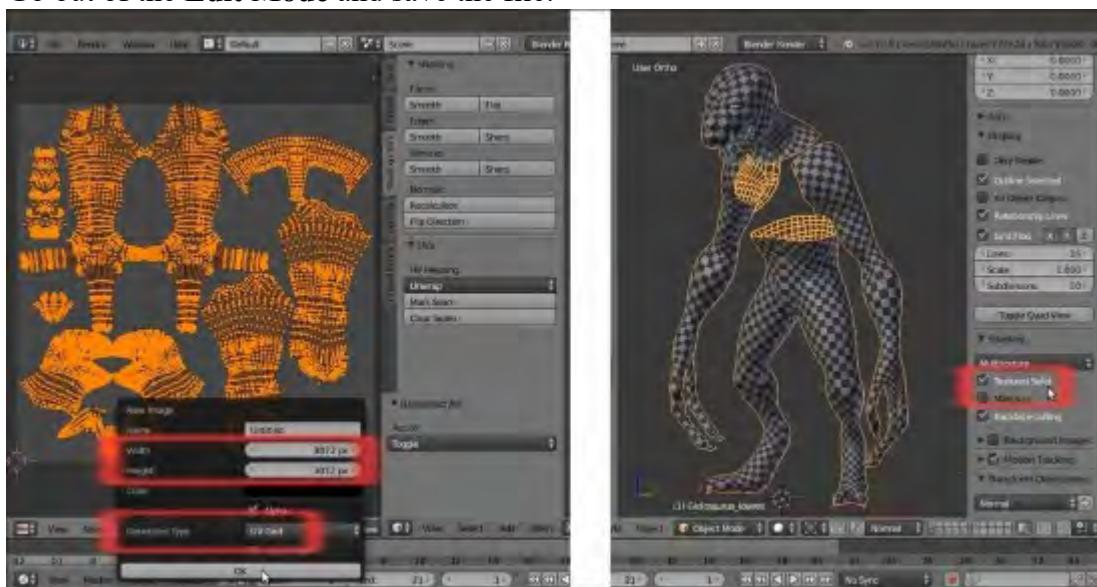


- The selected half body vertices and the corresponding UV islands outside the U0/V0 tile space*
8. Go to the 3D view and press *A* twice to select all the vertices; go to the **UV/Image Editor** window and press *Ctrl + A* to average the size of all the islands reciprocally.
 9. Select all the islands and press *Ctrl + P* to automatically pack all of them inside the UV tile.
 10. If you are not satisfied with the result of the **Pack islands** tool, adjust the position (*G* key), rotation (*R* key), and scale (*S* key) of the islands; group together the similar ones (for example, the **teeth**, **talons**, **arms**, and so on), but try to place them to fill the image tile space as much as possible. To select one island, just put the mouse over it and press *L*, and *Shift + L* to multiselect. Use the *X* and *Y* keys to constrain the movements of the islands on the corresponding axis:



Adjusting the UV islands' position

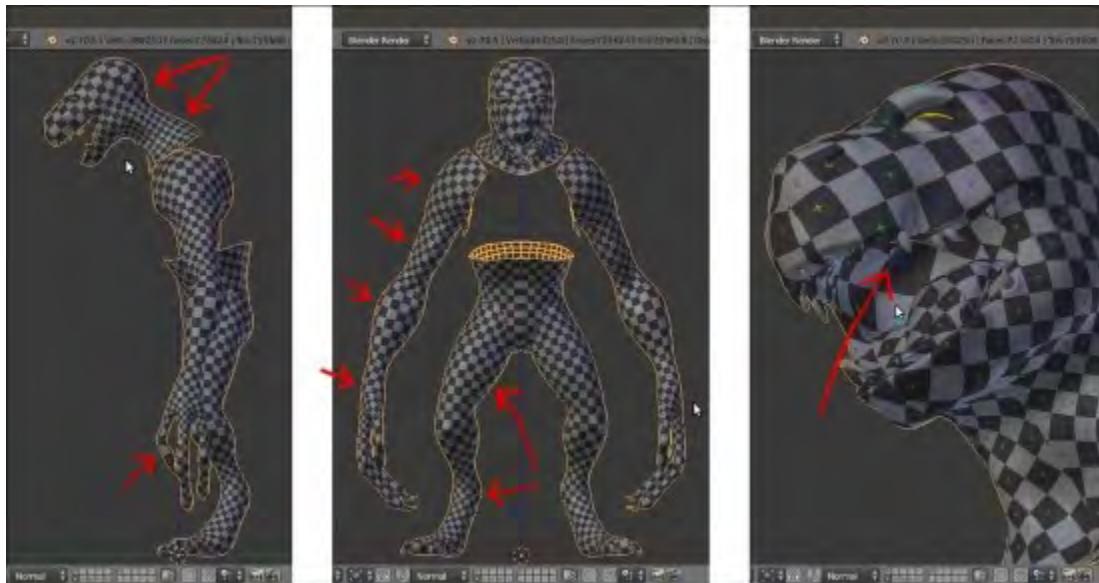
11. When you are done, ensure that all the vertices of all the islands in the **UV/Image Editor** window are selected, and click on the **New** button on the toolbar of the **UV/Image Editor** window; in the **New Image** pop-up panel, set **Width** and **Height** to **3072** pixels, and **Generated Type** should be set to **UV Grid**. Then, click on the **OK** button to confirm.
12. Go to the 3D window and press **Z** to go in the **Solid** viewport shading mode. Then, go to the **Properties** 3D view sidepanel and under the **Shading** subpanel, check the **Textured Solid** item.
13. Go out of the **Edit Mode** and save the file:



Assigning a grid image to the unwrapped UV islands

This should be enough; even if the halves of the mesh are disconnected, Blender can perfectly solve the mesh painting without visible seams.

However, if we look at the character shown in the **Textured Solid** mode in the 3D view, it's clear that the unwrap of some part of the mesh could be better; for example, you can see a difference in the size of the mapped grid in the **head/neck** area, inside the **mouth**, and on the **arms** and **legs** (look at the arrows in the following images):



Differences in the mapped grid image

Although this is not a very big issue, the unwrap can be refined further to avoid distortions as much as possible, as well as potential future problems when we'll paint the character textures; we are going to see this in the next recipe.

Editing the UV islands

We are now going to join the two UV islands' halves together, in order to improve the final look of the texturing; we are also going to modify, if possible, a little of the island proportions in order to obtain a more regular flow of the UV vertices, and fix the *distortions* we have seen in the last image of the previous recipe.

We are going to use the **pin** tool, which is normally used in conjunction with the **Live Unwrap** tool.

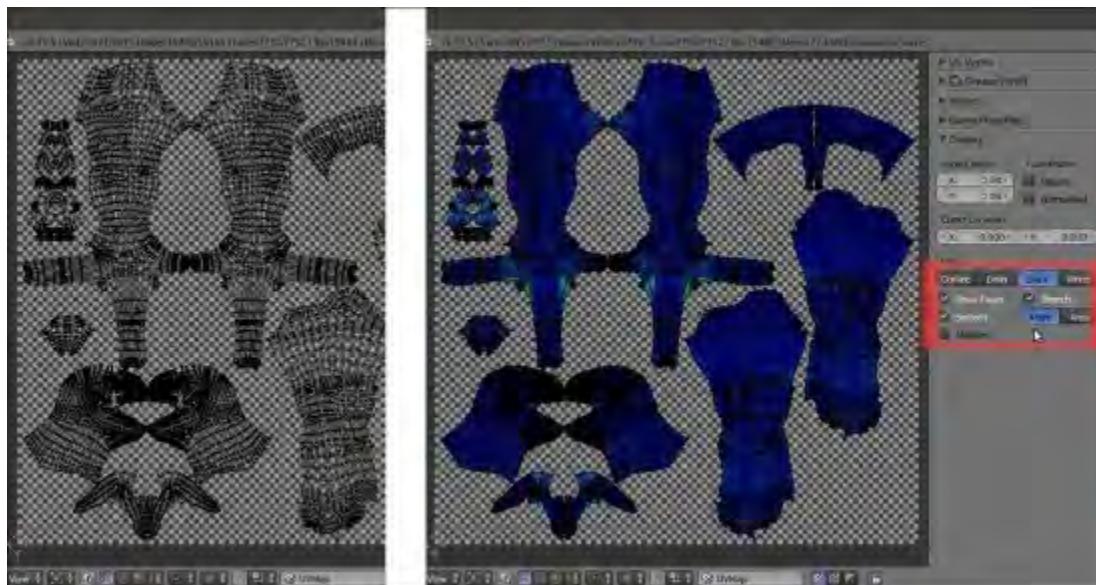
Getting ready

First, we'll try to recalculate the unwrap of some of the islands by modifying the seams of the mesh.

Before we start though, let's see if we can improve some of the visibility of the UV islands in the **UV/Image Editor**:

1. Put the mouse cursor in the **UV/Image Editor** window and press the **N** key.
2. In the **Properties** sidepanel that appears by pressing the **N** key on the right-hand side of the window, go to the **Display** subpanel and click on the **Black** or **White** button (depending on your preference) under the **UV** item. Check also the **Smooth** item box.
3. Also, check the **Stretch** item, which even though it was made for a different purpose, can increase the visibility of the islands a lot.
4. Press **N** again to get rid of the **Properties** sidepanel.

All these options enabled should make the islands more easily readable in the **UV/Image Editor** window:

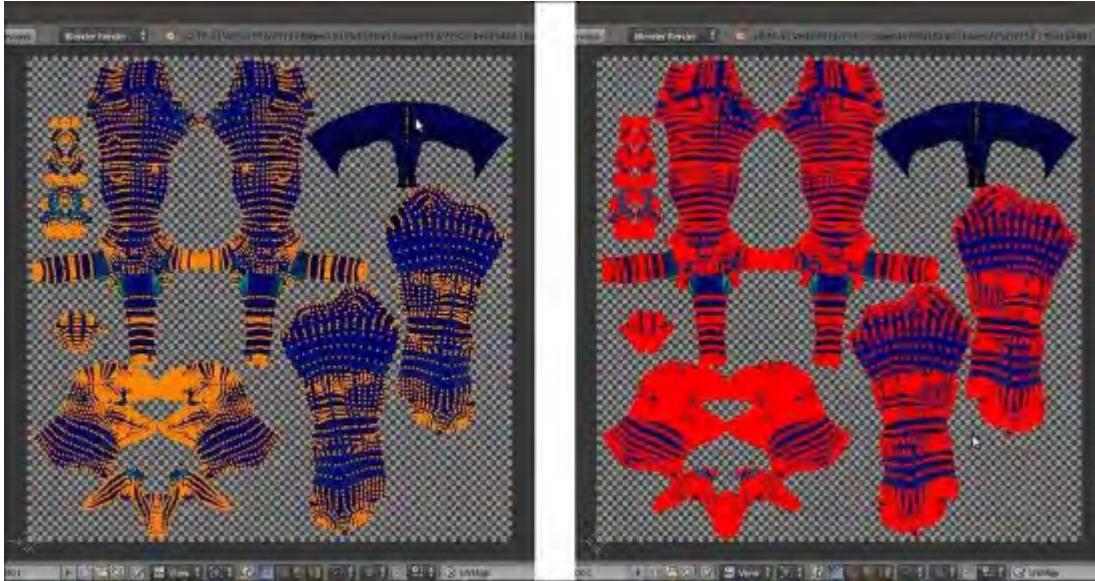


The UV islands made more easily readable by the enabled items

How to do it...

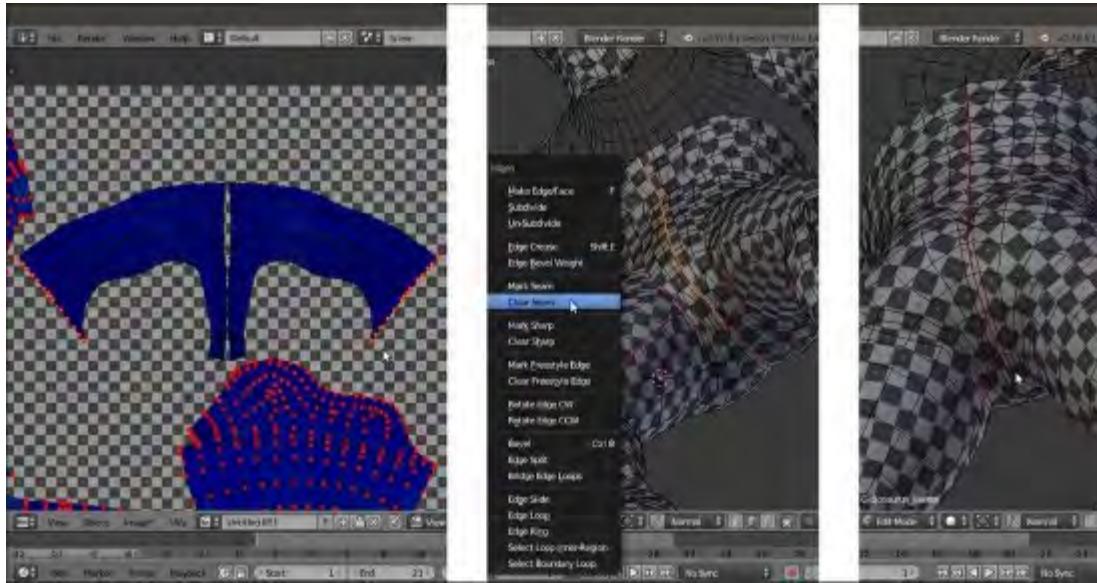
Now we can start with the editing; initially, we are going to freeze the islands that we don't want to modify because their unwrap is either satisfactory, or we will deal with it later. So, perform the following steps:

1. Press **A** to select all the islands, then by putting the mouse pointer on the two **pelvis** island halves and pressing **Shift + L**, multi-deselect them; press the **P** key to **pin** the remaining selected UV islands and then **A** to deselect everything:



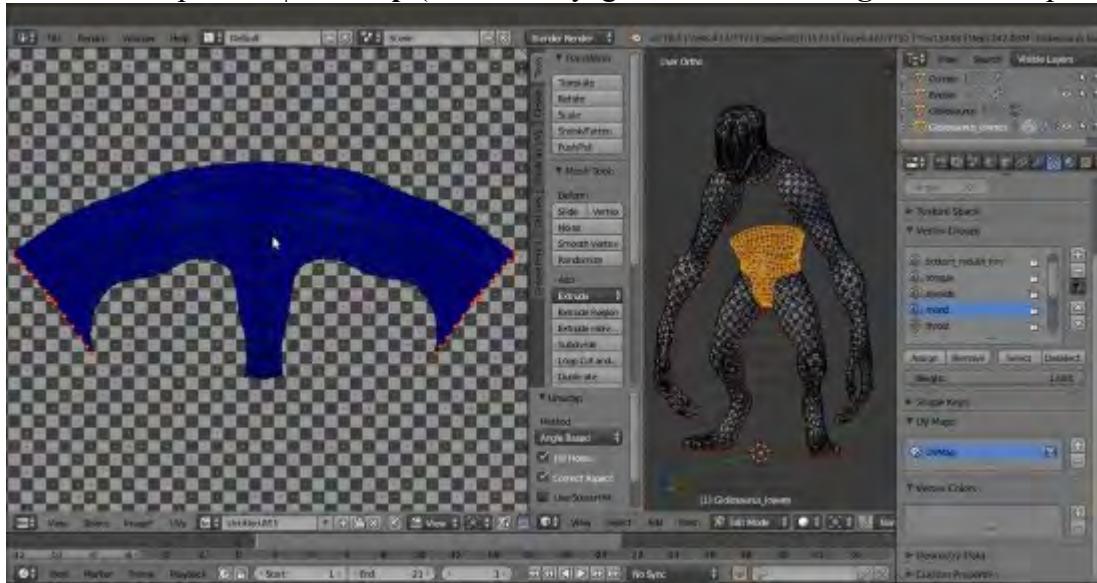
To the right-hand side, the pinned UV islands

2. Zoom in on the islands of the **pelvis**, select both the left and right outer edge-loops, as shown in the following left image, and press **P** to **pin** them.
3. Go to the 3D view and clear only the front part of the median seam on the **pelvis**. To do this, start to clear the seam from the front edges, go down and stop where it crosses the horizontal seam that passes the bottom part of the **groin** and **legs**, and leave the back part of the vertical median seam still marked:



Pinning the extreme vertices in the UV/Image Editor, and editing the seam on the mesh

4. Go into **Face** selection mode and select all the faces of the **pelvis**; put the mouse pointer in the 3D view and press **U | Unwrap** (alternatively, go into the **UV/Image Editor** and press **E**):

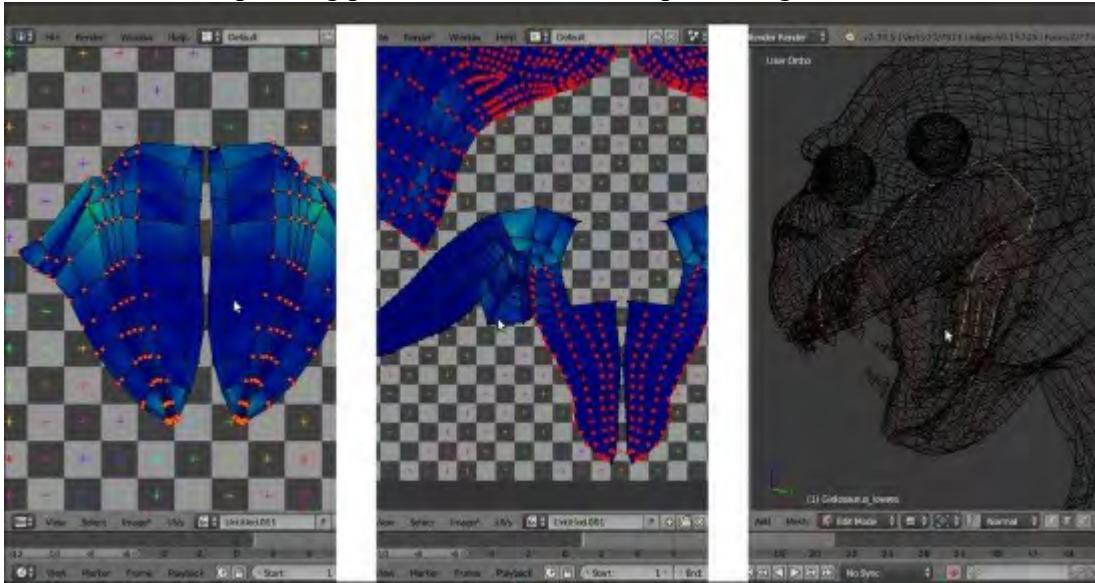


Unwrapping again with the pinning and a different seam

The island will keep the previous position because of the pinned edges, and is now unwrapped as one single piece (with the obvious exception of the seam on the back).

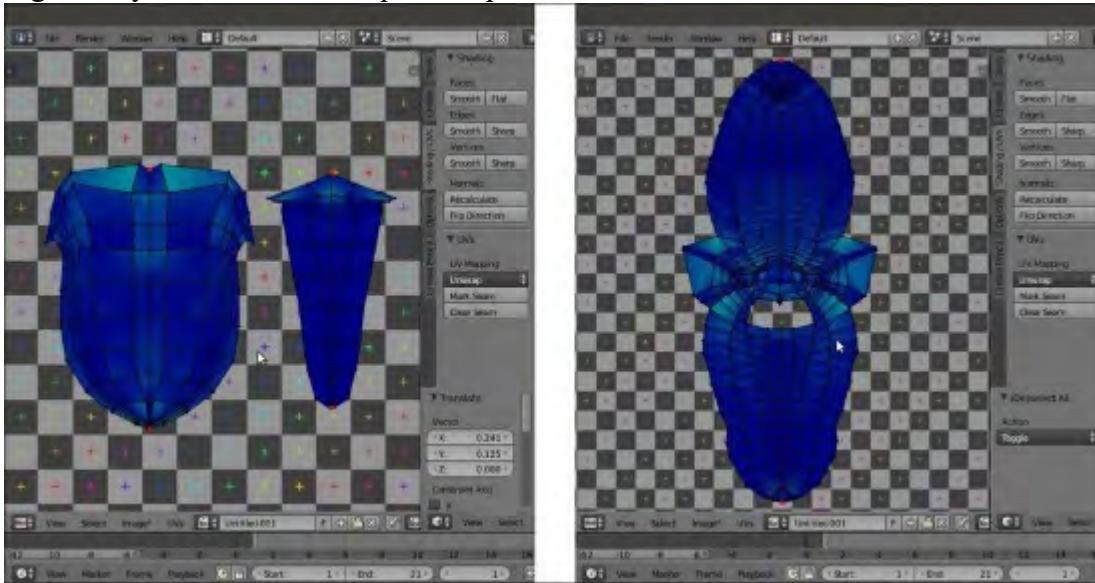
5. We won't modify the **pelvis** island any further, so select all its vertices and press **P** to pin all of them and then deselect them.

- Press **A** in the 3D view to select all the faces of the mesh and make all the islands visible in the **UV/Image Editor**. Note that they are all pinned at the moment, so just select the vertices you want to **unpin** (**Alt + P**) in the islands of the **tongue** and **inner mouth**. Then, clear the median seam in the corresponding pieces on the mesh, and press **E** again:



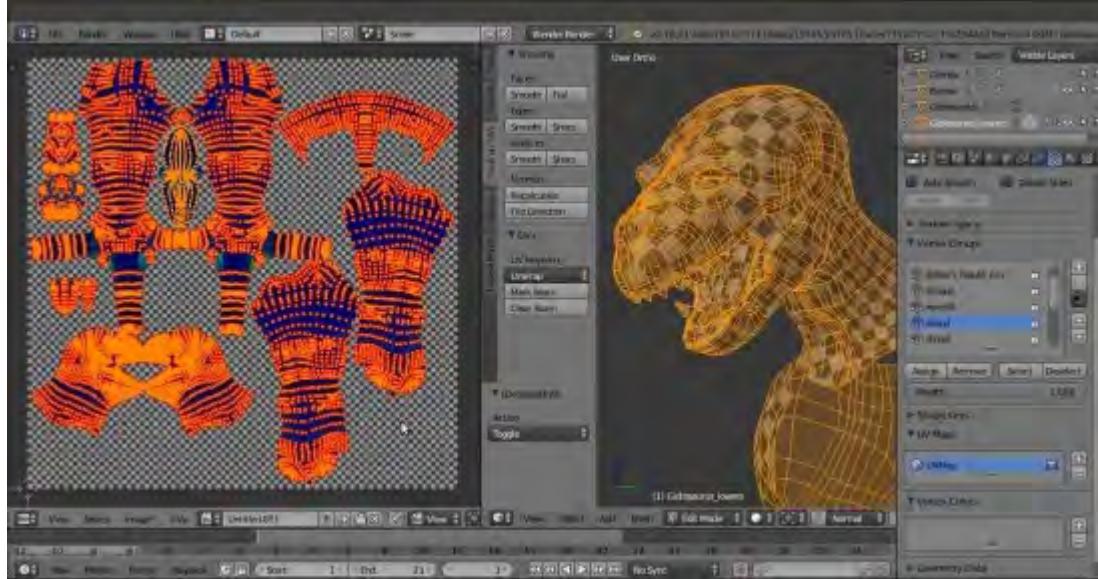
Re-unwrapping the tongue and inner mouth areas

- Select the UV vertices of the resulting islands and unpin them all; next, pin just one vertex at the top of the islands and one at the bottom, and unwrap again. This will result in a more organically distributed unwrap of the parts:



Re-unwrapping again with a different pinning

8. Select all the faces of the mesh, and then all the islands in the **UV/Image Editor** window. Press **Ctrl + A** to average their relative size and adjust their position in the default tile space:



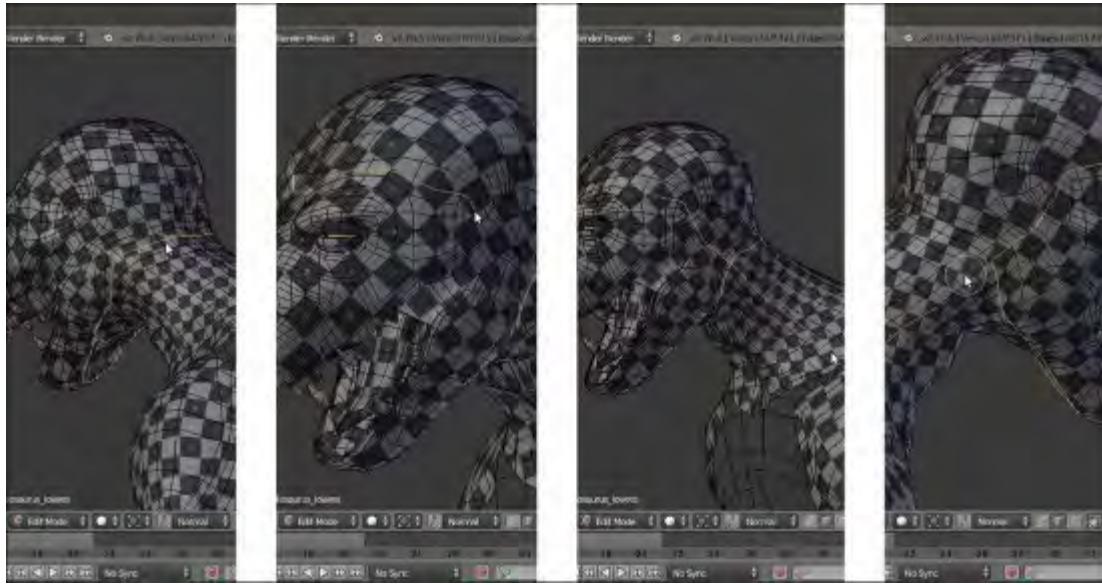
The rearranged UV islands

Now, let's work on the **head** piece that, as in every character, should be the most important and well-finished piece.

At the moment, the **face** is made using two separate islands; although this won't be visible in the final textured rendering of our character, it's always better, if possible, to join them in order to have a single piece, especially in the front mesh faces. Due to the elongated **snout** of the character, if we were to unwrap the **head** as a single piece simply without the median seam, we wouldn't get a nice evenly mapped result, so we must divide the whole **head** into more pieces.

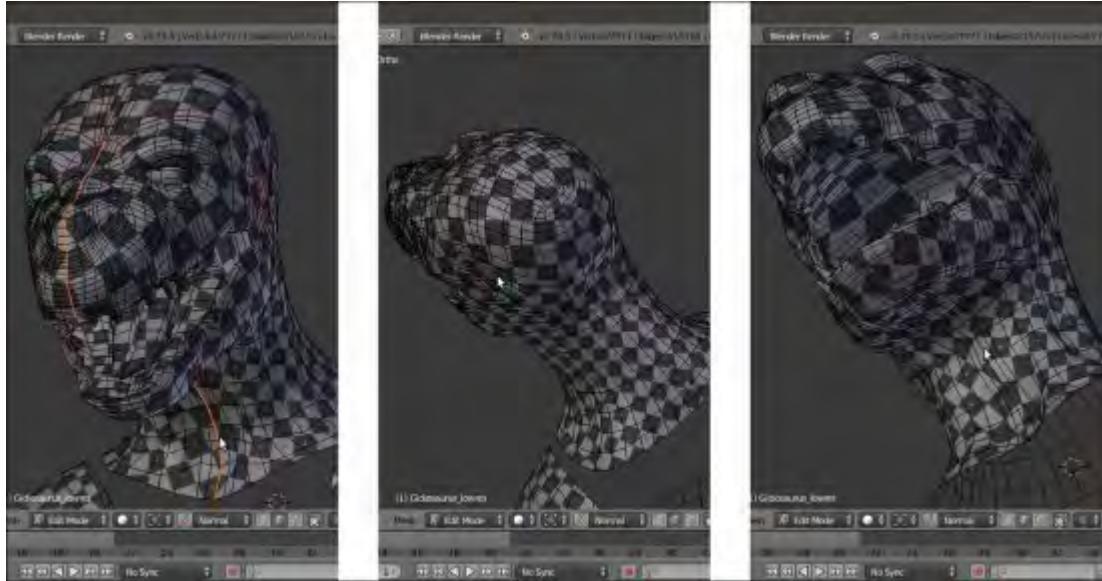
Actually, we can take advantage of the fact that the **Gidiosaurus** is wearing a **helmet** and that most of the **head** will be covered by it; this allows us to easily split the **face** from the rest of the mesh, hiding the seams under the **helmet**.

9. Go into **Edge** selection mode and mark the seams, dividing the **face** from the **cranium** and **neck** as shown in the following screenshots. Select the crossing edge-loops, and then clear the unnecessary parts:



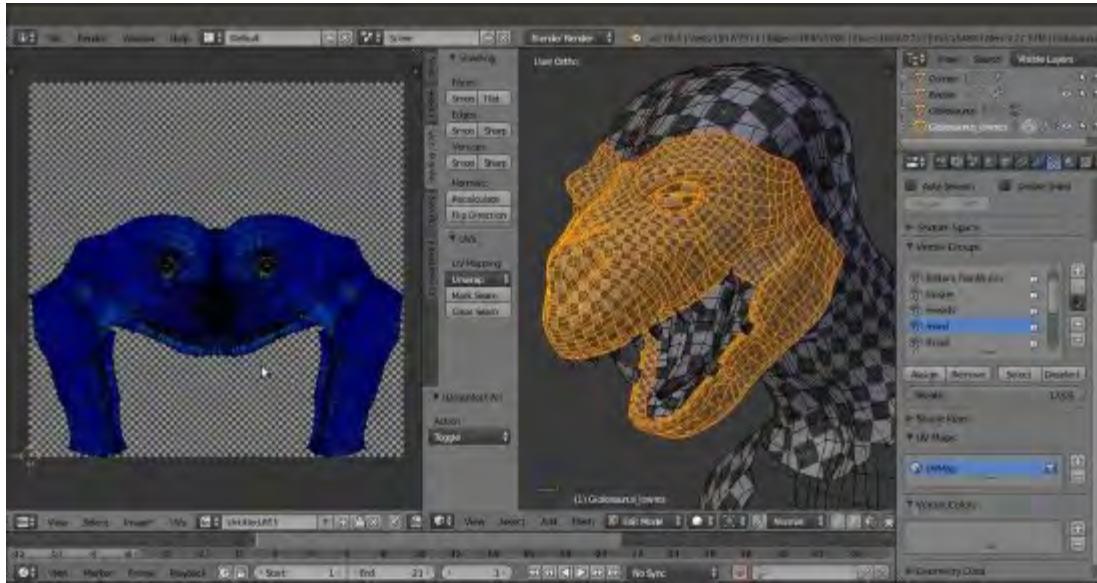
New seams for the character's head part 1

10. Also clear the median seam in the upper **face** part, and under the seam on the bottom **jaw**, leaving it only on the front **mandible** and on the back of the **cranium** and **neck**:



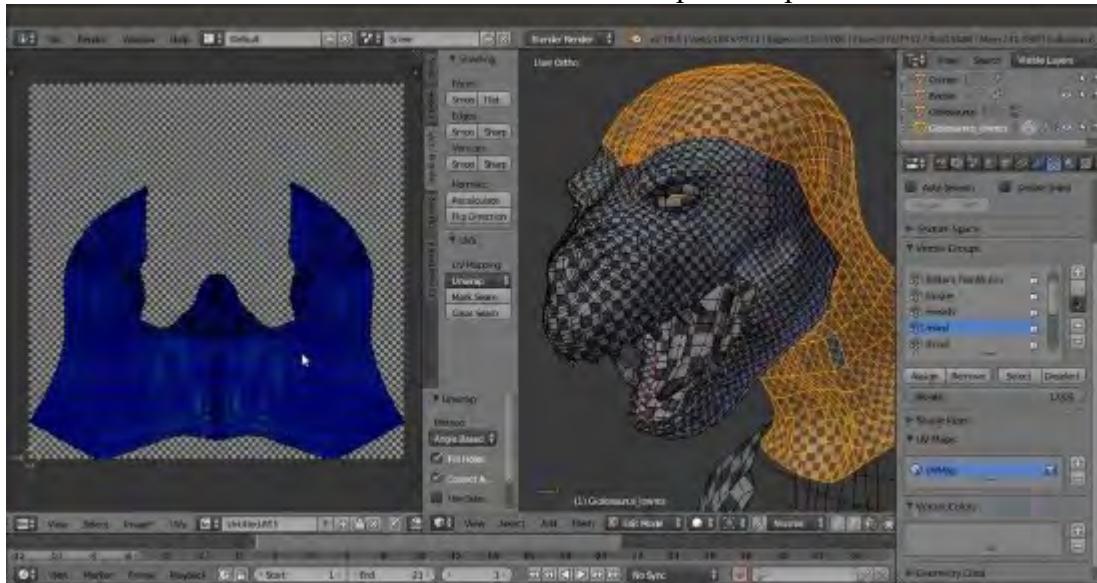
New seams for the character's head part 2

11. Go in the **Face** selection mode and select only the **face** section of the mesh, and then press **E** to unwrap. The new unwrap comes upside down, so select all the UV vertices and rotate the island by **180** degrees:



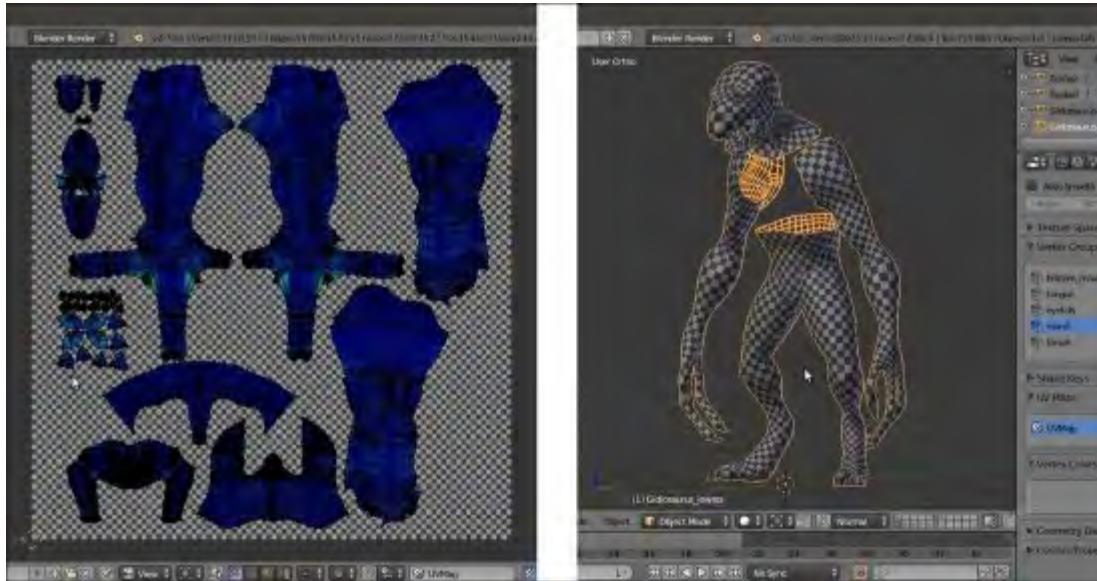
The character's face unwrapped

12. Select the **cranium/neck** section on the mesh and repeat the process:



The rest of the head mesh unwrapped as a whole piece

13. Now, select all the faces of the mesh and all the islands in the **UV/Image Editor**, and press **Ctrl + A** to average their reciprocal size.
14. Once again, adjust the position of the islands inside the UV tile (**Ctrl + P** to automatically pack them inside the available space, and then tweak their position, rotation, and scale):



The character's UV islands packed inside the default U0/V0 tile space

How it works...

Starting from the UV unwrap in the previous recipe, we improved some of the islands by joining together the halves representing common mesh parts. When doing this, we tried to retain the already good parts of the unwrap by **pinning** the UV vertices that we didn't want to modify; this way, the new unwrap process was forced to calculate the position of the unpinned vertices using the constraints of the pinned ones (**pelvis**, **tongue**, and **inner mouth**). In other cases, we totally cleared the old seams on the model and marked new ones, in order to have a completely new unwrap of the mesh part (the **head**), we also used the character furniture (such as the **armor**) to hide the seams (which in any case, won't be visible at all).

There's more...

At this point, looking at the **UV/Image Editor** window containing the islands, it's evident that if we want to keep several parts in proportion to each other, some of the islands are a little too small to give a good amount of detail when texturing; for example, the Gidiosaurus's face.

A technique for a good unwrap that is the current standard in the industry is **UDIM UV Mapping**, which means **U-Dimension**; basically, after the usual unwrap, the islands are scaled bigger and placed outside the default **U0/V0** tile space.

Look at the following screenshots, showing the Blender **UV/Image Editor** window:

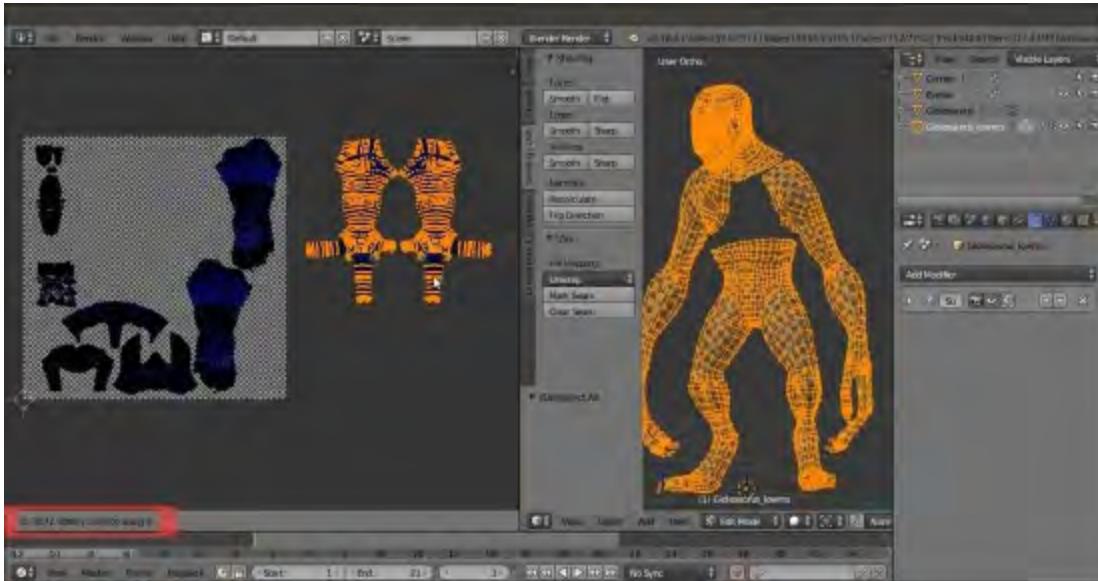


The default U0/V0 tile space and the possible consecutive other tile spaces

On the left-hand side, you can see, highlighted with red lines, the single UV tile that at present is the standard for Blender, which is identified by the UV coordinates **0** and **0**: that is, **U** (horizontal) = **0** and **V** (vertical) = **0**.

Although not visible in the **UV/Image Editor** window, all the other possible consecutive tiles can be identified by the corresponding UV coordinates, as shown on the right-hand side of the preceding screenshot (again, highlighted with red lines). So, adjacent to the tile **U0/V0**, we can have the row with the tiles **U1/V0**, **U2/V0**, and so on, but we can also go upwards: **U0/V1**, **U1/V1**, **U2/V1**, and so on.

To help you identify the tiles, Blender will show you the amount of pixels and also the number of tiles you are moving the islands in the toolbar of the **UV/Image Editor** window. In the following screenshot, the **arm** islands have been moved horizontally (on the negative *x* axis) by **-3072.000** pixels; this is correct because that's exactly the *X* size of the grid image we loaded in the previous recipes. In fact, in the toolbar of the **UV/Image Editor** window, while moving the islands we can read **D: -3072.000** (pixels) and (inside brackets) **1.0000** (tile) **along X**; effectively, **3072** pixels = **1** tile.



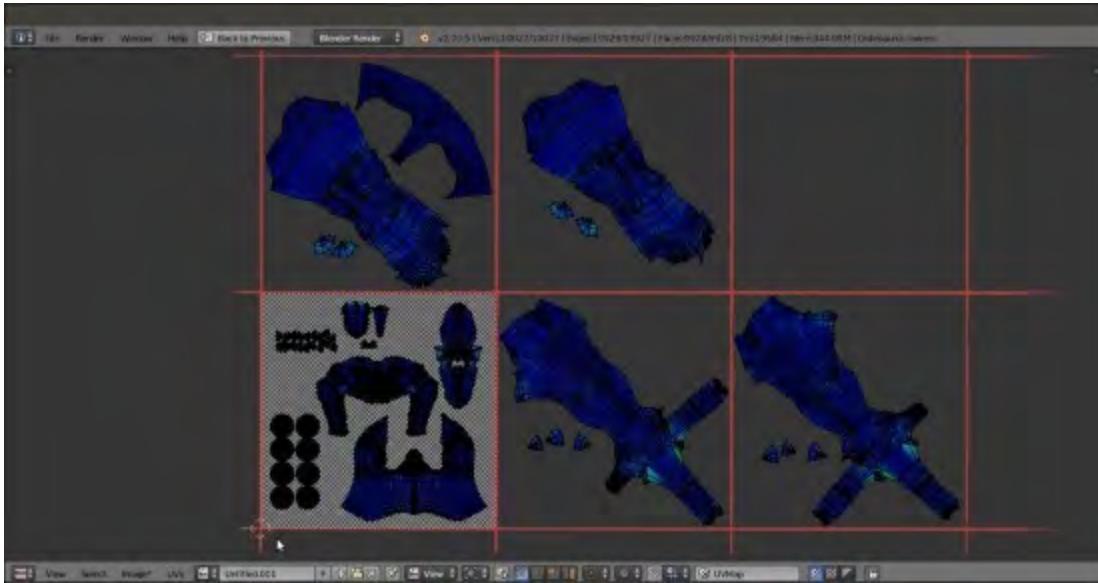
Moving the arm islands to the U1/V0 tile space

When moving UV islands from tile to tile, remember to check that the **Constrain to Image Bounds** item in the **UVs** menu on the toolbar of the **UV/Image Editor** window is disabled; also, enabling the **Normalized** item inside the **Display** subpanel under the **N** key *Properties* sidepanel of the same editor window will display the UV coordinates from **0.0** to **1.0**, rather than in pixels. More, pressing the **Ctrl** key while moving the islands will constrain the movement to intervals, making it easy to translate them to exactly 1 tile space.

Because at the moment Blender doesn't support the **UDIM UV Mapping** standard, simply moving an island outside the default **U0/V0** tile, for example to **U1/V0**, will *repeat the image* you loaded in the **U0/V0** tile and on the faces associated with the moved islands. To solve this, it's necessary, after moving the islands, to *assign a different material, if necessary with its own different image textures*, to each group of vertices/faces associated with each tile space. So, if you shared your islands over **4** tiles, you need to assign **4** different materials to your object, and each material must load the proper image texture.

The goal of this process is obviously to obtain bigger islands mapped with bigger texture images, by selecting all the islands, scaling them bigger *together* using the largest ones as a guide, and then tweaking their position and distribution.

One last thing: it is also better to unwrap the **corneas** and **eyes** (which are separate objects from the **Gidiosaurus** body mesh) and add their islands to the tiles where you put the **face**, **mouth**, **teeth**, and so on (use the **Draw Other Objects** tool in the **View** menu of the **UV/Image Editor** window to also show the UV islands of the other *nonjoined* unwrapped objects):



UV islands unwrapped, following the UDIM UV Mapping standard

In our case, we assigned the **Gidiosaurus** body islands to **5** different tiles, **U0/V0**, **U1/V0**, **U2/V0**, **U0/V1**, and **U1/V1**, so we'll have to assign **5** different materials. However, we will cover this in a later recipe.

Note that for exposition purposes only, in the preceding screenshot, you can see the cornea and eye islands together with the **Gidiosaurus** body islands because I temporarily joined the objects; however, it's usually better to maintain the eyes and corneas as separate objects from the main body.

Using the Smart UV Project tool

Now, we are going to use a much easier and faster method to do the unwrapping of the **Armor**: the **Smart UV Project** tool.

Getting ready

The first thing to do is to prepare the **armor** pieces for the unwrap process, so perform the following steps:

1. Starting from the last `Gidiosaurus_unwrap.blend` file you saved, click on the **13th** scene layer to reveal the **armor** and at the same time, hide the **Gidiosaurus_lowres** object.
2. Go to the **Outliner** and select the first item, the **Breastplate**; then, use *Shift* to multiselect all the other visible objects.
3. Press *Ctrl + J* to join them into a single object, and then in the **Outliner**, rename the result as **Armor**.
4. Go to the **Object Modifiers** window and expand the **Mirror** modifier subpanel; be sure that the **Clipping** item is activated and click on the **Apply** button:



The Armor as a single object and the Mirror modifier

How to do it...

Here is the unwrap process:

1. Press *Tab* to go into **Edit Mode** and press the *A* key to select all the vertices of the **Armor**.
2. With the mouse cursor in the 3D view, press the *U* key, and in the **UV Mapping** pop-up menu that just appeared, select the second item from the top, **Smart UV Project**.

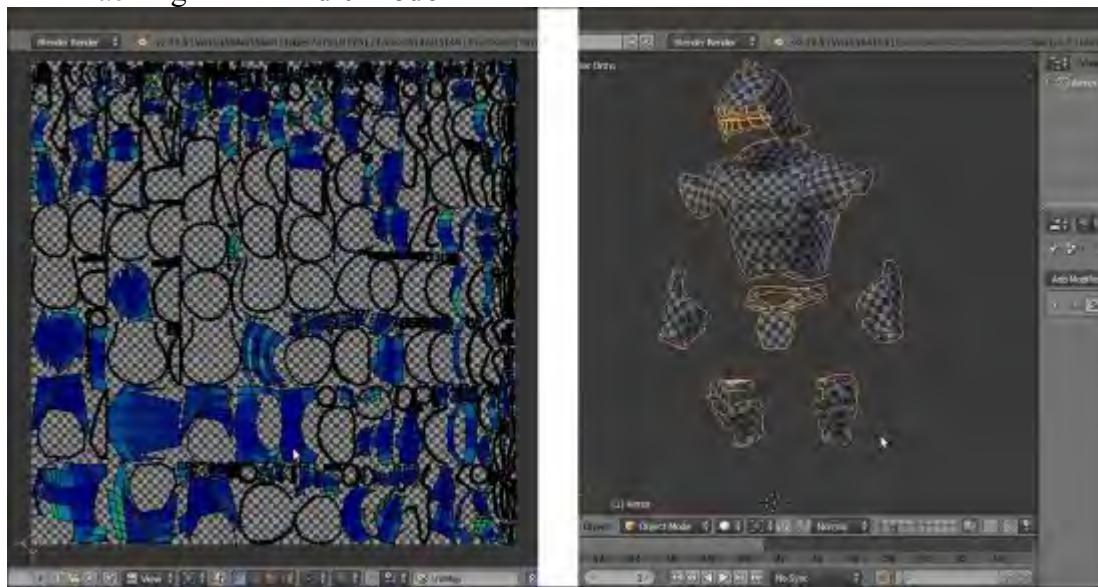
- A second pop-up appears with some options that you can leave as they are, besides **Angle Limit** (the maximum angle in the mesh used by the tool to separate the islands), which by default is set to **66.00**; raise it to the maximum, which is **89.00**, and then click on the big **OK** button:



The Smart UV Project tool

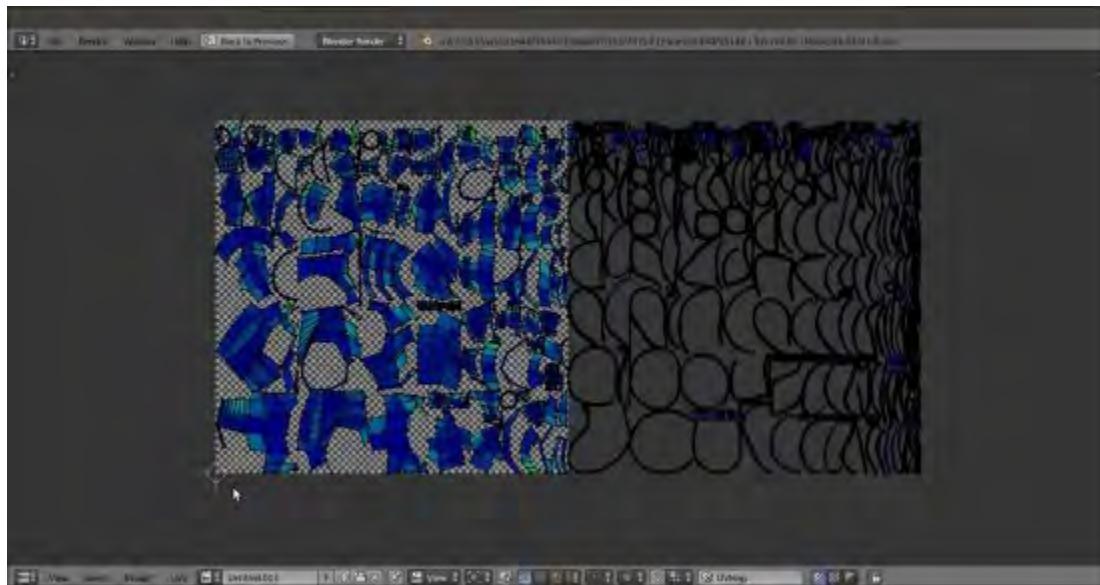
The mesh has been divided into several smaller unwrapped parts and is automatically packed inside the **U0/V0** UV tile.

- Select all the islands in the **UV/Image Editor** window, click on the small double-arrow icon on the toolbar, close to the **New** and **Open** buttons, and select the **Untitled.001** image (the same grid image we used for the **Gidiosaurus** unwrap).
- Press **Tab** to go out of **Edit Mode**:



The unwrapped Armor

Considering the amount of tiny islands that the tool created, it's better to separate the big **armor** parts (basically, the **plates**) from the smaller ones (**belts**, **Borders**, and so on) and re-unwrap them with the **Smart UV Project** tool, as we did for the **Gidiosaurus** body in the previous recipe; then, place them into two adjacent tiles:



The Armor islands inside the U0/V0 and U1/V0 tiles

Modifying the mesh and the UV islands

At this point, when we look at the **Gidiosaurus** mesh, we realize that some detail in the model is still missing; for example, the **lower teeth**. In fact, we modeled the **mouth** closed and the **lower teeth**, enclosed in the **upper mouth rim**, weren't visible.

It's now time to add them; in fact, even though we have already done the unwrapping stage, it's still possible to modify the mesh further and also update the UV islands accordingly.

Getting ready

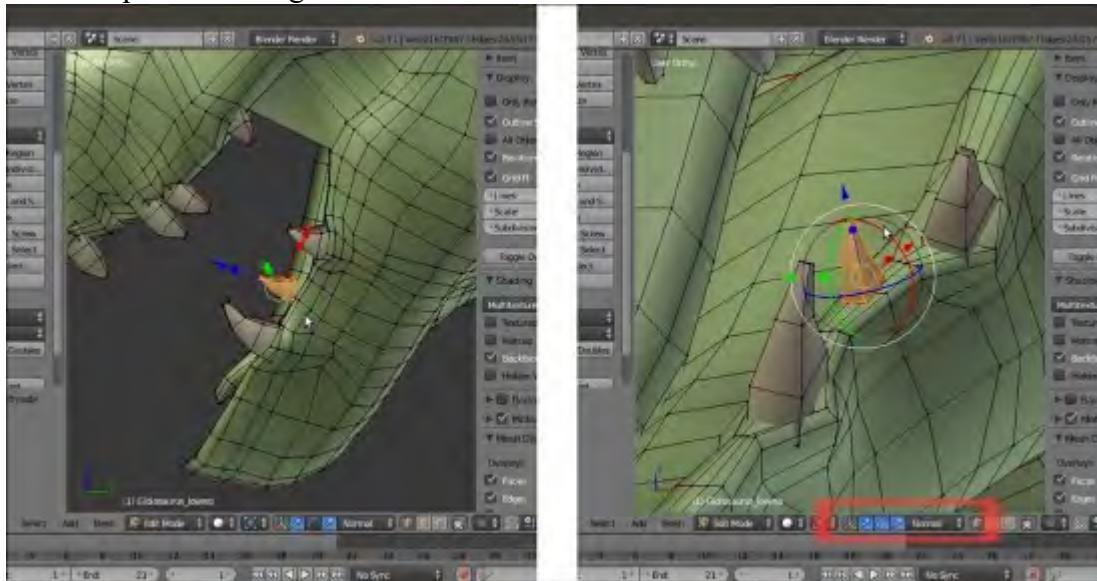
Start from the last `Gidiosaurus_unwrap.blend` file you saved:

1. Press **N** to open the **Properties** 3D view sidepanel, and disable the **Textured Solid** item under the **Shading** subpanel.
2. Click on the **11th** scene layer button to reveal the **Gidiosaurus_lowres** object and select it; go into the **Side** view, zoom in to the **head**, and enter **Edit Mode**. Press the **A** key to select all the vertices of the mesh.
3. Put the mouse pointer inside the **UV/Image Editor**, select all the UV vertices (again the **A** key), and pin them by pressing the **P** key; then deselect everything (the **A** key once more).

How to do it...

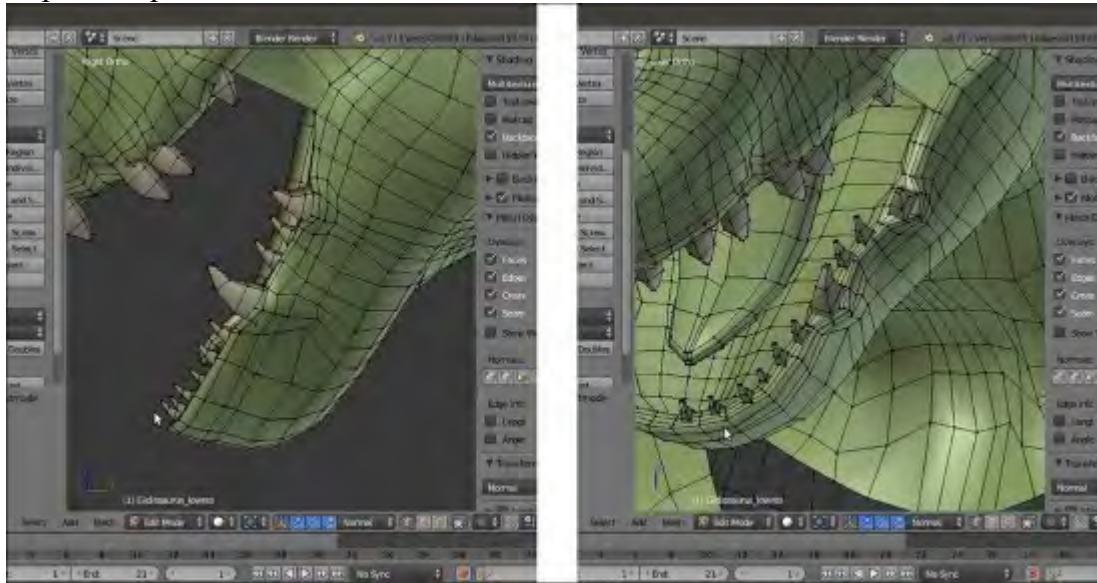
We can now start to add new **teeth**:

1. Select the vertices of one **lower tooth** and press **Shift + D** to duplicate it; using the **Transform Orientation** widget set to **Normal**, scale it smaller, rotate and modify it a bit, and then move it in a new position along the **mandible rim**:



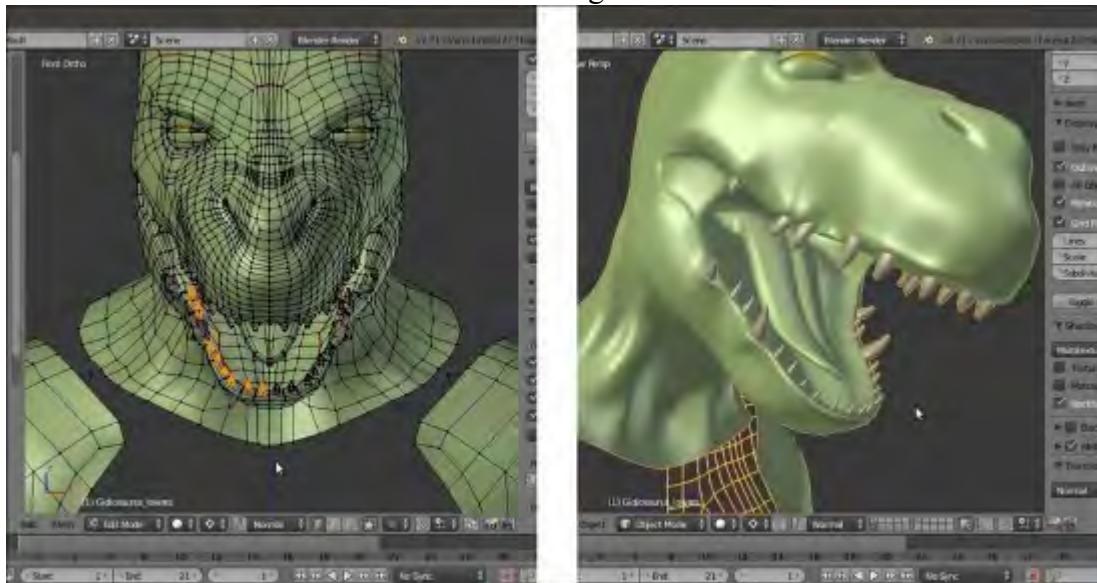
The new added tooth

2. Repeat the process to create the **bottom teeth row** on the left-hand side of the **mandible**:



Adding the missing teeth

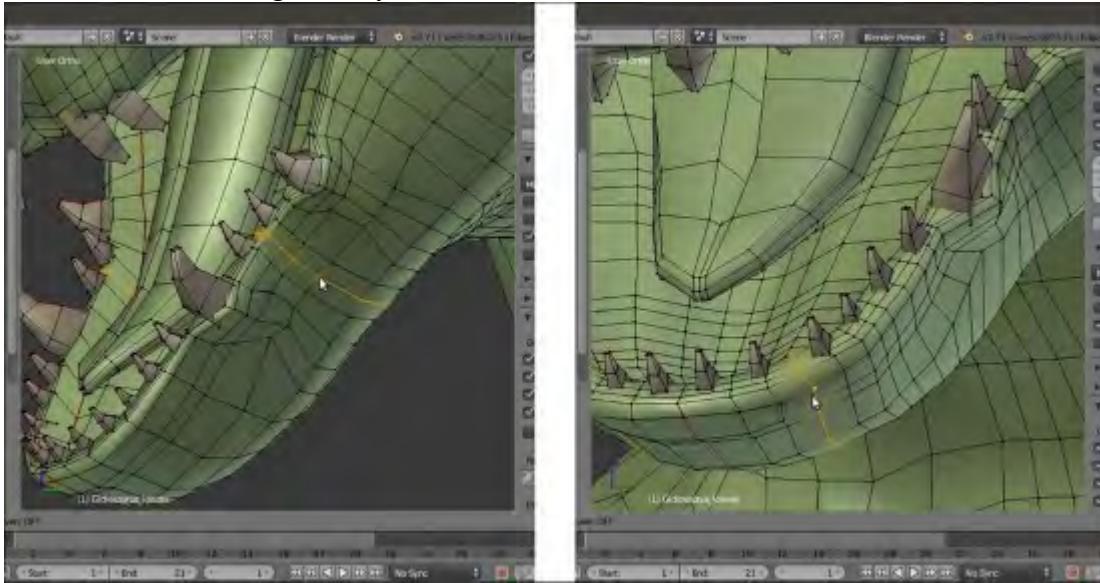
3. Go out of the **Edit Mode** and put the **3D Cursor** at the pivot point of the mesh (coincidentally, at the center of the scene), and then press the period key (.) to set the **Pivot Point** around the **3D Cursor**.
4. Press *I* on the numpad to go in the **Front** view, enter **Edit Mode** again, and select all the new **teeth**; press *Shift + D* to duplicate them, and then right-click; then, press *Ctrl + M | X | Enter* to mirror them on the *x* axis to the right-hand side of the **mandible**.
5. Press *Ctrl + N* to recalculate the normals and go out of the **Edit Mode**:



The new teeth mirrored on the x axis

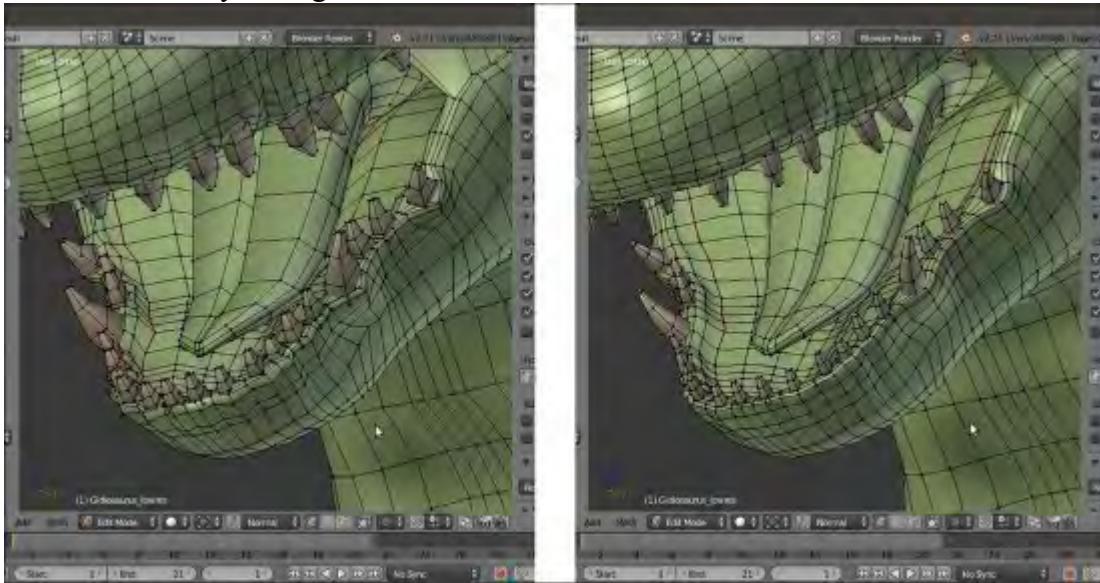
Now, we must adjust the **rim** of the **mandible** where we added the new **teeth**, in order to create the **alveoli**.

6. Go back into the **Edit Mode** and start to add vertical edge-loops on the **lower mouth rim**, in order to create more geometry for the **alveoli**:



Adding new edge-loops

7. Click on the **Options** tab under the **Tool Shelf** to the left-hand side of the 3D window and enable the **X Mirror** item under the **Mesh Options** subpanel.
8. Tweak the vertices to create the **alveoli** around the **new teeth**; enable the **Subdivision Surface** modifier visibility during **Edit Mode** in order to have better feedback:



Modeling the alveoli

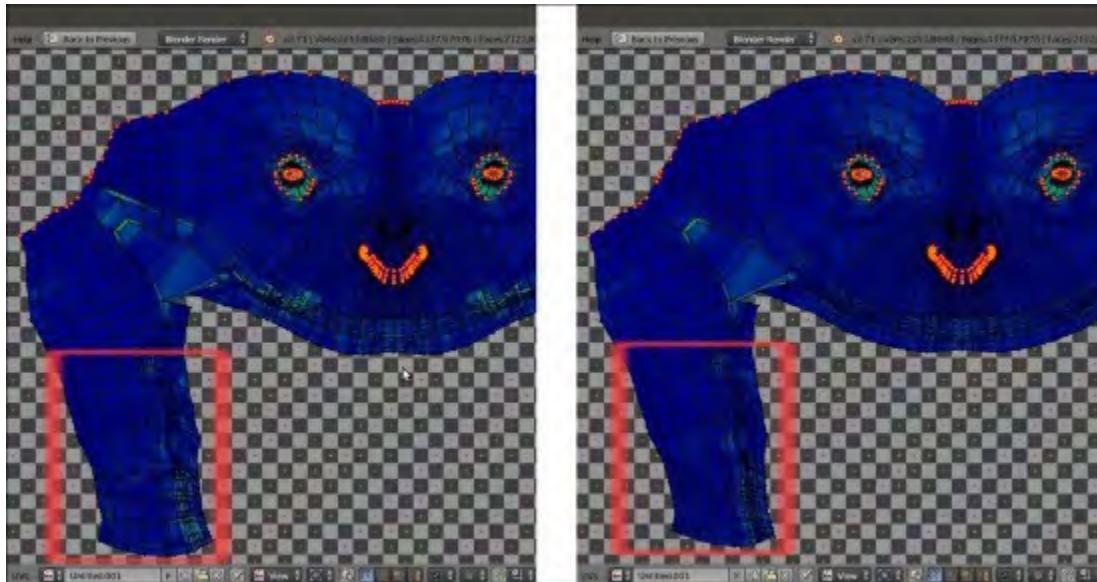
9. Press **Alt + S** to scale the **teeth** vertices on their normals, in order to thicken them, and add edge-loops where needed to make the transition from the **alveoli** to **inner mouth** as natural as possible.

When you are done, it's time to update the unwrapped UV layer with the new modifications.

10. In **Edit Mode**, first select the vertices of the **new teeth**. Because we made them by duplicating one of the already unwrapped **fangs**, the **new teeth** will share the same UV island. In the **UV/ Image Editor**, press **A** to select all their UV vertices and **Alt + P** to unpin them, and then press **E** for a new unwrap.
11. Scale the new **teeth** islands to **0.200**, and then select the original **teeth** on the mesh; adjust the size and position of the new islands based on the old ones and then pin them.
12. Now, switch to the **Face** selection mode and select the **Gidiosaurus** face; in the **UV/Image Editor** window, unpin all the vertices of its island (**Alt + P**), and then pin only the vertices of certain areas such as the **eyes**, **nose**, and upper outside edge-loop (**P** key).

With this method, the unwrap of all the new geometry gets recalculated together with the old one.

Thanks to the pinned UV vertices, it will keep the previous size and position as much as possible. In the following image, you can see the face island before (left) and after (right):



The updated unwrap

Note that you need to recalculate the unwrap for all the islands involved in the mesh's modification, and then save the file.

Setting up additional UV layers

Up until now, we have set just one UV layer whose name is, by default, **UVMap** (go to the **Object Data** window and look under the **UV Maps** subpanel):



The **UV Maps** subpanel with the **UV Map coordinates** layer

Actually, in Blender, it is possible to set more than one UV coordinates layer on the same object in order to mix different UV projections that can eventually also be baked into a single image map.

The names of the UV layers under the **UV Maps** subpanel are important, because they specify which one of the projections a material has to use for the mapping of a texture. By clicking on the + icon to the side of the **UV Maps** subpanel, it is possible to add a new UV layer (whose name, in this case, will be **UVMap.001** by default; of course it's possible to change these names by using *Ctrl* + clicking on them and typing the new ones).

Getting ready

We are now going to add a new UV layer to the **Gidiosaurus** object:

1. Ensure that the **Gidiosaurus** object is selected and go to the **Object Data** window under the main **Properties** panel to the right-hand of the screen.
2. Go to the **UV Maps** subpanel and click on the + icon to the right-hand side of the names window; a new UV layer is added to the list, right under the first one, and its name is **UVMap.001** (in case you don't see it, it may be because the window is too small; just put the mouse cursor on the = sign at the bottom of the window and drag it down to enlarge it):



The new UV coordinates layer

3. Use *Ctrl* + left-click on the **UVMap.001** item and rename it as **UVMap_scales**. Then, press *Enter* to confirm.

How to do it...

Now we must set the projection of the UV layer:

1. Go into **Edit Mode**, switch to the **Face** selection mode, put the mouse pointer on the mesh, and press the *L* key to select all the faces of the skin of the **Gidiosaurus** mesh.
2. Go to the **UV/Image Editor** window, select all the visible islands and unpin them (*Alt* + *P*).
3. Click on the **Image** item on the toolbar and select the **Open Image** item in the pop-up menu (or else, put the mouse cursor in the **UV/Image Editor** window and press *Alt* + *O*); browse to the **textures** folder and load the **scales_tiles.png** image.
4. With the mouse pointer in 3D view, press *U* and from the **UV Mapping** pop-up menu, select the **Cube Projection** item.
5. In the **UV/Image Editor** window, select all the islands and scale them **5** times bigger (*A* | *S* | **5** | *Enter*):



The Cube Projection mapping

6. Go out of the **Edit Mode** and into the **Properties** 3D view sidepanel, enable the **Textured Solid** item under the **Shading** subpanel to see the result of the unwrapping in the 3D viewport:



The scales_tiles.png image mapped on the model using the second UV coordinates layer

At this point, as you can see in the **UV Maps** subpanel, the **Gidiosaurus** object has 2 different UV coordinate layers, **UVMap** and **UVMap_scales**. We will use the **UVMap_scales** layer to map the scales image texture on the body and thereby to bake it on the first **UVMap** layer; this

will be the one we'll use in the end for the rendering of the model. However, we'll see this in detail in the texturing and baking recipes.

Repeat the process for the **Armor**.

7. Add a new UV layer and rename it **UVMap_rust**; go into **Edit Mode**, select all the vertices and all the islands in the **UV/Image Editor** window, and load the **iron_tiles.png** image.
8. Switch to the **Face** selection mode, and in the 3D view, press **U** and select **Reset** (the last item) from the pop-up menu. Then press **U** again, and this time select the **Cube Projection** item.
9. Go out of **Edit Mode**.

As you can see, there are a few visible seams. This will be easily fixed during the texturing stage, but for the moment we are done:



The second UV coordinates layer for the Armor

Exporting the UV Map layout

In this last recipe, we are going to see how to export the UV coordinate layers outside Blender, in order to be used as a guide to paint textures inside any 2D image editing software.

Getting ready

We have seen that the **Gidiosaurus** object and also the **Armor** object have more than one UV coordinate layer, so the first thing to do is to be sure to have set the right layer as the *active* one.

To do this, simply click on the name of the chosen layer inside the **UV Maps** subpanel under the **Object Data** window; if you are in **Edit Mode**, by clicking on the different names, you can also see the different layers switch in real time in the **UV/Image Editor** window.

How to do it...

After you have selected the desired UV layer, do the following:

1. Click on the **UVs** item in the toolbar of the **UV/Image Editor** window, and from the menu, select the **Export UV Layout** item (the top item).
2. You can browse the directory where the .blend file is saved, as the directory opens, at the bottom-left side of the screen is the **Export UV Layout** option panel where you can decide on several items: the size and format of the exported image, and the opacity of the islands (by default, for mysterious reasons, it is set to **25 percent** rather than **100 percent**). Moreover, you can decide if you want to export all the islands of the selected object or only the visible ones, and also if you want the modifiers applied to the islands (for example, the **Subdivision Surface** modifier).
3. Browse to the folder where you decided to save the UV layout of your model, or click on the side of the path in the upper line after the slash, and write the name of a new directory. Press **Enter** and click on the pop-up panel with the **OK? Create New Directory** message to confirm (this actually creates a brand new directory).
4. Write the name of the UV layout in the second line and click on the **Export UV Layout** button at the top-right of the screen.

Note that if you want to export all the different tiles placed outside of the default **U0/V0** tile space, as illustrated in the *There's more...* section of the *Editing the UV islands* recipe, at least for the moment, you have to temporarily (using **Ctrl**) move each island at a time to the default **U0/V0** tile space and export it.

Chapter 6. Rigging the Low Resolution Mesh

In this chapter, we will cover the following recipes:

- Building the character's Armature from scratch
- Perfecting the Armature to also function as a rig for the Armor
- Building the character's Armature through the Human Meta-Rig
- Building the animation controls and the Inverse Kinematic
- Generating the character's Armature by using the Rigify add-on

Introduction

To be able to animate our character, we have to build the rig, which in Blender is commonly referred to as an **Armature**, and this is the *skeleton* that will deform the **Gidiosaurus** low resolution mesh.

The rigging process in Blender can be accomplished basically in two different ways:

- By building the **Armature** by hands from scratch
- By using the provided **Human Meta-Rig** or the **Rigify** add-on

Building the **Armature** manually by hand can be a lot of work, but in my opinion, is the only way to really learn and understand how a rig works; on the other hand, the **Rigify** add-on gives several tools to speed up and automate the rig creation process, and this in many occasions, can be very handy.

Building the character's Armature from scratch

So, the first recipe of this chapter is about the making of the **Armature** by hands for our **Gidiosaurus**.

Getting ready

In this first recipe, we are going to build by hands the **basic rig**, which is the skeleton made only by the **deforming bones**.

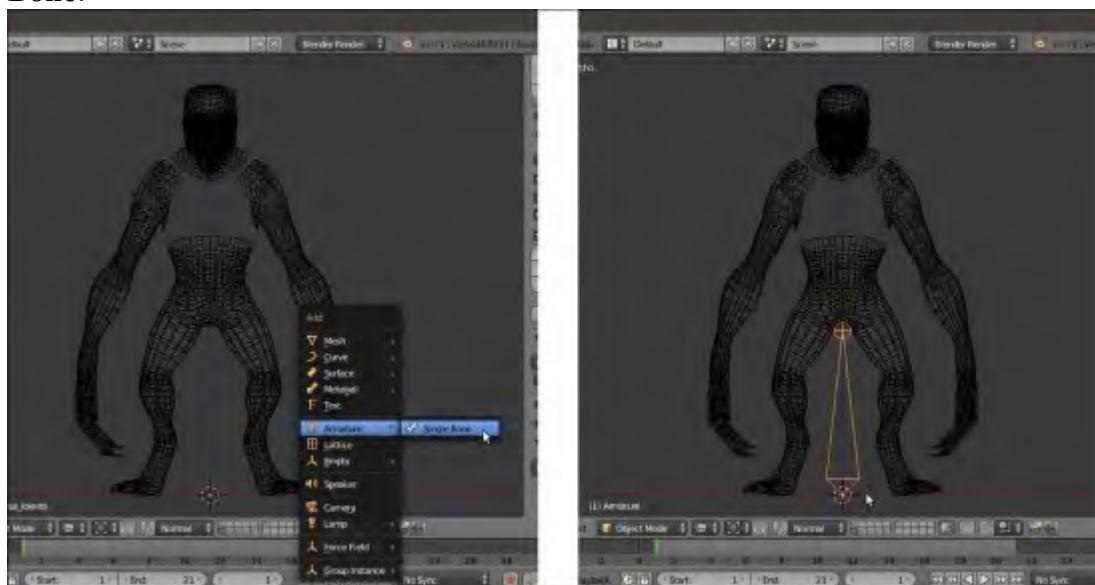
However, first, let's prepare a bit the file to be worked:

1. Start Blender and open the `Gidiosaurus_unwrap_final.blend` file.
2. Disable the **Textured Solid** and **Backface Culling** items in the 3D view **Properties** sidepanel, join the 3D window with the **UV/Image Editor** window, and click on the **11th** scene layer to have only the **Gidiosaurus** mesh visible in the viewport.
3. Go to the **Object** window under the **Display** subpanel and enable the **Wire** item. This will be useful in the process in order to have an idea of the mesh topology when in **Object Mode** and **Solid** viewport shading mode. However, for the moment, press the **Z** key to go in the **Wireframe** viewport shading mode.
4. Press **1** on the numpad to go in the **Front** view, and press **5** on the numpad again to switch to the **Ortho** view.
5. Save the file as `Gidiosaurus_rig_from_scratch_start.blend`.

How to do it...

Let's start:

1. Be sure that the **3D Cursor** is at the origin pivot point of the **Gidiosaurus** mesh. Put the mouse pointer in the 3D view, press **Shift + A**, and in the **Add** pop-up menu, select **Armature | Single Bone**:



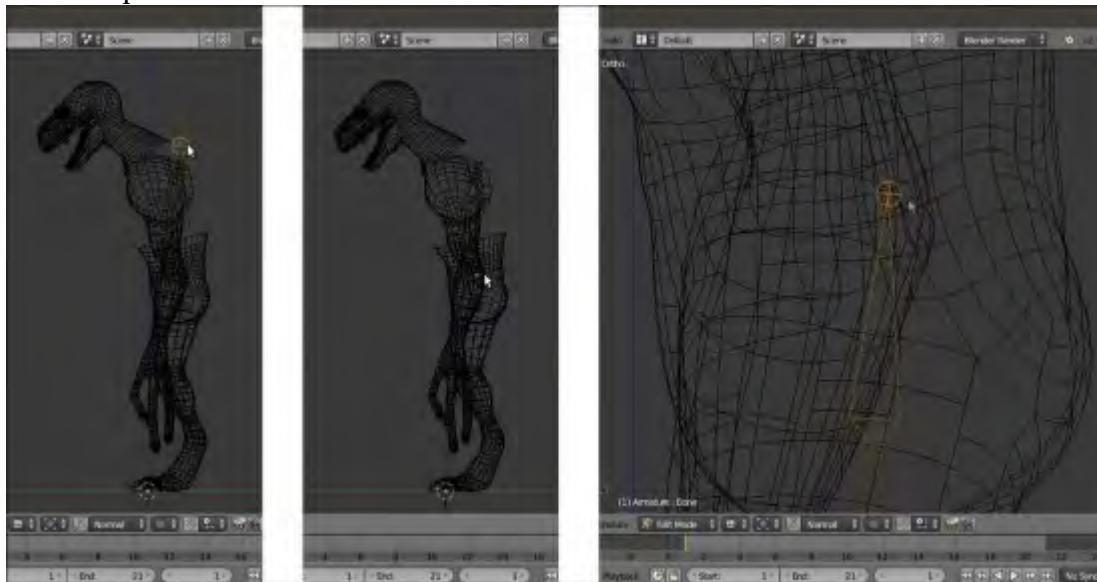
Adding the first Armature's bone

2. Press **Tab** to go into **Edit Mode** and select the whole bone by right-clicking on its **central part**; move the bone upwards to the **Gidiosaurus's** hips area (**G | Z | Enter** or left-click to confirm), and then go in the **Side** view (**3** key on the numpad) and center its position by moving it on the **y** axis:



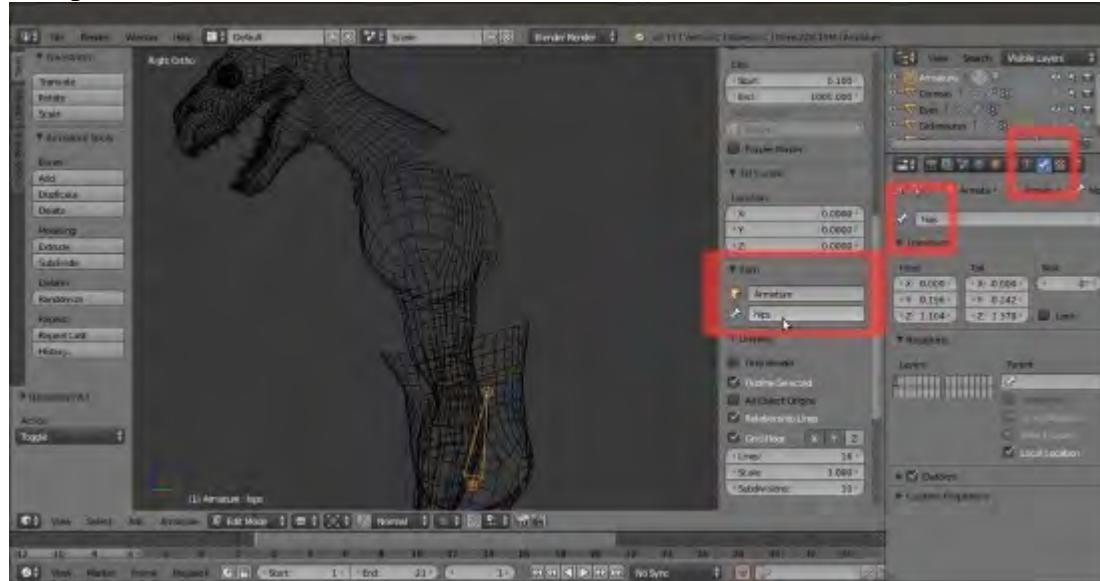
Positioning the bone in Edit Mode

3. Right-click on the **Head** of the bone to select it and by pressing **G** to move it, scale the bone size to fit the pelvis area:



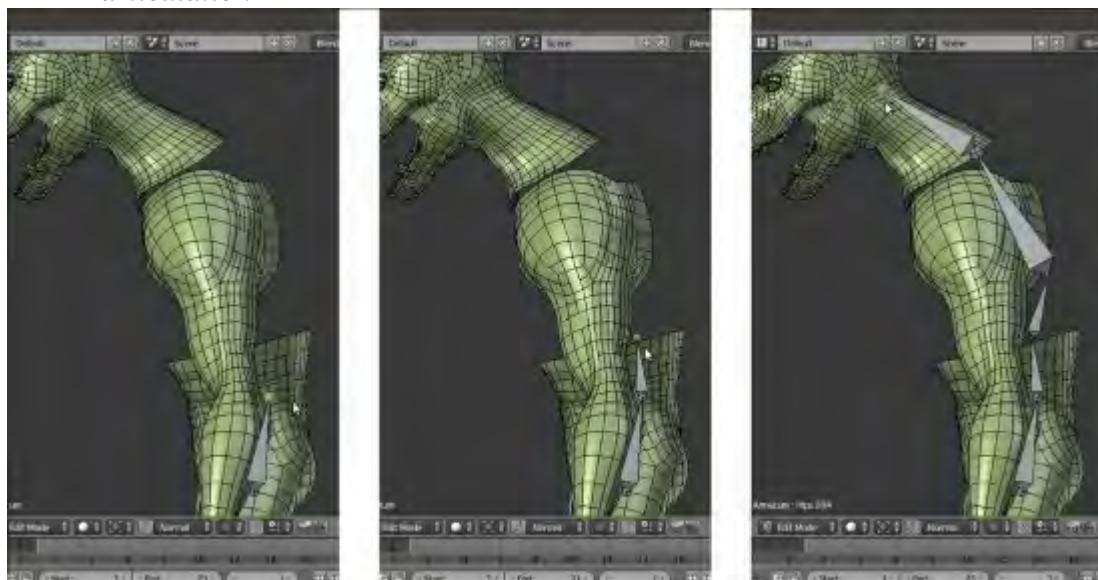
Scaling the bone in Edit Mode

4. Go to the **Item** subpanel under the 3D view **Properties** sidepanel, or in the **Bone** window under the main **Properties** panel to the right-hand side of the screen, and rename **Bone** (default name) as **hips**:



Renaming the bone

5. Press Z to go in the **Solid** viewport shading mode, and then go to the **Object Data** window and enable the **X-Ray** item under the **Display** subpanel.
 6. With the tip of the bone selected (the **Head**), press the **E** key to extrude it. By this process, and by following the wire topology visible on the mesh as a guide, go upwards to build the **Gidiosaurus spine** (2 bones), **chest** (1 bone), and **neck** (1 bone); as much as possible, try to place the **Heads** (the tips/joints) of the bones aligned with the transversal edge-loops on the mesh's *articulation*:



Extruding the bone to build the spine

7. Go again to the **Object Data** window under the **Display** subpanel, and enable the **Names** item (in the following screenshot, all the bones have been selected just to highlight them and their respective names). As you can see in the screenshot, the extruded bones get their names from the previous one, so we have **hips**, then **hips.001**, **hips.002**, and so on:



The bones' names

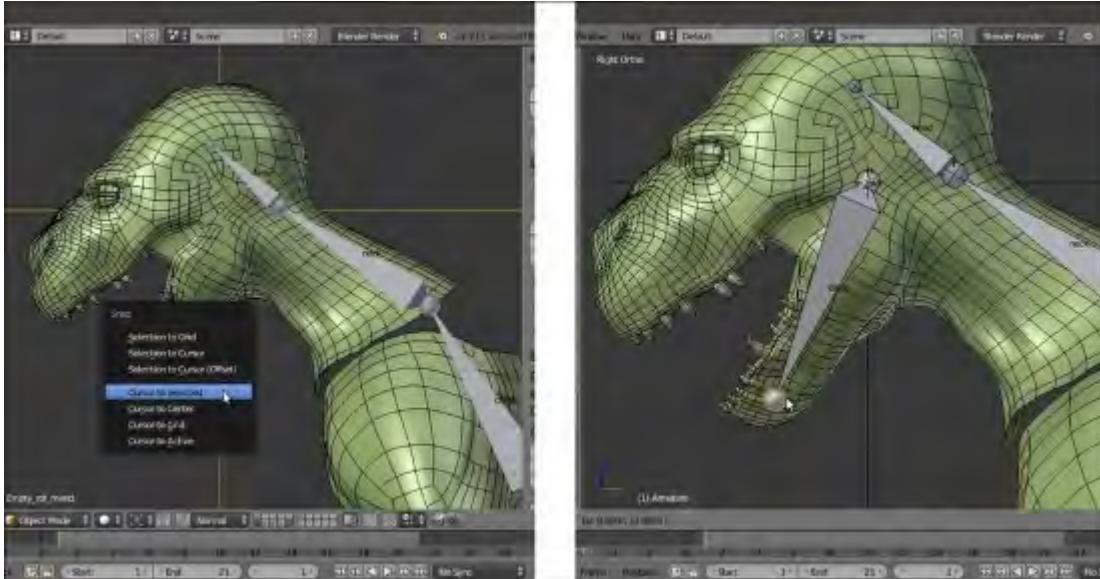
8. Select the **hips.001** bone and rename it **spine.001**; select the **hips.002** bone and rename it **spine.002**.
9. Select the **hips.003** bone and rename it **chest**; select the **hips.004** bone and rename it **neck**.
10. Select the tip of the **neck** bone and extrude it; rename the new bone (**neck.001**) as **head**:



The renamed bones and the head bone

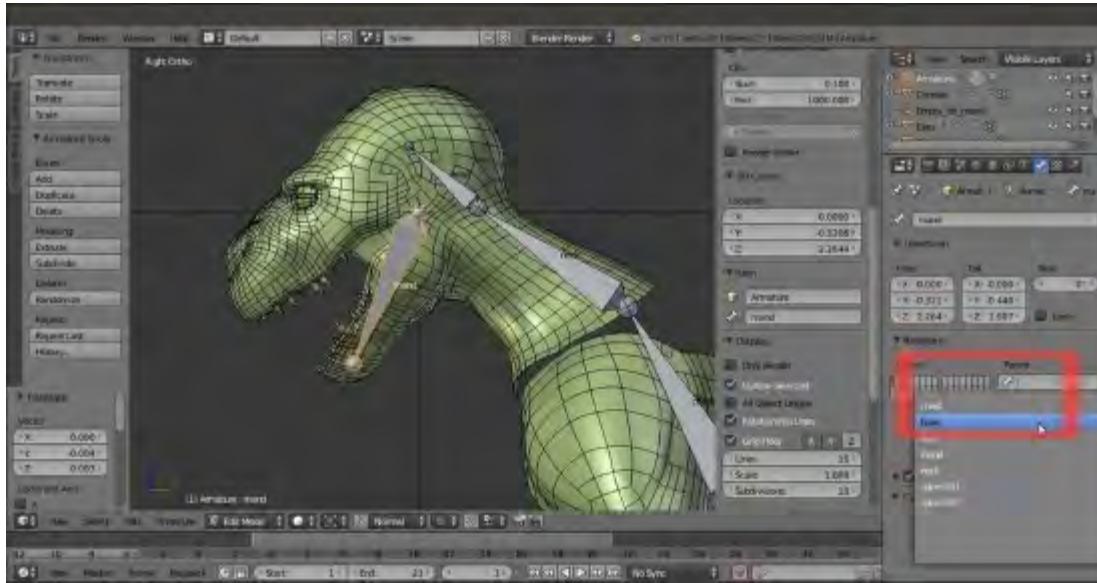
So, now we have built the **spine - neck - head** part of the **Armature**; actually, one thing is still missing: the bone to animate the **mandible**.

11. Press **Tab** to get out of **Edit Mode**. In the **Side** view, enable the **15th** scene layer on the 3D viewport toolbar, in order to show the **Empty_rot_mand** object; select it and press **Shift + S** to call the **Snap** pop-up menu. Then, select the **Cursor to Selected** item.
12. Reselect the **Armature** and go again into **Edit Mode**. Press **Shift + A** to add a new bone; move its **Head** to resize and fit it inside the **mandible** of the **Gidiosaurus**:



The mandible's bone

13. Rename it **mand** and in the **Bone** window under the main **Properties** panel, in the **Relations** subpanel, click on the **Parent** slot to select the **head** item from the pop-up menu with the bones list. Leave the **Connected** item **unchecked**:



The Parent slot and the pop-up menu to select the parent bone

- At this point, we can already see some particular setting to be applied to the bones.
14. Go to the **Object Data** window under the **Properties** panel and in the **Display** subpanel, switch from the default **Octahedral** to the **B-bone** button:



The bones visualized as B-bones

15. Press A to select all the bones, and then press **Ctrl + Alt + S** (or go to the **Armature** item in the window toolbar, and then go to **Transform | Scale Bbone**) and scale the **B-bones** to **0.200** (hold

the **Ctrl** key to constrain the scaling values; the B-bones scaling works both in **Edit Mode** and **Pose Mode**).

16. Select only the **chest** bone and scale it bigger to **2.500**; select the **head** bone and scale it to **4.000**:



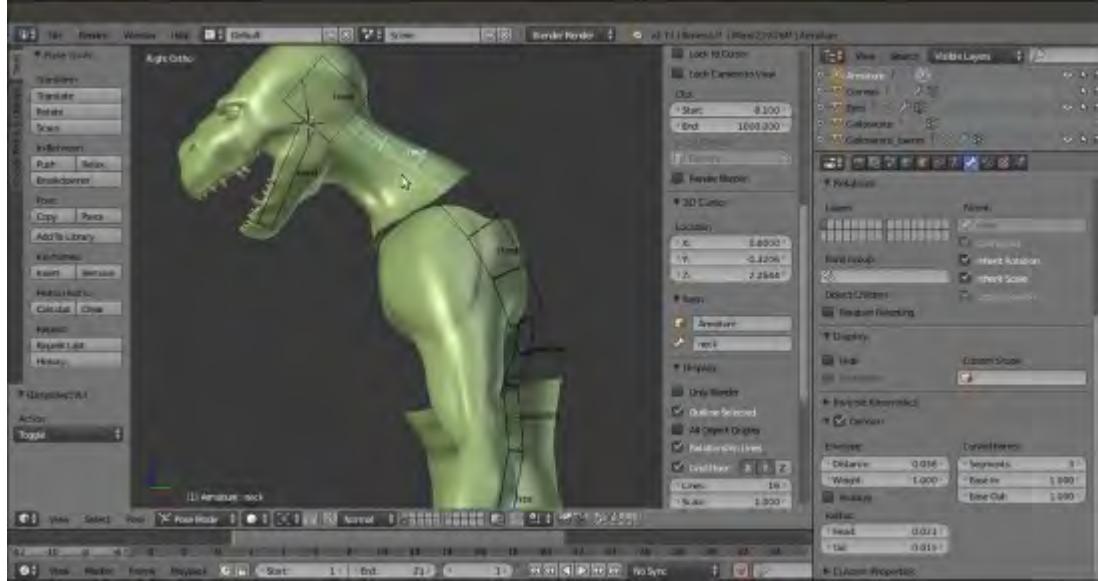
The B-bones scaled for better visualization

17. Go to the **Object** window and under the **Display** subpanel, click on the **Maximum Draw Type** slot (set to **Textured** by default) and switch it to **Wire**.
18. Press **Ctrl + Tab** to switch the **Armature** directly from **Edit Mode** to **Pose Mode**. Right-click on the **chest** bone to select it and go to the **Bone** window under the main **Properties** panel; in the **Deform** subpanel, set **Segments** under the **Curved Bones** item to **3**:



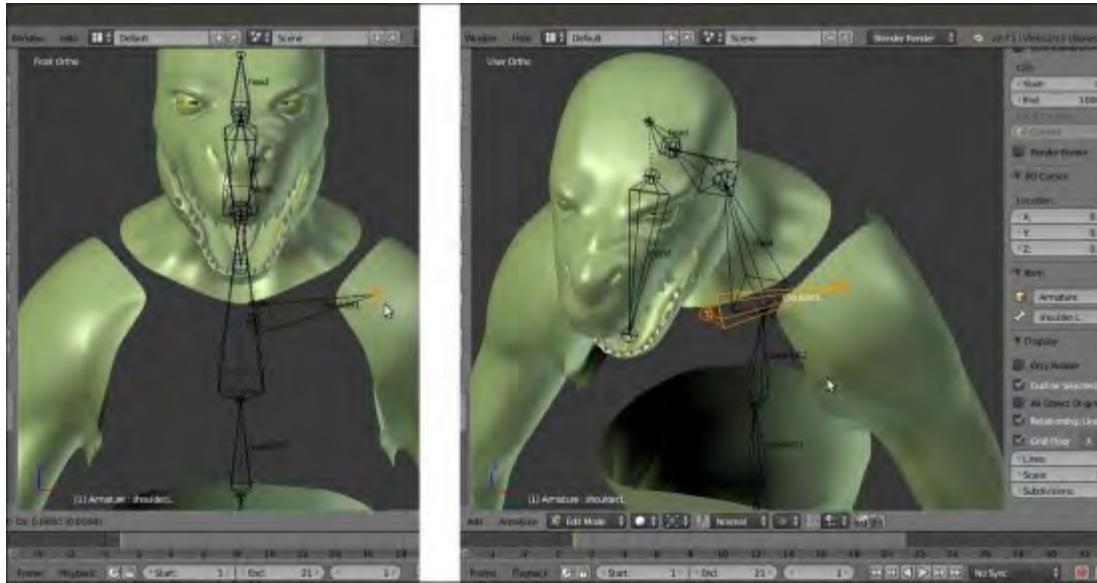
The chest B-bone with 3 curved segments

19. Select the **spine.002** and **spine.001** bones and set **Segments** to **2**. Select the **neck** bone and set **Segments** to **3**.
20. Select the **Gidiosaurus** mesh, go to the **Object** window, and disable the **Wire** item under the **Display** subpanel:



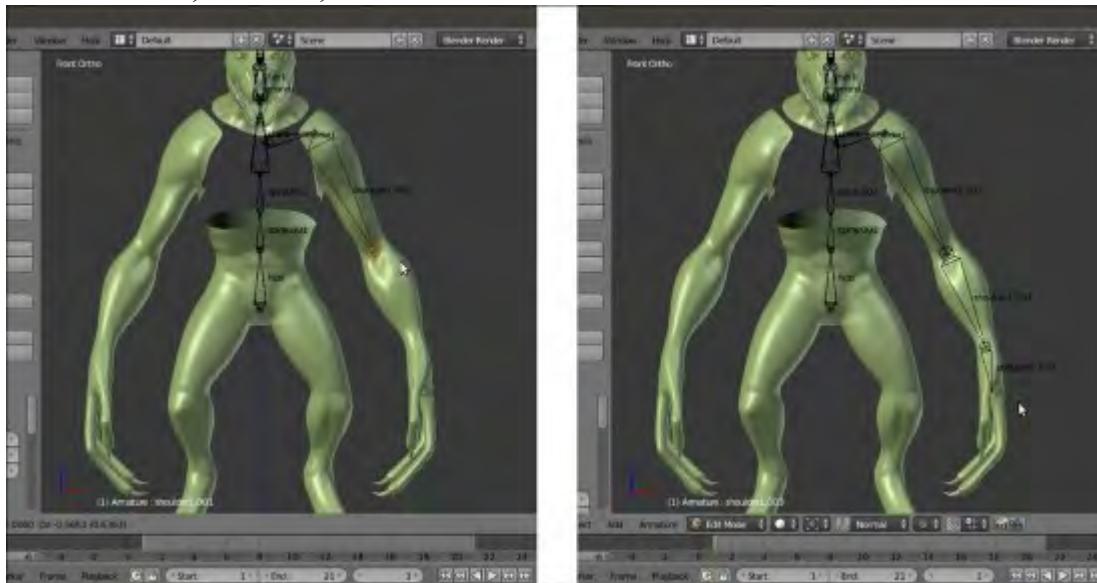
The rig so far

21. Press **Ctrl + Tab** to go out of the **Pose Mode**, and then **Shift + S | Cursor to Selected** to put the **3D Cursor** at the **rig/mesh/center of the scene** pivot point.
22. Press **Tab** to go into **Edit Mode** and press the **1** key on the numpad to go in the **Front** view; go to the **Object Data** window, under the **Display** subpanel, and switch back from **B-bone** to **Octahedral** (even if the visualization mode is different, the bones set as **B-Splines** still keep their *curved* properties in **Pose Mode**).
23. Press **Shift + A** to add a new bone at the cursor position. Move and resize it to put it as the **clavicle** bone—almost horizontal and slightly backward oriented, on the left-hand side of the rig. Rename it **shoulder.L** and in the **Parent** slot under the **Relations** subpanel, select the **chest** item:



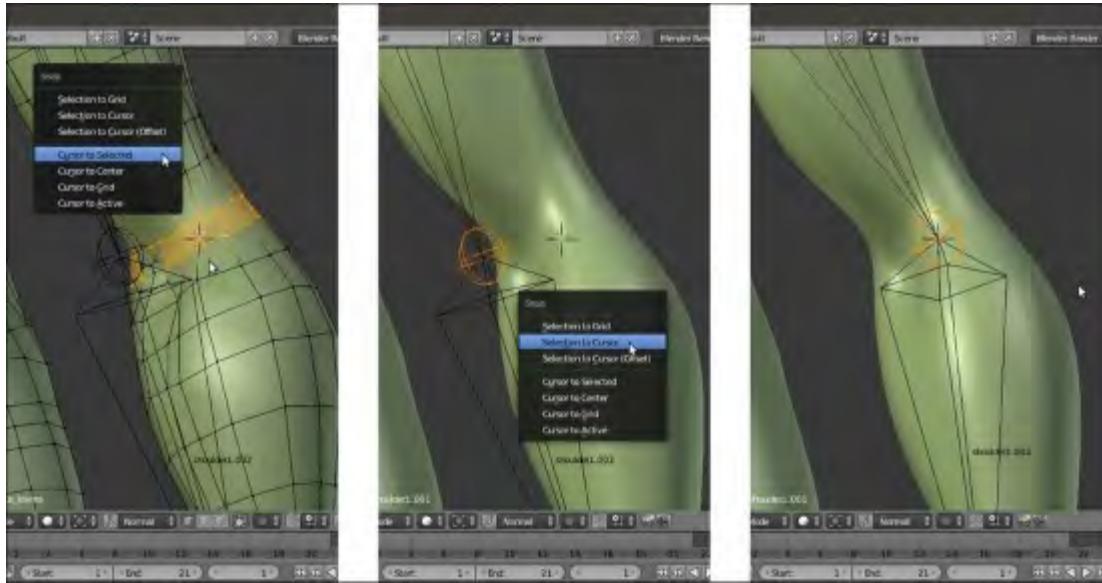
The shoulder.L bone

24. In the **Front** view, select the **Head** of the **shoulder.L** bone and extrude it **3** times to build the bones for **arm**, **forearm**, and **hand**:



Extruding the shoulder.L bone to obtain the skeleton's bones for the arm

25. Now, exit **Edit Mode** and right-click to select the **Gidiosaurus** mesh; enter **Edit Mode**, select one or more edge-loops at the **elbow** level, and press **Shift + S | Cursor to Selected**.
26. Get out of **Edit Mode**, select the **Armature**; go into **Edit Mode**, select the joint between the **arm** and **forearm** bones and press **Shift + S | Selection to Cursor**:



Placing the elbow joint

27. This is the easiest way to correctly align the rig joints with the mesh edge-loops. Do the same for the joint of the **wrist** and the bone of the **hand**; rename the bones as **arm.L**, **forearm.L**, and **hand.L**:



Fixing the position of the wrist joint and hand's bone

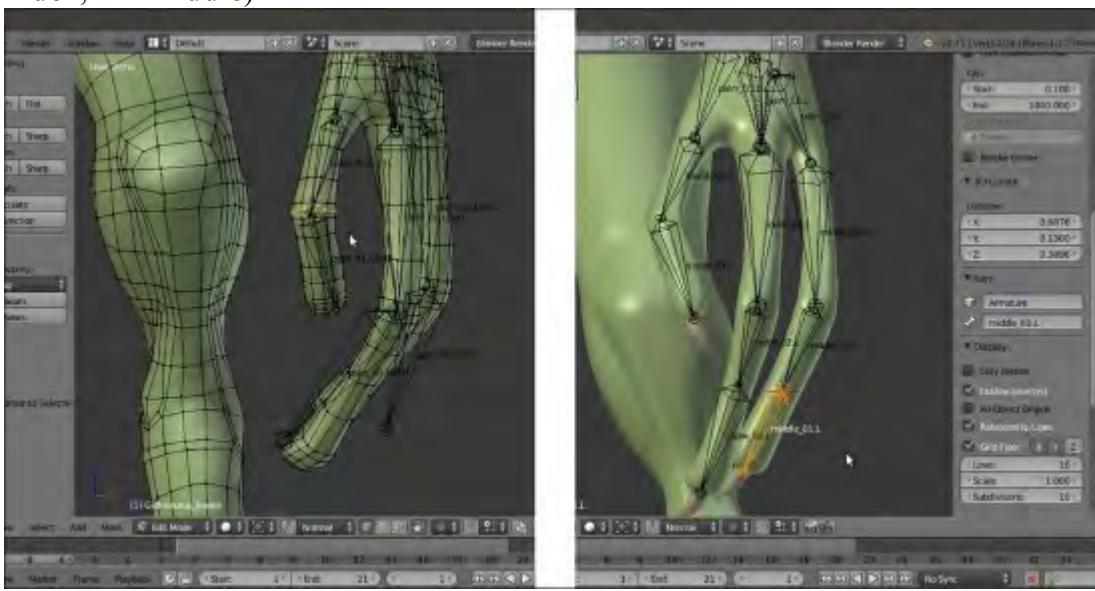
28. Select the **hand.L** bone and use *Shift + D* to duplicate it; scale it smaller (*S | 0.600 | Enter*), rename it **palm_01.L**, and move it above the joining of the **palm** with the **thumb**. Use *Shift + D* to duplicate it 2 more times and move the new bones above the joining of the other two **fingers**; rename them **palm_02.L** and **palm_03.L**.

29. Use **Shift** to select the three **palm** bones and, as the last one, the **hand.L** bone; press **Ctrl + P** | **Keep Offset** to parent them (**not connected**) to the latter one:



Adding the palm bones

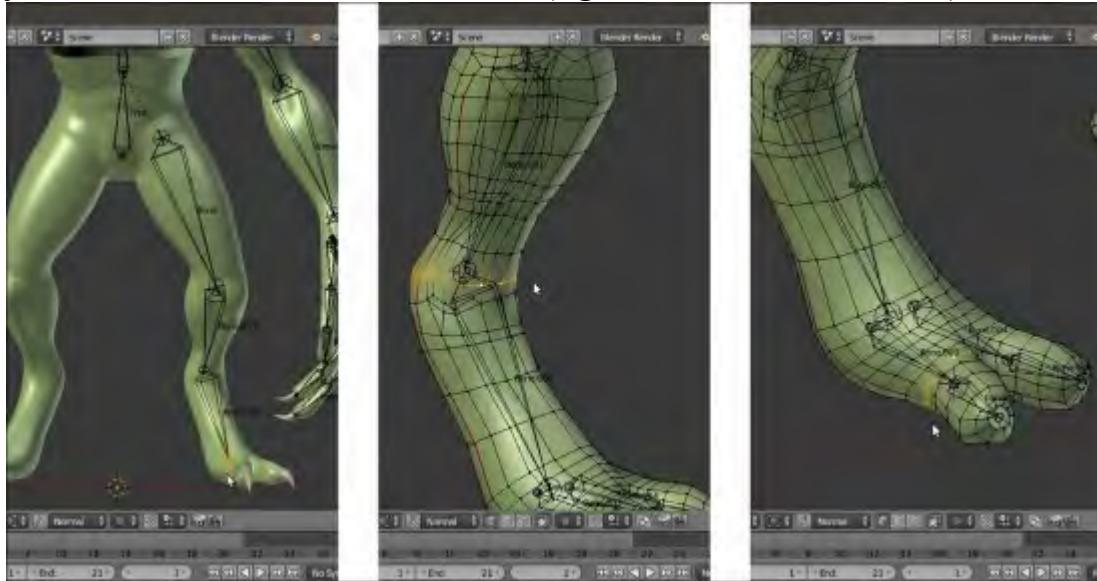
30. Select (individually) the **Heads** of each **palm** bones and extrude the bones for the **fingers**; center their joints with the **3D Cursor/Snap** menu method and rename them properly (**thumb**, **index**, and **middle**):



The bones for the fingers

31. Again with the **3D Cursor** at the rig pivot point, add a new bone and shape it to fit inside the left **thigh**, the **Tail** at the top, close to the **hips** bone, and the **Head** at the **knee** location; select it

- and use *Shift* to select the **hips** bone, and then press *Ctrl + P | Keep Offset*. Extrude the bone's **Head** three times to build the **leg – foot** skeleton.
32. Extrude also the bones for the **toes** and repeat the previously described process to center the joints, and then rename all the new bones (**leg**, **calf**, **foot**, **toe inn**, and **ext**):



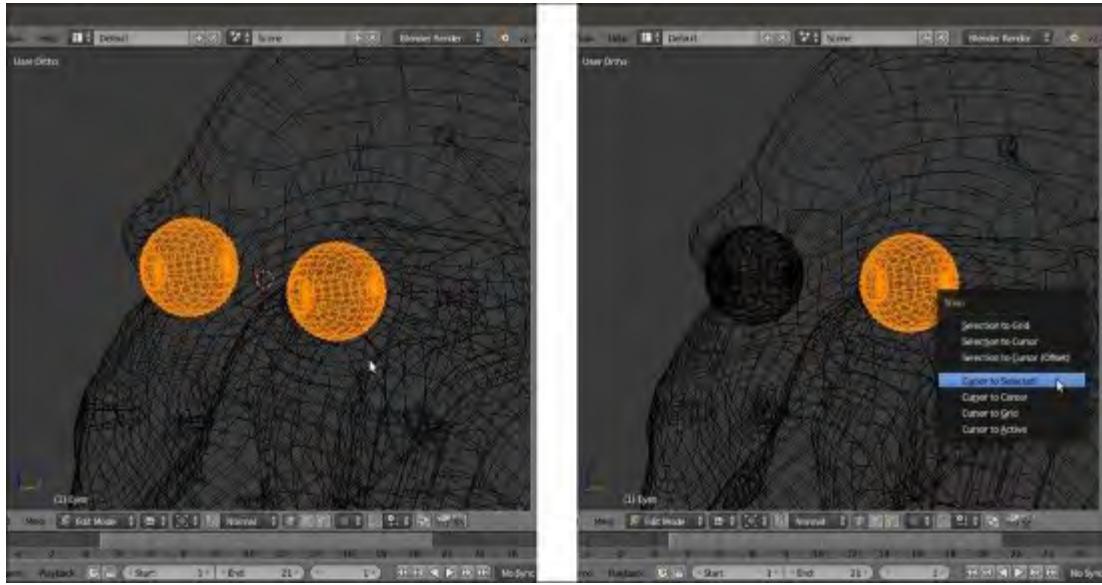
The bones for the leg and toes

- In the preceding screenshots, you can see that we have hidden the **talons** vertices in **Edit Mode** (*H* key), in order to have the possibility to easily select the last edge-loops on **fingers** and **toes**.
33. Save the file as `Gidiosaurus_rig_from_scratch_01.blend`.

Building the rig for the secondary parts

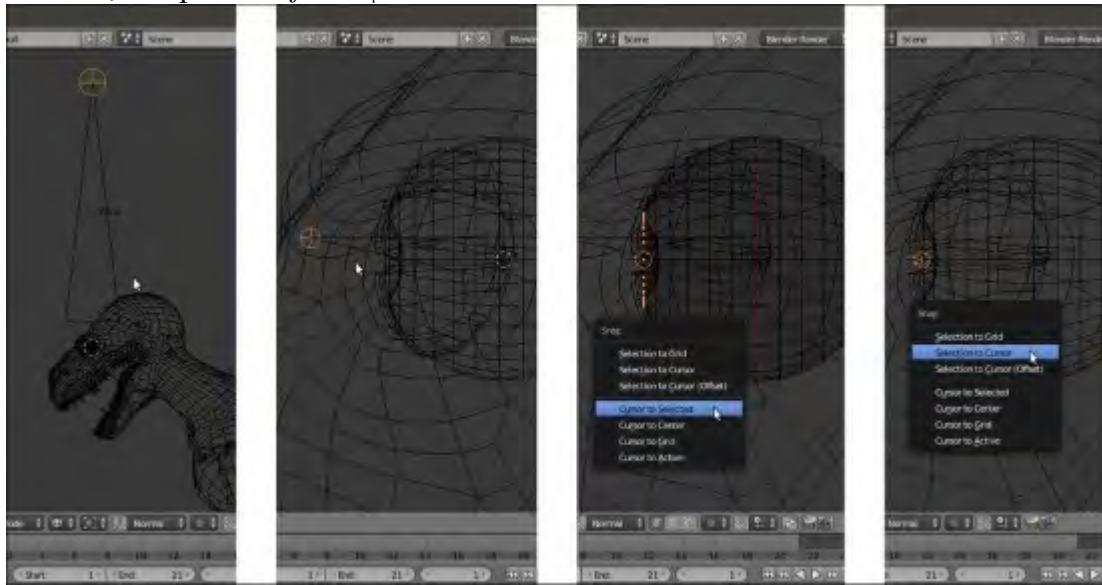
Now that we have completed the main body rigging system, it's time to build the rig for **eyes**, **eyelids**, and **tongue**:

1. Get out of **Edit Mode** and select the **Eyes** item in the **Outliner**; press the dot (.) key on the numpad to center the view on the selected object, the *Z* key to go in **Wireframe** viewport shading mode, and *Tab* to go into **Edit Mode**.
2. Press the *A* key to select all the **eye** vertices and then box-deselect (the *B* key and the middle mouse button) the vertices of the **right eye**; use *Shift + S* to call the **Snap** pop-up menu and select the **Cursor to Selected** item to place the **3D Cursor** at the center of the left eye mesh:



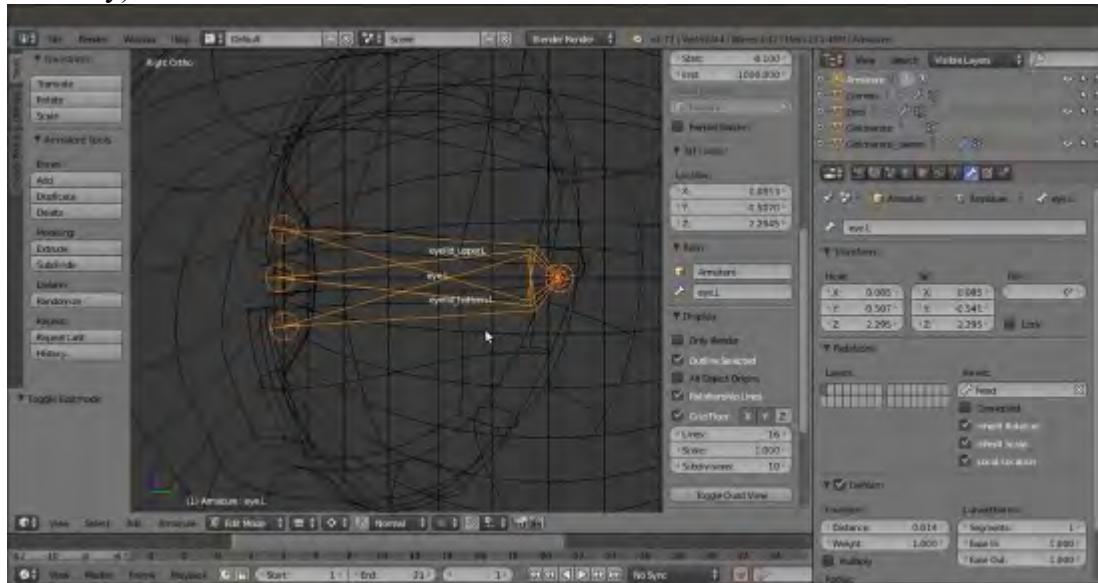
Placing the 3D Cursor at the center of the selected vertices

3. Get out of **Edit Mode**, press the 3 key on the numpad to go in the **Side** view and reselect the **Armature** item in the **Outliner**; press **Tab** to go into **Edit Mode**, and then use **Shift + A** to add a new bone at the cursor position. Press **G** to grab the already selected **Head** of the new bone and move it close to the center of the **eye** to resize it smaller.
4. Get out of **Edit Mode** and select the **Eyes** item; enter **Edit Mode** and deselect all the vertices except for the external last **iris** edge-loop. Then, press **Shift + S** | **Cursor to Selected** and get out of **Edit Mode**.
5. Again, select the **Armature**, go into **Edit Mode**, be sure that the **Head** of the new bone is still selected, and press **Shift + S** | **Selection to Cursor**:



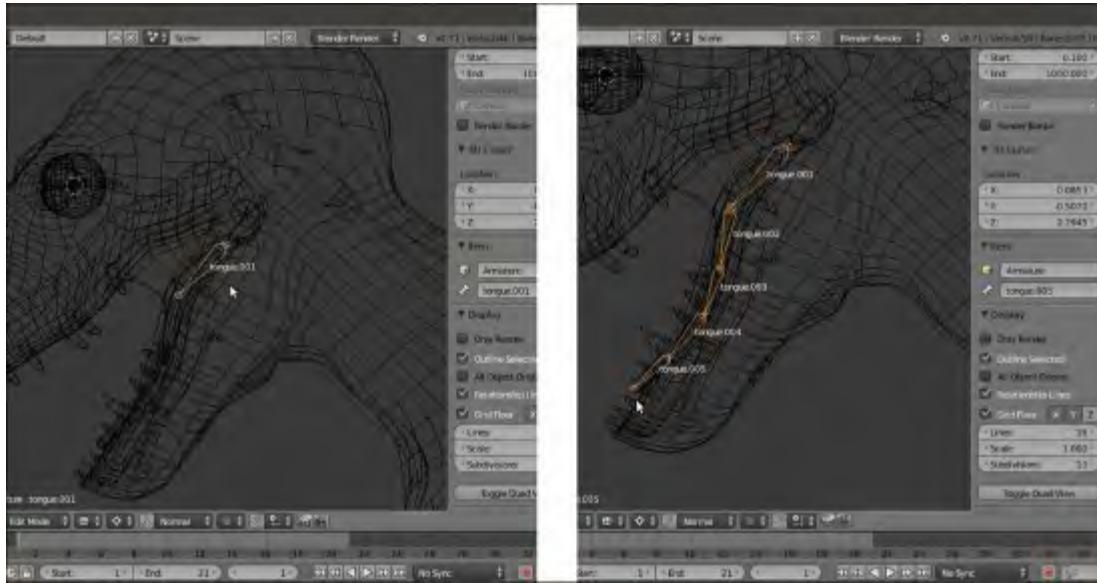
Placing the bone's head at the iris center location

6. Rename the new bone as **eye.L** and in the **Relations** subpanel under the **Bone** window, parent it to the **head** bone (not **Connected**), or use **Shift** to select the **eye.L** and **head** bones and press **Ctrl + P | Keep Offset**.
7. Now, select the **Tail** of the **eye.L** bone and press **Shift + S | Cursor to Selected** to put the **3D Cursor** on it, and then press the period (.) key to switch the **Pivot Point** around the **3D Cursor**; select the whole **eye.L** bone and use **Shift + D** to duplicate it, and soon after, click with the right mouse button to leave the duplicated bone untouched; rotate it **10** degrees clockwise on the cursor position (**Shift + D | right-click | R | X | 10 | Enter**).
8. Rename the new bone as **eyelid_upper.L**.
9. Reselect the whole **eye.L** bone and repeat the duplication procedure; rotate the new duplicate **10** degrees counterclockwise (**Shift + D | right-click | R | X | -10 | Enter**).
10. Rename the new bone as **eyelid_bottom.L** (in the following screenshot, all the three new bones—**eyelid_upper.L**, **eye.L**, and **eyelid_bottom.L**—have been selected just to enhance their visibility):



The bones for the eye and eyelids

11. Now, duplicate the **head** bone, resize it smaller, and move it to the joining of the **tongue** with the **inner mouth**; rename it from **head.001** to **tongue.001** and in the **Relations** subpanel, change its parenting from **neck** to **mand**.
12. Select the **Head** of the **tongue.001** bone and press the **E** key to extrude **4** new bones:



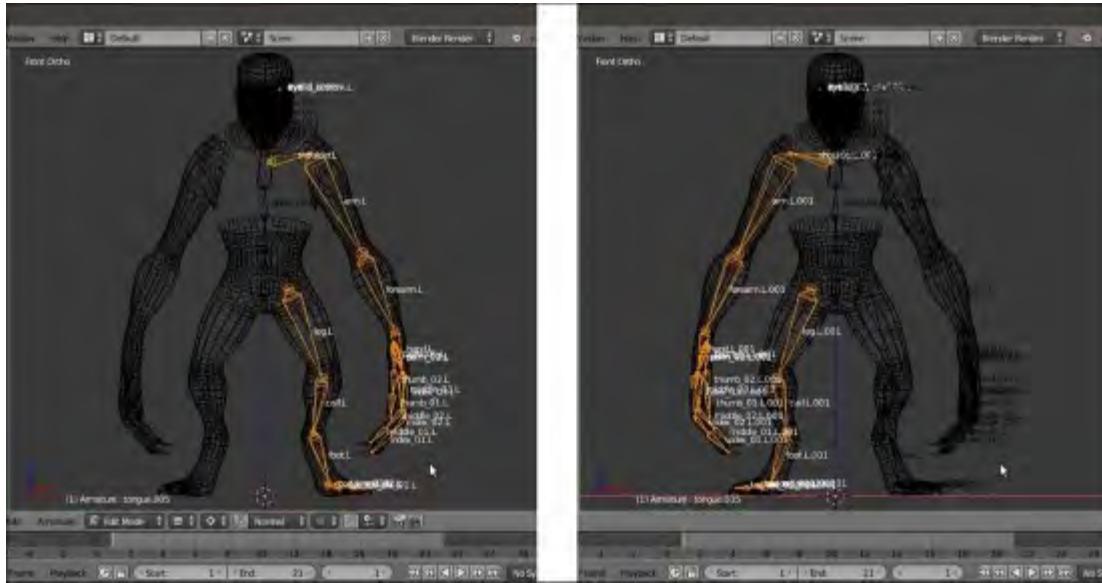
The tongue bones

13. Rename them accordingly, and then use *Shift* to select from the **tongue.001** to **tongue.005** bones and press *Ctrl + R*; move the mouse pointer horizontally to *roll* them on their *y* axis by **180°** (hold *Ctrl* to constrain the rolling to intervals of **5** degrees; alternatively, the roll value can also be set by typing it in the **Roll** button in the **Transform** subpanel under the 3D viewport **Properties** sidepanel).

Completing the rig

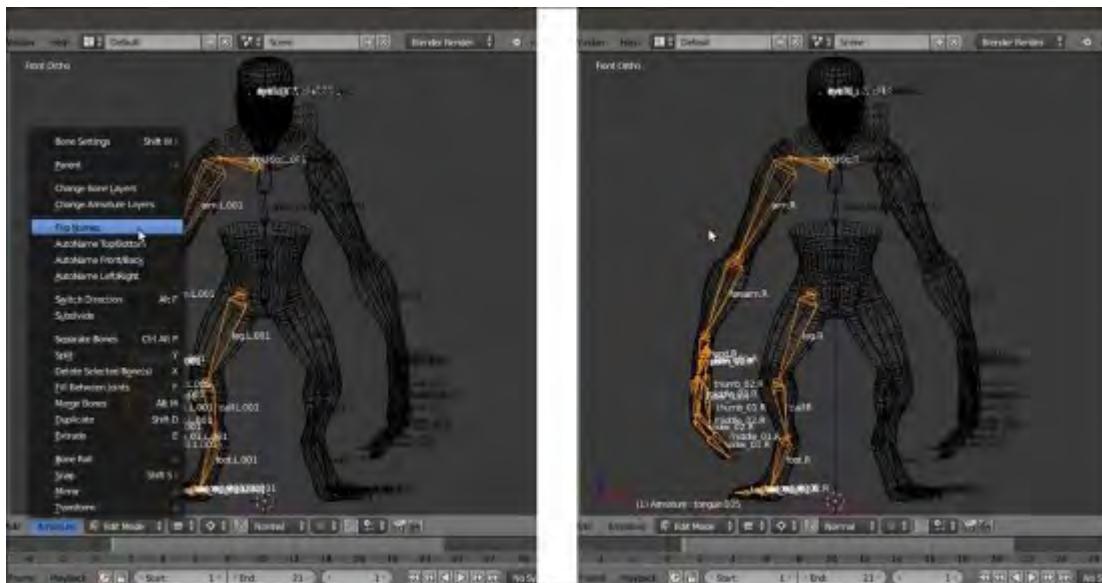
At this point, the basic rig building process is almost done, even if it is only for the left-half part of the mesh:

1. Get out of **Edit Mode** and press *Shift + S* | **Cursor to Selected** to place the **3D Cursor** at the median pivot point of the **Armature**.
2. Go back into **Edit Mode**, select only the left-half part bones and *not* the median ones (meaning: leave the **hips**, **spine**, **neck**, **head**, **mouth**, and **tongue** bones unselected), press *Shift + D* to duplicate them, and then right-click with the mouse button; press *Ctrl + M*, then the *X* key to mirror the duplicated bones on the *x* axis, in order to build the missing right-half part of the rig:



Mirroring the duplicated bones on the x axis

- With the duplicated bones still selected, go to the **3D window toolbar** and click on the **Armature** item; in the pop-up menu, select the **Flip Names** item to automatically rename them with the correct **.R** suffix:



Renaming the suffix of the duplicated bones

As a very last thing for this recipe, we must verify that the alignment of the bones, especially the last duplicated ones, is correct and, just in case, recalculate the roll rotation, that is, the rotation around the *y* axis of the bone itself.

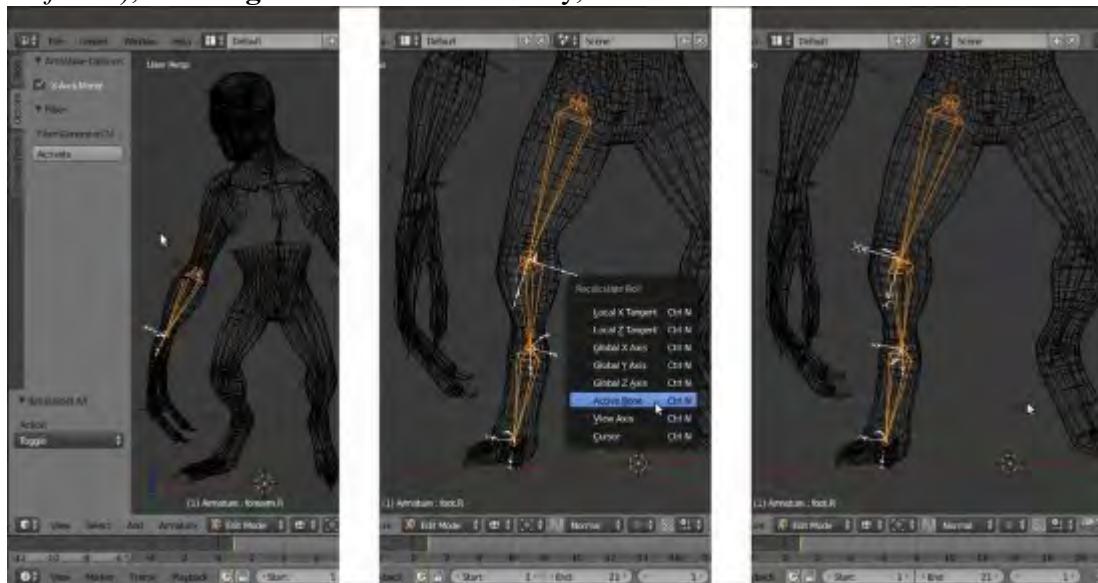
- In the **Object Data** window, under the **Display** subpanel, check the **Axes** item to make the bones orientation axes visible (only in **Edit Mode** and **Pose Mode**) in the 3D view.
- Select all the bones and press **Ctrl + N** to recalculate the rolling of all of them; in the **Recalculate Roll** pop-up menu, there are several different options: because basically the **z** axis of the bones must match from the left to the right side of the whole rig, with the **Armature** (and the mesh) oriented along the **y** global axis, as in our **Gidiosaurus** case, the first top item, **Local X Tangent**, can be a good start.

By the way, it is good practice to not trust this automated procedure alone, because sometimes it can give inconsistent results; so, do the following:

- After the recalculation, check that the axes of each bone are actually correctly orientated in a consistent way; effectively, there are some bones that didn't get consistently oriented, meaning that their **x** and **z** local axes are oriented differently from the other bones.
- In this case, select the incorrectly oriented bone, press **Ctrl + R**, and move the mouse to change the rolling; press the **Ctrl** key to constrain the rolling to intervals of **5** degrees. Alternatively, select the wrong bones, and then use **Shift** to select one bone that is correctly oriented and press **Ctrl + N | Active Bone** to copy the rolling from the last selected bone.

By enabling the **X-Axis Mirror** item in the **Armature Options** tab under the **Tool Shelf**, you can recalculate only the bones of one side; the other side bones will follow automatically.

If you want to make sure the bones' orientations are correct and everything is going to work in animation, just go into **Pose Mode** and rotate one bone, for example **leg.L**, and then click on the *Copies the current pose of the selected bones to copy/paste buffer button (**Ctrl + C**)*, which is the first left one of the last three buttons to the right-hand side of the viewport toolbar; then, select the symmetrical bone, **leg.R**, and click on the last right button to paste the flipped pose (**Ctrl + Shift + V**); if the **leg.R** bone rotates correctly, then the orientation is OK:



Recalculating the roll of incorrectly oriented bones

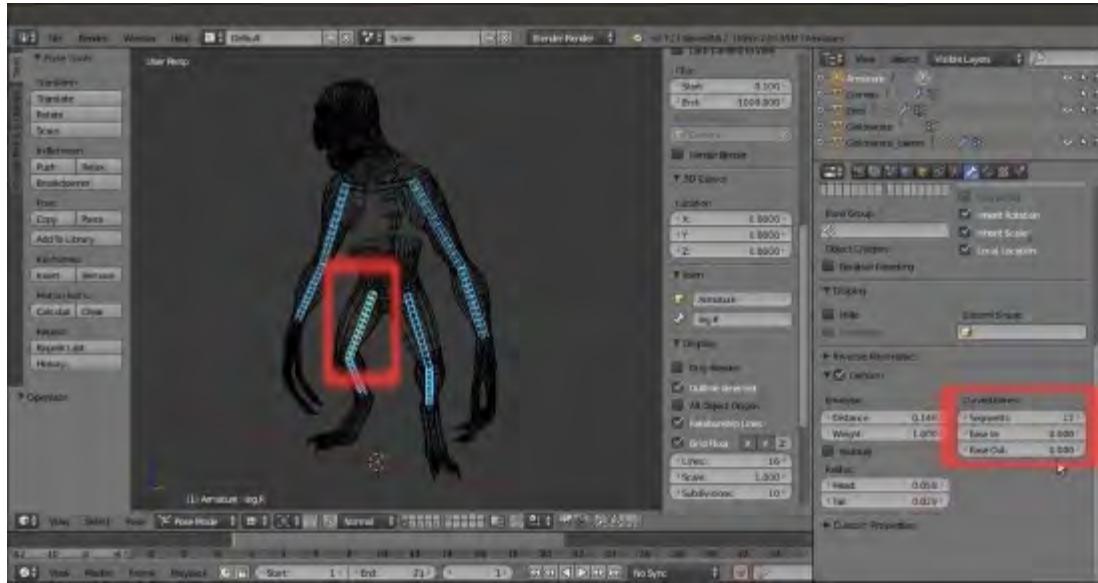
- Now, go to the **Object Data** window under the main **Properties** panel and under **Display**, switch again the bones visualization from **Octahedral** to **B-bone**; select the bones and by pressing **Ctrl + Alt + S**, scale the **B-bones** smaller or bigger, depending on the visual effect you want to obtain:



The almost completed Armature in B-bones visualization

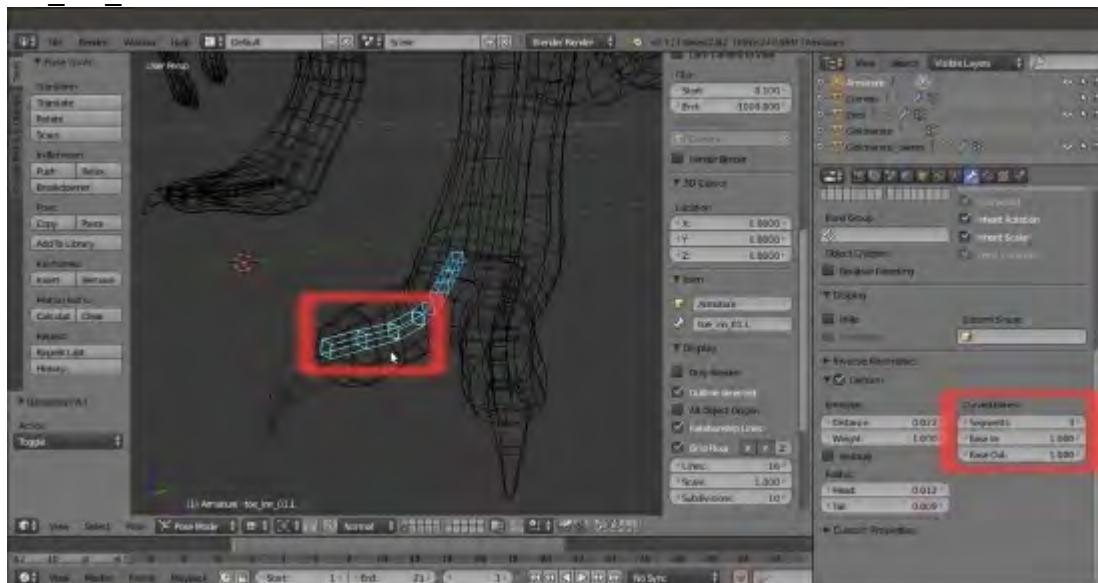
- Press **Ctrl + Tab** to pass directly from **Edit Mode** to **Pose Mode** and select the **forearm.L** bone; in the **Deform** subpanel under the **Properties** panel, set the **Segments** for the **Curved Bones** to **12**, and then set the **Ease In** and **Ease Out** values to **0.000**.
- Repeat this for the **forearm.R** bone and also for the **calf.L** and **calf.R** bones; repeat also for the **arm.L**, **arm.R**, **leg.L**, and **leg.R** bones.

In the following screenshot, all the eight **B-bones** have been selected to make them more visible. By the way, the highlighted **leg.R** bone is the active one and shows the **Curved Bones** setting in the highlighted **Deform** subpanel to the right-hand side of the screen.



The Segments setting for the leg bone

11. Select the **toe_inn_02.L** bone and in the **Deform** subpanel under the **Properties** panel, set the **Segments** to **6** and leave the **Ease In** and **Ease Out** values to **1.000**.
12. Repeat this for the **toe_ext_02.L** bone; then, do the same also to the **toe_inn_02.R** and **toe_ext_02.R** bones.
13. Select the **toe_inn_01.L** bone and set the **Segments** to **3**; leave the **Ease In** and **Ease Out** values to **1.000**.
14. Repeat for the **toe_ext_01.L** bone; then, do the same also to the **toe_inn_01.R** and **toe_ext_01.R** bones:



The Segments setting for the toes bones

15. Save the file.

How it works...

Although it's often a really time consuming task, the handmade rigging is quite self-explicative; it is, however, better to explain some of the concepts behind this.

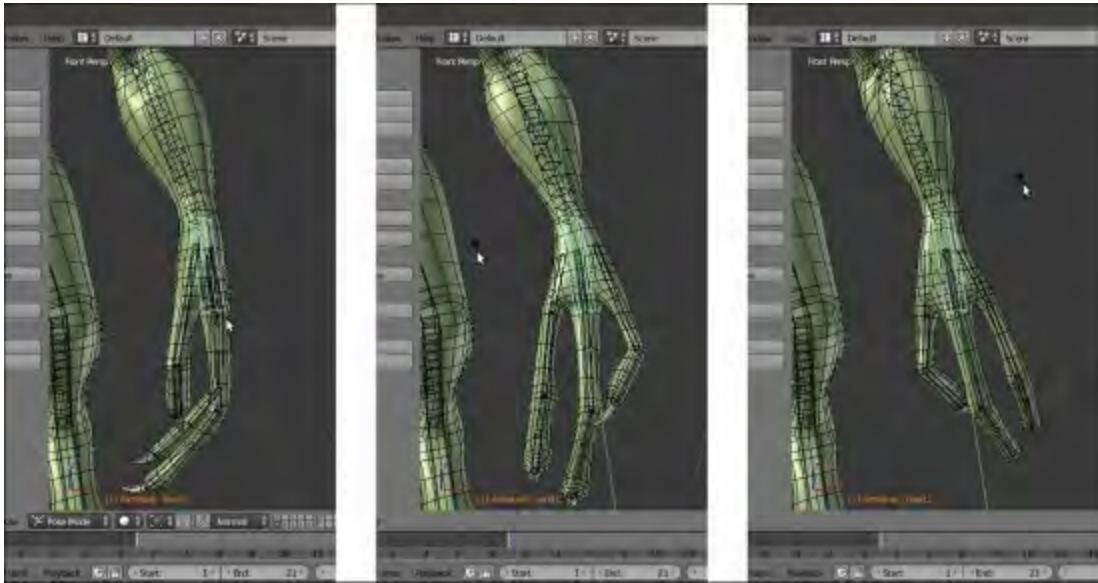
The proper renaming of the bones is important, considering that each deforming bone will affect a vertex group sharing the same name on the mesh; although in some cases, as for the **tongue** bones, the bone naming process can be automated in some way, usually it is better to spend time in giving meaningful names to each bone, in order to avoid mistakes in the following skinning process.

It's also very important to build the hierarchy of the bones so that a bone at a higher level can lead all of the children bones, as it would be in a real skeleton (that is, for example, the **hand** bone leads all the **fingers** bones, the **forearm** bone leads the **hand** bone, and so on).

Parenting a bone and then obtaining the others by extruding and/or duplicating simplifies the work because an extruded bone is automatically parented to the bone it has been extruded from, and a duplicated bone obviously inherits the parenting of the original one; in the case of the **tongue.001** bone, extruding the others has given us a chain with bones automatically parented and named as **tongue.002**, **tongue.003**, **tongue.004**, and **tongue.005**.

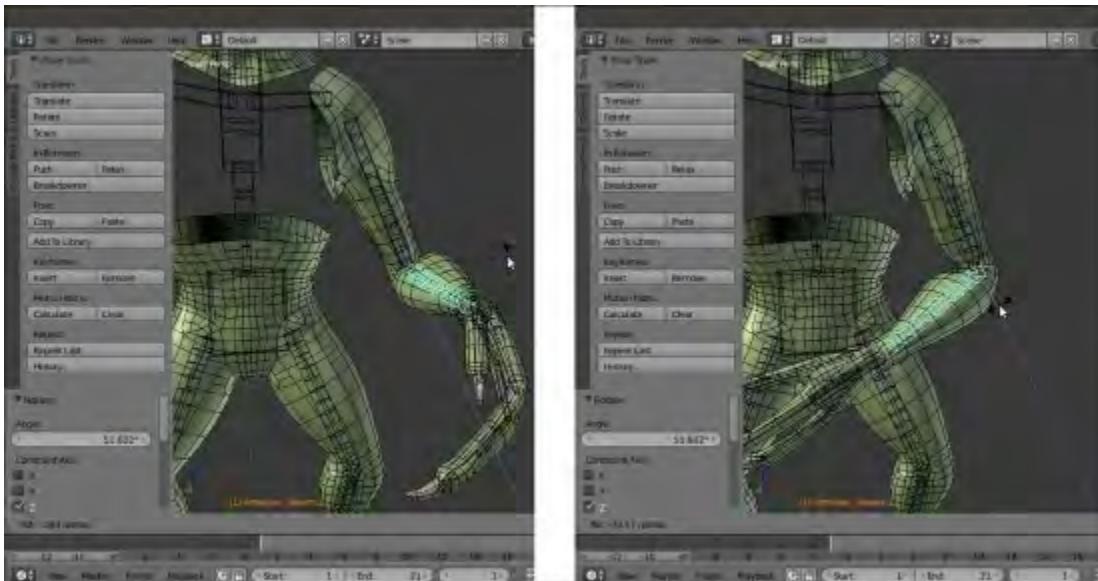
B-bones are both a visualization mode for the bones and a way of working; **B-bones**, in fact, can work inside a chain as splines, which means that the bones are curved according to the number of **Segments** and the values of the **Ease In** and **Ease Out** items. For the bones of the **arms** and **legs**, we have set the **Ease In** and **Ease Out** values to **0.000** (default is **1.000**; maximum is **2.000**), in order to have the B-bones rotating only on their **y** axis but remaining straight along their length, and hence, mimic the twisting by not only the rotation (*pronation* and *supination* of the lower arm) of both the *Ulna-Radius* and *Tibia-Fibula* articulation complexes, but also the (limited) rotation of *Femur* and *Humerus*.

In some way, **B-bones** can work as a kind of simulation for a very basic muscle system; in the following screenshot, you can see their effect on the skinned mesh for the **forearm** by rotating the **hand.L** bone on the local **y** axis (to enhance the visibility of the mesh surface's modifications, the *wireframe over solid drawing* item has been enabled in the **Display** subpanel under the **Object** window):



The effect of the rotation of the hand bone on the forearm B-bone and skinned mesh

Here is the effect of the rotation of the **forearm.L** bone on the **Gidiosaurus** high arm:



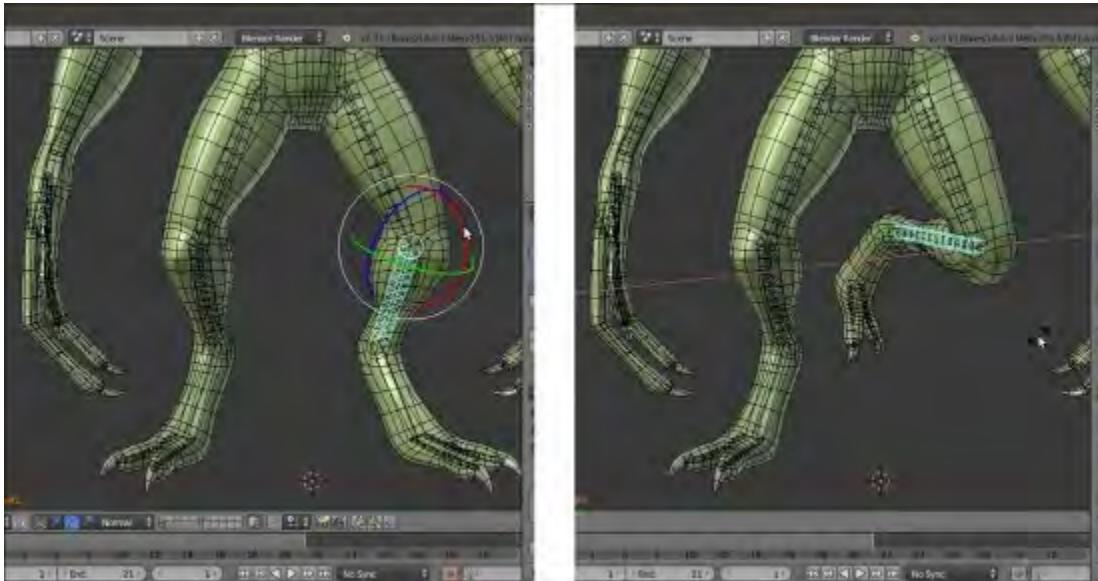
The effect of the rotation of the forearm bone on the upper arm b-bone and skinned mesh

The effect acts on the **shin** as well, by rotating the **foot.L** bone on the global **z** axis:



The same effect obtained on the calf b-bone by rotating the foot on its local y axis

Also, the same effect acts on the **thigh** by rotating the **calf.L** bone:



The same effect obtained on the leg b-bone by rotating the calf

Note that the **Gidiosaurus** is a **digitigrade biped humanoid**: the bones that, from our plantigrade point of view, look like the **foot** are actually the **toes**, while the almost vertical structure that we would call an

ankle is the real **foot** (this is a very common condition among the majority of the terrestrial animals, both still alive and extinct).

Perfecting the Armature to also function as a rig for the Armor

So, in the previous recipe, we have built the body deforming **Armature** for the **Gidiosaurus** character.

However, the **Gidiosaurus** is an (almost) evolved and a civilized creature, and being also a warrior, it wears a metallic **Armor**; this armor will need to be later parented to the rig as well in order to be animated.

Some of the bones that we have already created will be perfect to skin the **Armor** object too, by assigning the right vertex group to the right mesh part (for example, the **head** vertex group for the **Helm** or the **chest** vertex group for the **Breastplate**). However, because the **Armor** is made also by different parts that cannot be simply driven by the already existing bones (for example, the **belts**, **Vambraces**, and especially **Groingroup**), some modification and/or addition to the rig must be done anyway.

Getting ready

Start from the previously saved `Gidiosaurus_rig_from_scratch_01.blend` file:

1. Enable the **13th** scene layer to show the **Armor** object.
2. Select it and go to the **Object Modifiers** window under the main **Properties** panel. Expand the **Subdivision Surface** modifier tab and click on the *Display modifier in viewport* button, the one with the eye icon, to disable it.
3. Go to the **Outliner** and click on the arrow icon to the side of the **Armor** item to make it unselectable.
4. Click on the arrow icon to the side of the **Gidiosaurus_lowres** item to make it unselectable as well.
5. Save the file as `Gidiosaurus_rig_from_scratch_02.blend`.

How to do it...

Let's start by adding bones dedicated to the **Armor**:

1. Go into **Edit Mode** and select the **forearm.L** bone; use *Shift + D* to duplicate it and rename it **vanbrace.L**. Press *M* and in the **Change Bone Layers** pop-up, click on the **2nd** button to move the duplicated bone to that bone layer.
2. Do the same for the **forearm.R** bone (**vanbrace.R**) and for the **calf.L** (**greave.L**) and **calf.R** bones (**greave.R**).
3. Now, go to the **Object Data** window and click on the **2nd** button under the **Layers** item in the **Skeleton** subpanel, in order to show only the four duplicated bones in the 3D viewport; press *Tab* to get out of **Edit Mode**.
4. Select the **vanbrace.L** bone and go to the **Bone** window under the **Deform** subpanel; under the **Curved Bones** item, set back the **Segments** and **Ease In** and **Ease Out** values to default, that is, **1, 1.000** and **1.000**.
5. Go back into **Edit Mode** and click on the **Connected** item under the **Relations** subpanel.

6. Get out of **Edit Mode** and go to the **Bone Constraints** window under the main **Properties** panel (not to be confused with the **Object Constraints** window); click on the **Add Bone Constraint** button and select a **Copy Rotation** constraint from the pop-up menu (the bone turns light green, in order to show that it has a constraint assigned now).
7. In the **Target** field, select **Armature**; in the **Bone** field, select the **forearm.L** item; in the **Space** fields, select **Pose Space** for both.

Alternatively, for steps 6 and 7, select the **forearm.L** bone and then use *Shift* to select the **vanbrace.L** bone. Hence, press *Shift + Ctrl + C* to call the **Add Constraint (with Targets)** pop-up menu and select the **Copy Rotation** item. This will automatically add the **Copy Rotation** constraint to the **vanbrace.L** bone, with the first selected bone (**forearm.L**) as a target; the other setting must be enabled and/or tweaked in the constraint subpanel instead.

8. Click again on the **Add Bone Constraint** button and this time, select an **Inverse Kinematics** constraint (the bone turns yellow, in order to show that an **IK solver** has been assigned). In the **Target** field, select the **Armature** item, in the **Bone** field, select the **hand.L** bone, and set the **Chain Length** to 1; deselect **Stretch** and select **Rotation**, lowering the weight to the minimum (that is **0.010**):



The constraints assigned to the forearm.L b-bone

9. Repeat the steps from 4 to 8 for the other three duplicated bones (obviously, setting the appropriate bones as targets for each pair of constraints; the target bone for the **IK** constraint assigned to the **greave** bones is the respective foot bone).

The rig can now drive the **vambraces** and **greaves**; let's see the **knee guards** and **Groinguard**.

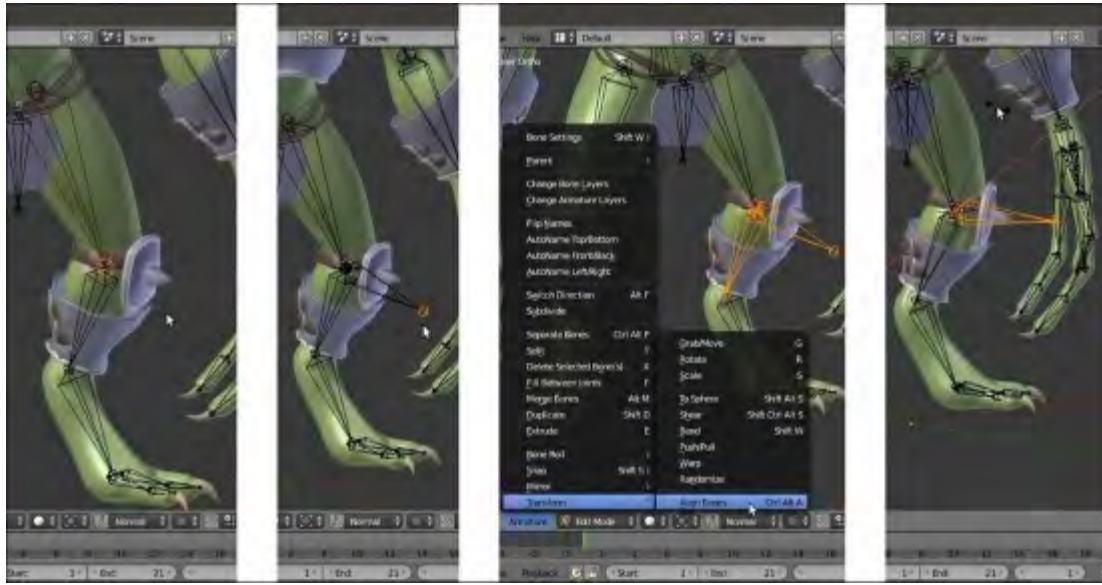
10. First, switch the **Armature** visualization back to **Octahedral**, then go into **Edit Mode**, select the **hips** bone and use *Shift + D* to duplicate it; in the **Side** view, rotate the duplicate 170 degrees, then move it on the **Groinguard** part of the armor, in order to have the **Head** of the bone placed to the joint of the **plate** with the **ties**; select the **Tail** of the **groinguard** bone and scale it smaller to fit the part.

11. To position the bone more precisely, go to the **Transform** subpanel under the **Properties 3D** view sidepanel and set the following values for the **Head** (of the bone): **X = 0.001**, **Y = 0.020**, and **Z = 1.147**; for the **Tail** set the following values: **X = 0.001**, **Y = 0.022**, and **Z = 0.873**.
12. Go to the **Item** subpanel and rename the bone from **hips.001** to **groinguard**:



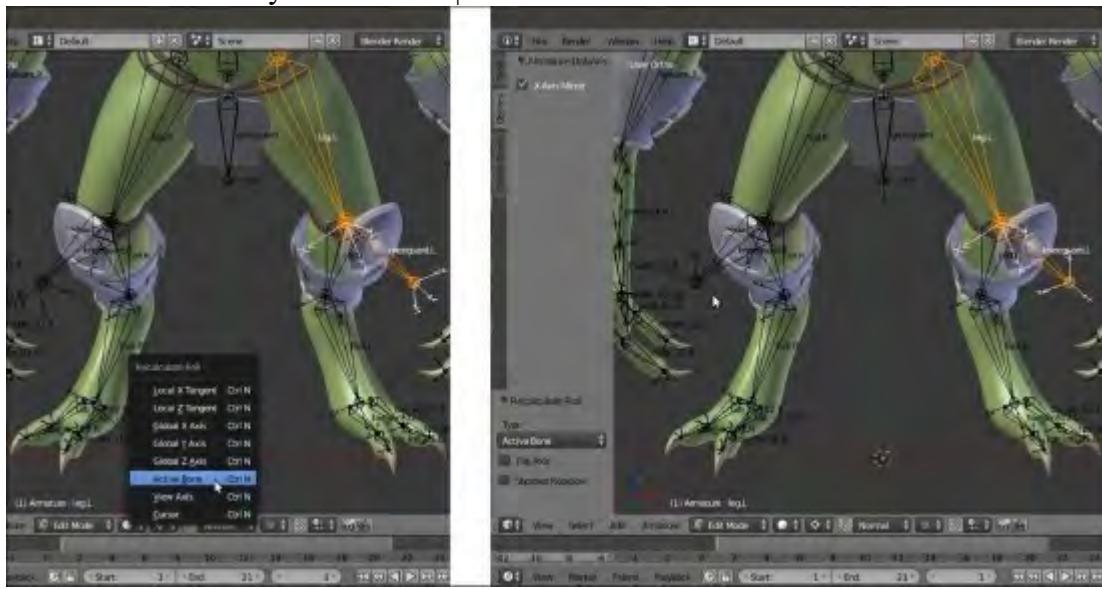
The groinguard bone

13. Go to the **Bone** window in the main **Properties** panel, and under the **Relations** subpanel, click on the **Parent** empty slot to select the **hips** item.
14. Now, select the joint of the **leg.L** bone with the **calf.L** bone and press **Shift + S | Cursor to Selected**; press **Shift + A** to add a new bone and rescale it smaller.
15. Select the whole new bone and use **Shift** to select the **calf.L** bone. Then, go in the 3D view toolbar and click on the **Armature** item; go to **Transform | Align Bones** (or else, press the **Ctrl + Alt + A** keys) to align the new bone as the **calf.L** one.
16. Enable the widget (**Ctrl + spacebar**), set the **Transform Orientation** to **Normal**, and the rotation pivot on the **3D Cursor**. Then, rotate the new bone **110** degrees on the normal **x** axis (the red wheel of the widget, or else **R | X | X | 110 | Enter**):



Aligning the new bone

17. Go into **Object Mode** and press **Shift + S | Cursor to Selected** to place the **3D Cursor** at the median pivot point of the **Armature**; go back into **Edit Mode**, press **Shift + D** to duplicate the new bone, then **Ctrl + M | X** to mirror it on the other side.
18. Rename the new bones as **kneeguard.L** and **kneeguard.R**; enable the axis visibility and recalculate the roll by the **Ctrl + N | Active Bone** tool:



Recalculating the roll angle of the kneeguard bones

19. Parent the **kneeguard.L** bone to the **leg.L** bone and the **kneeguard.R** bone to the **leg.R** one (not connected).

20. Select the **groeguard** bone and use *Shift + D* to duplicate it, and then scale the duplicated bone a little bit bigger and rename it as **groeguard_ctrl**; uncheck the box of the **Deform** subpanel under the **Bone** window:



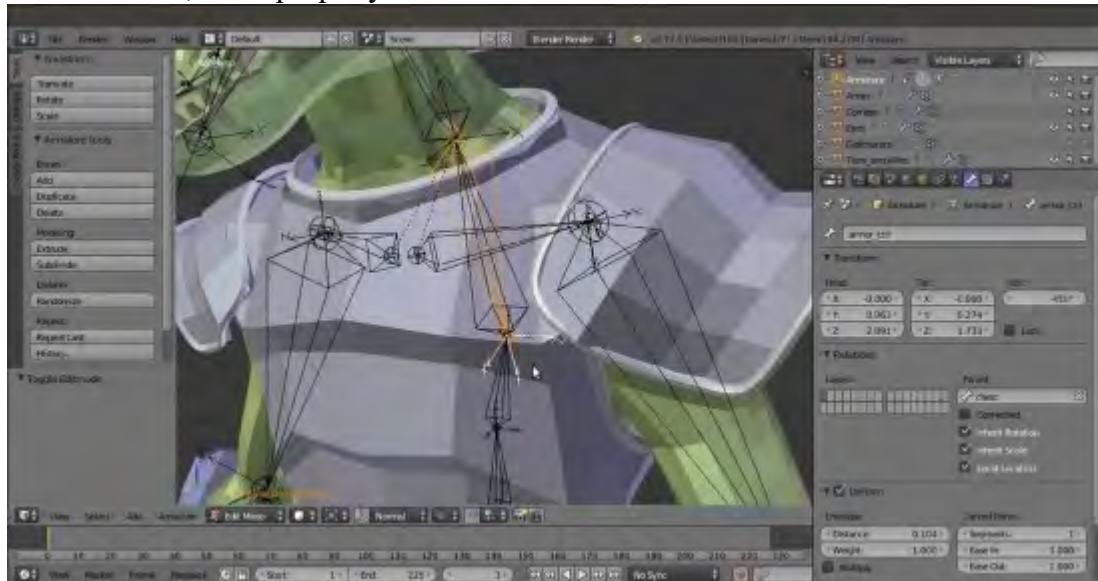
Creating a control bone for the groeguard bone

21. Select the **groeguard** bone, go to the **Relations** subpanel, and click in the **Parent** field to select the **groeguard_ctrl** bone.
22. Get out of **Edit Mode** and in **Pose Mode**, select the **groeguard_ctrl** bone.
23. Go to the **Bone Constraints** window under the main **Properties** panel; click on the **Add Bone Constraint** button and select a **Locked Track** constraint from the pop-up menu.
24. In the **Target** field, select the **Armature** item; in the **Bone** field, select the **kneeguard.L** item. Set the **Head/Tail** value to **0.500**: **To** (*Axis that points to the target object*) = **-X** and **Lock** (*Axis that points upward*) = **Y**. In the **Constraint Name** field, rename it as **Locked Track.L**.
25. Add a new **Locked Track** constraint and repeat everything as in the previous one, except in the **Bone** field, select the **kneeguard.R** item; rename it as **Locked Track.R**.
26. Add a **Damped Track** constraint: **Target** = **Armature**, **Bone** = **kneeguard.L**, **Head/Tail** = **0.728**, **To** = **Y**, and **Influence** = **0.263**. Rename it as **Damped Track.L**.
27. Add a new **Damped Track** constraint and repeat everything as in the previous one, except again in the **Bone** field, select the **kneeguard.R** item; rename it as **Damped Track.R**.
28. Just to be sure, save the file!
29. Go back into **Edit Mode** and in the **Side** view, select the **chest** bone and use *Shift + D* to duplicate it. Press **W** to call the **Specials** pop-up menu and select the **Switch_Direction** item, or else press **Alt + F** directly:



The Specials pop-up menu for the bones

30. Go to the **Bone** window and click on the **Parent** slot under the **Relations** subpanel to select the **chest** item (not connected); then, go to the Deform subpanel and set the **Segments** under **Curved Bones** to 1. Rename the new bone as **armor_ctrl**.
31. Press *Ctrl + R* to roll the **armor_ctrl** bone, in order to be sure that its local *x* axis is pointing towards the front of the model; this is important to make the **Transformation** constraints, which we'll add later, work properly:



The armor control bone

32. Go in the **Front** view. Note that the **X-Axis Mirror** item in the **Armature Options** panel under the **Tool Shelf** is still enabled; select the **Tail** of the **shoulder.L** bone and extrude a new bone going towards the external edge of the armor **spaulder**. Then, select the extruded bone, press **Alt + P | Clear Parent**, and move its **Head** to be positioned above the *joint* of the **spaulder** with the **chest plate**.



Creating the bone for the spaulder

33. Rename the extruded bone and the corresponding mirrored one as **spaulder.L** and **spaulder.R**; parent them to the **armor_ctrl** bone (enable the **Keep Offset** item).
 34. Use **Shift** to select the **spaulder.L** and **arm.L** bones and press **Ctrl + N | Active Bone**; do the same with the **spaulder.R** and **arm.R** bones.
 35. Now, put the **3D Cursor** at the **spaulder.L** bone's **Head** location, and then set the **Pivot Point** to the **3D Cursor** in the 3D window toolbar. Use **Shift + D** to duplicate the **spaulder.L** bone and rotate the duplicate **70** degrees (in the **Front** view, **R | 70 | Enter**).
 36. Place the **3D Cursor** at the **shoulder.L** bone's **Tail** location, select the duplicated bone, and press **Shift + S | Selected to Cursor**. Rename the duplicated bone and the mirrored one as **rotarmor.L** and **rotarmor.R**. Go to the **Relations** subpanel and set the **rotarmor.L** bone as the child of the **arm.L** bone and the **rotarmor.R** bone as the child of the **arm.R** bone. Disable the **Deform** item for both of them:

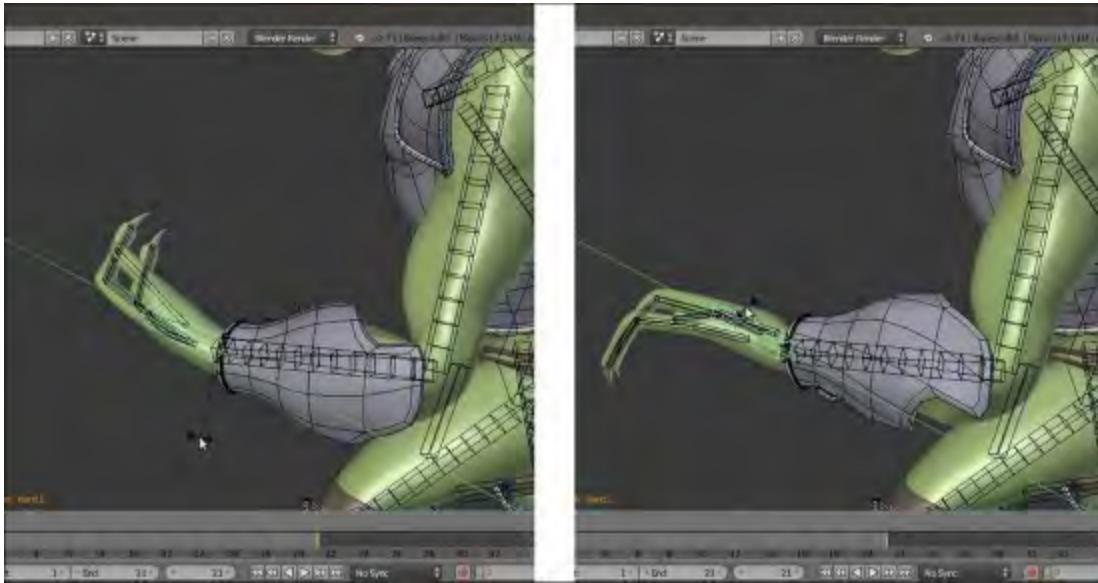


Using the 3D Cursor and the Snap menu to exactly place the bones

37. Go into **Pose Mode**. Select the **spaulder.L** bone and in the **Bone Constraints** window, assign a **Copy Rotation** constraint: **Target = Armature, Bone = arm.L, Space = Pose Space to Pose Space**, and **Influence = 0.200**.
38. Select the **spaulder.R** bone and repeat with the **Bone = arm.R** target.
39. Now, select the **armor_ctrl** bone and assign a **Transformation** constraint. Set **Target = Armature, Bone = rotarmor.L, Source = Rot**, and **Z Max = 20°; Source To Destination Mapping = switch X with Z; Destination = Rot, X Max = 4°**, and **Space = Pose Space to Pose Space**. Rename the constraint as **Transformation_rot.L** and collapse the panel.
40. Assign a second **Transformation** constraint; set everything as in the previous one, except for the target **Bone = rotarmor.R, Source = Rot, Z Min = -20°**, and **Destination X Min = -4°**. Rename the constraint as **Transformation_rot.R** and collapse it.
41. Assign a third **Transformation** constraint; set everything as in the first one, except do not switch X with Z, set **Destination = Loc** and **Z Max = 0.050**. Rename the constraint as **Transformation_move.L** and collapse it.
42. Assign a fourth **Transformation** constraint; set everything as in the second one, except do not switch X with Z; set **Destination = Loc** and **Z Min = 0.050**. Rename the constraint as **Transformation_move.R** and collapse it.
43. Save the file.

How it works...

We couldn't directly use the **forearm** and **calf** bones to rig the **vanbraces** and **greaves** parts because being subdivided **B-bones**, they would *curve* these **armor** parts along the length as they actually do by deforming organic parts as the **forearms** and **shins**, and this would look awkward, as you can see in the following screenshot:

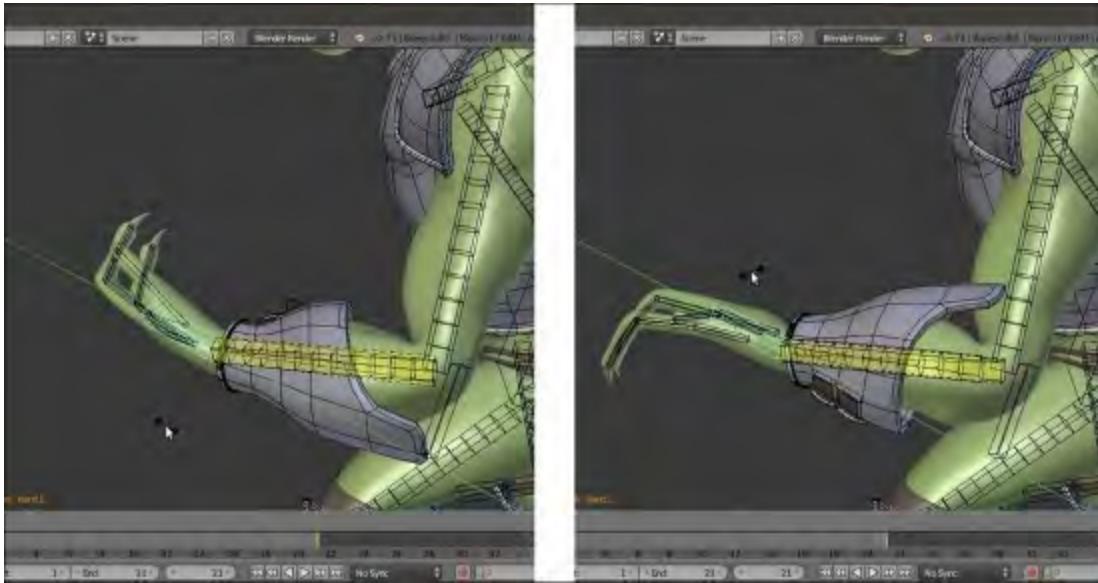


B-bones erroneously deforming stiff objects

Instead, we just duplicated the bones, restored **Segments** and **Ease In** and **Ease Out** to default values, and assigned **2** bone constraints (note that, as already mentioned, the bones have a **Bone Constraints** panel of their own, which is different from the **Object Constraints** one).

The **Copy Rotation** constraint, as the name itself explains, copies the rotation in space of the target **B-bone**; the position inside the chain is granted because the duplicated bones, although not connected, are children of the same bones as the original ones.

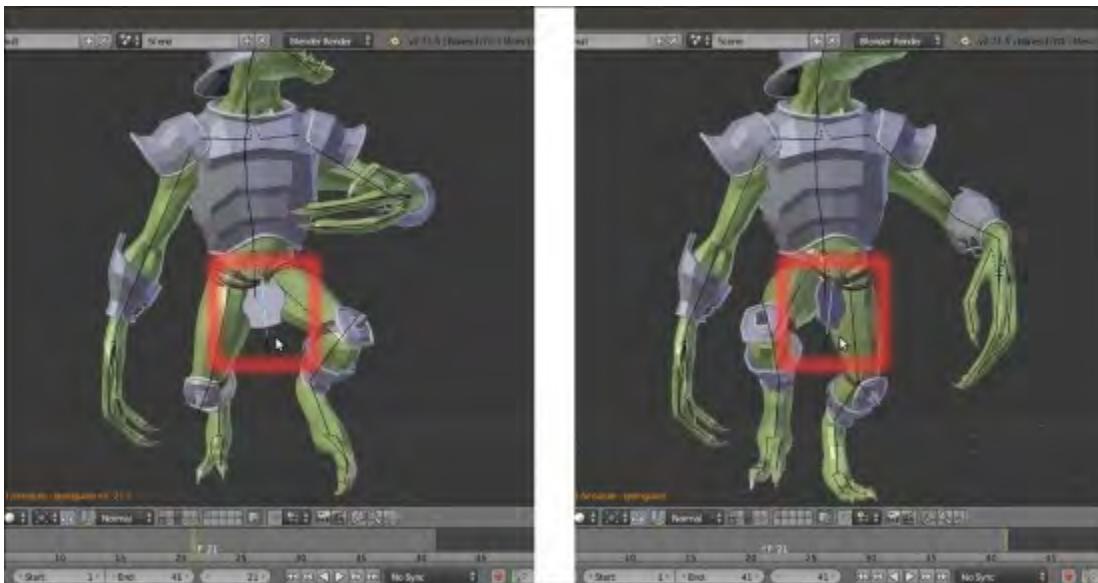
The **Inverse Kinematics** constraint—in this case, is used simply to track the **local y** rotation of the **hand** bone in order to rotate correctly on its **y** axis—is necessary because the **Copy Rotation** constraint doesn't seem to read the **local y** rotation of a subdivided **B-bone** (besides the technical details, it makes sense because that's actually not a rotation in space):



The correct rotation of the stiff armor parts

The constraints assigned to the **groinguard_ctrl** bone are a cheap, but quite an effective, way to fake a rigid body simulation for the **plate** that—in actions, for example, a walk cycle—should interact by colliding with the **Gidiosaurus thighs**. The **Locked Track** constraints, targeted to the **leg** bones, automatically rotate the **plate** according to the **thighs** movements, and the **Dumped Track** constraints, targeted to the **leg** bones as well but with a low influence, add a swinging movement.

The **groinguard** bone, actually the one affecting the **armor plate**, is the child of the **groinguard_ctrl** bone, and so it inherits the constraint's movements but can be used to refine, tweak, or modify the final animation of the plate by hands:



The groinguard bone (and plate) automatically rotating during the walk cycle

The **armor_ctrl** bone is the bone controlling the **armor's Breastplate**; it's the child of the **chest** bone, so it inherits the rotation of the **chest**, but has four **Transformation** constraints.

By using as an input the rotation angle of the **rotarmor.L** and **rotarmor.R** bones (which are children themselves of the **arm.L** and **arm.R** bones), the constraints give to the **Armor chest plate** a slight rotation on the vertical axis and a lateral swinging, driven by the oscillations of the **Gidiosaurus arms**, and simulating of the character's **shoulders** colliding with the **armor plate** during the walk.

Also, the **spaulders** are, in turn, partially rotated by bones with the **Copy Rotation** constraints targeted to the **arms**, but with quite a low influence.

Although better appreciated in motion, the following screenshot will show you the effects as the **arms** rotate backward:



The rotation and swinging of the armor chest plate according to the arms' movements

Building the character's Armature through the Human Meta-Rig

In the previous long and quite complex recipe, we hand-built the deforming elements of an average basic rig for the **Gidiosaurus** character; actually, in Blender, there are other tools to build rigs, particularly meant to facilitate the task, and we'll see them in this recipe and in the following ones.

Now, we are going to take a look at the **Human Meta-Rig** tool.

Getting ready

To be able to use the **Human Meta-Rig** tool, we must first enable the proper add-on:

1. Start Blender and press *Ctrl + Alt + U* to call the **User Preferences** panel. Go to the **Add-ons** tab and under **Categories** on the left-hand side, click on the **Rigging** item. Go to the right-hand side of the panel and check the box to the side of the **Rigging: Rigify** add-on to enable it.
2. Click on the **Save User Settings** button at the bottom-left of the panel and then close it. Because we are starting a rig from scratch again, load the `Gidiosaurus_unwrap_final.blend` file.
3. Disable the **Textured Solid** and **Backface Culling** items in the 3D view **Properties** panel, join the 3D window with the **UV/Image Editor** window, and click on the **11th** scene layer to have only the **Gidiosaurus** mesh visible.
4. Go to the **Object** window and under the **Display** subpanel, enable the **Wire** item; this will be useful in the process to have an idea of the mesh topology when in **Object Mode** and in **Solid** viewport shading mode. However, for the moment, press *Z* to go in the **Wireframe** viewport shading mode.
5. Press *I* on the numpad to go in the **Front** view and *5* on the numpad again to switch to the **Ortho** view.
6. Save the file as `Gidiosaurus_meta_rigging.blend`.

How to do it...

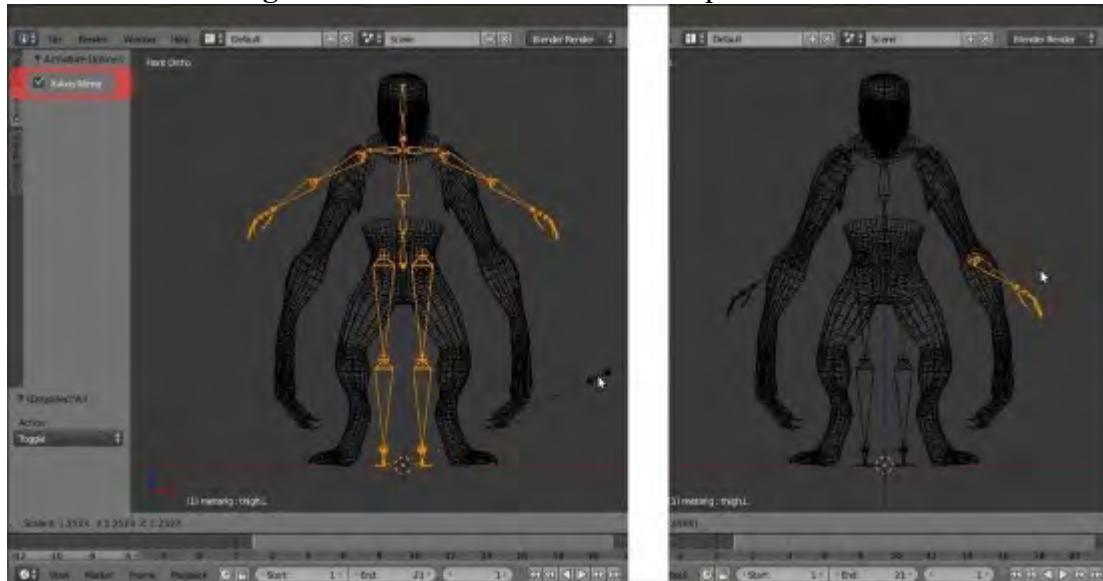
Let's go with the **metarig** itself:

1. Ensure that the **3D Cursor** is at the origin pivot point of the **Gidiosaurus** mesh. Put the mouse cursor in the 3D viewport, press *Shift + A*, and in the pop-up menu, select **Armature | Human (Meta-Rig)**; a biped **Armature**, automatically named **metarig** in the **Outliner**, appears at the **3D Cursor** location:



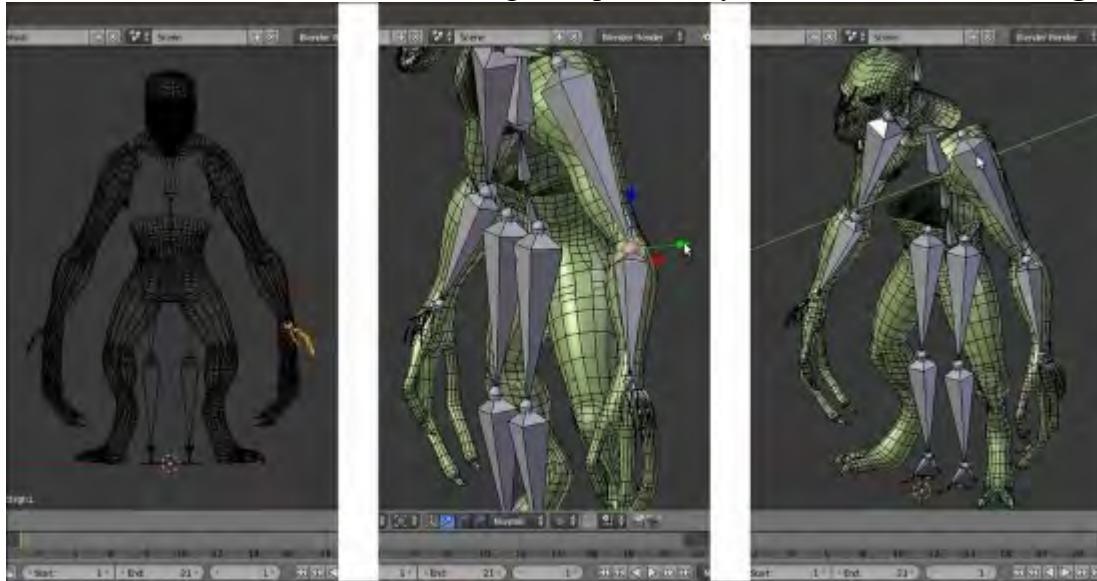
The Human (Meta-Rig) menu and the rig

2. Press the **Tab** key to enter **Edit Mode** and go to the **Options** tab that appeared under the **Tool Shelf** on the left-hand side of the screen; check the box to enable the **X-Axis Mirror** tool under the **Armature Options** item.
3. First, press the period (.) key to set the pivot point around the **3D Cursor** and scale the whole armature bigger while still in **Edit Mode**, and then start to edit locations and proportions of the bones of the **metarig** to fit inside the **Gidiosaurus** shape:



Tweaking the proportions of the bones of the metarig

4. Select the single joints to move them on the right location according to the mesh topology; to do this in a more exact way, just use the snap technique explained in steps 25 and 26 of the *How to do it...* section of the *Building the character's Armature from scratch* recipe. Because of the **X-Axis Mirror** tool we enabled, it's enough to operate only on one side of the **metarig**:



Further tweaking of the bones in Edit Mode

5. Delete the bones that you don't need, for example the extra **fingers** (consider that the **Gidiosaurus** has only three **fingers** in each **hand**), use *Shift + D* to duplicate the bones to be added, for example for the **toes**, and add new bones where missing, for example for the **jaw**, and then parent them. In short, just edit the rig as usual. Again, it should be enough to do all these operations just on one side of the rig:



The completed skeleton rig

6. Save the file.

We can also add premade rigging sets, for example a whole new **leg**, **spine**, or **arm**, by going, with the **metarig** still in **Edit Mode**, to the **Rigify Buttons** subpanel under the **Armature** window in the main **Properties** panel. Select the desired item to be added to the rig and click on the **Add sample** button; the new part gets added to the rig's **pivot point** location and must be moved to the right place and tweaked, rotated, and scaled as needed. Also, the new bones must be named with the correct **.R** or **.L** suffix and the top chain bone must be parented to the bottom **metarig** bone; for example, in the case of a **biped.leg** part addition, the **thigh** bone must be parented (**Ctrl + P | Keep Offset**) to the **hips** bone:



Adding premade rig to the skeleton

How it works...

The **Human metarig** is actually only the first part of a more complex and complete auto-rigging system named **Rigify**, and this we'll see in the next recipe. However, even used by itself, it gives us a ready-made humanoid skeleton to be simply tweaked to fit the character's shape: a good shortcut to quickly build the **Armature** rig considering that, at least in its basic form, all the bones are already properly connected and named with the **.L** and **.R** suffices.

Building the animation controls and the Inverse Kinematic

Whether we built the **Gidiosaurus** deforming rig part by hands from scratch or by the **Human Meta-rig**, we must now add the necessary constraints and controls to allow the animators to easily manipulate the character.

Note

Note that once the mesh is skinned, the rig, as it is at this point, can actually already work by directly selecting the interested bones and rotating them in **Forward Kinematics**; however, to simplify the animator's work (and complicate our life a little bit more), it's good practice to add the **Inverse Kinematic** constraints and the control bones.

Getting ready

Let's start by opening the `Gidiosaurus_rig_from_scratch_02.blend` file; as usual, enter **Edit Mode** to ensure that the **X-Axis Mirror** item in the **Armature Options** subpanel under the **Tool Shelf** is enabled.

How to do it...

We now need to create the control bones; we can do it by extruding from the bones they will drive:

1. Press the **3** key on the numpad to go in the **Side** view and, if not already, select the **Armature**; if necessary, in the **Display** subpanel, change the visualization of the bones from **B-Bone** to **Octahedral**.
2. While still in **Edit Mode**, use **Shift** to select the joints of the **hand** with the **forearm** and the **calf** with the **foot** (it's enough only on one side) and extrude them going backwards (**0.400** along global **y** axis).
3. Rename the new extruded bones as **ctrl_hand.L**, **ctrl_hand.R**, **ctrl_foot.L**, and **ctrl_foot.R** respectively. Deselect the **Deform** item and unparent them all.
4. Select the **Head** of the **hips** bone and repeat: rename the extruded bone as **MAIN**.
5. Select the **hips** bone and in the **Relations** subpanel, parent it as a child of the **MAIN** bone:



Extruding the control bones part 1

6. Select the **elbow** joint (between the **forearm** and **arm**) and extrude a new bone backwards; rename the extruded bone and the mirrored one as **elbow.L** and **elbow.R**. Disable the **Deform** item and parent them (**Keep Offset**) to the **MAIN** bone. Move them backwards by **0.500** along the global y axis.
7. Select the **knee** joint (between the **thigh** and **calf**) and extrude forward; rename the new bones as **knee.L** and **knee.R**. Disable the **Deform** item and parent them (**Keep Offset**) to the **MAIN** bone as well. Move them forward by **-0.500** along the global y axis;



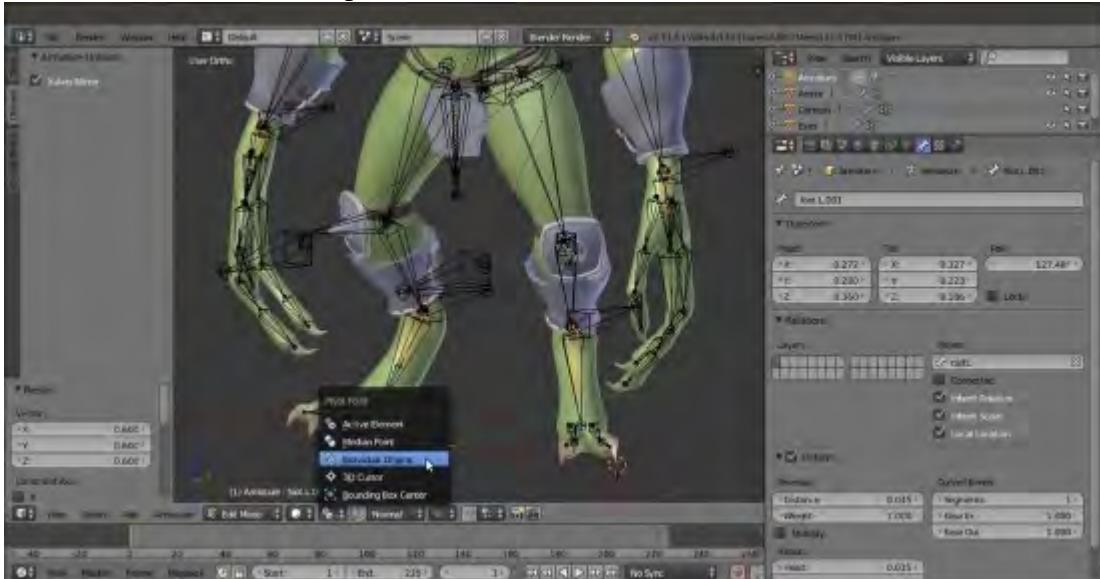
Extruding the control bones part 2

8. Go into **Pose Mode** and select the **forearm.L** bone; go to the **Bone Constraints** window and assign an **Inverse Kinematics** constraint. Set **Target = Armature**, **Bone = ctrl_hand.L**, **Pole Target = Armature**, **Bone = elbow.L**, **Pole Angle = -90°**, and **Chain Length = 2**, and deselect **Stretch**. Repeat the process for the **forearm.R** bone:



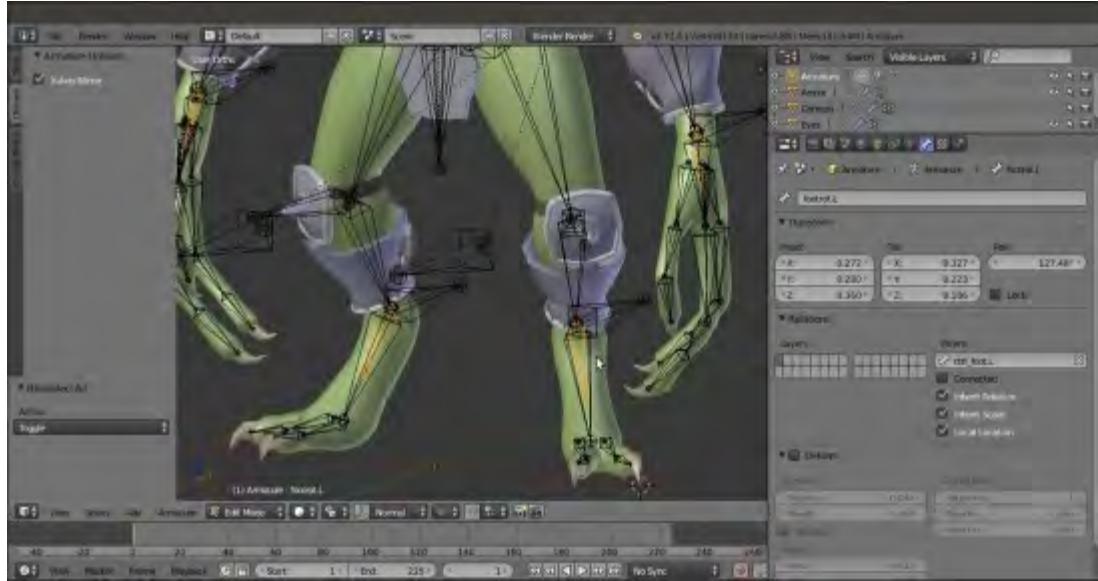
Assigning the IK constraint to the forearm.L bone

9. Do the same for the **calf.L** and **calf.R** bones, using the **ctrl_foot.L** and **ctrl_foot.R** bones as targets and the **knee.L** and **knee.R** bones as poles, but set the **Pole Angle** to **90°** for both.
10. Now, go back into **Edit Mode**, select the **hand** and **foot** bones, and use **Shift + D** to duplicate them. Click on the **Pivot Point** button on the 3D view toolbar, select the **Individual Origins** item, and then scale the duplicated bones smaller to **0.600**:



Scaling the bones smaller on their individual origin

- Deselect the **Deform** item for all of them, and then rename them as: **handrot.L**, **handrot.R**, **footrot.L**, and **footrot.R**.
- In the **Relations** subpanel (or by the *Ctrl + P | Keep Offset* shortcut), parent **handrot.L** to **ctrl_hand.L**, **handrot.R** to **ctrl_hand.R**, **footrot.L** to **ctrl_foot.L**, and **footrot.R** to **ctrl_foot.R**:



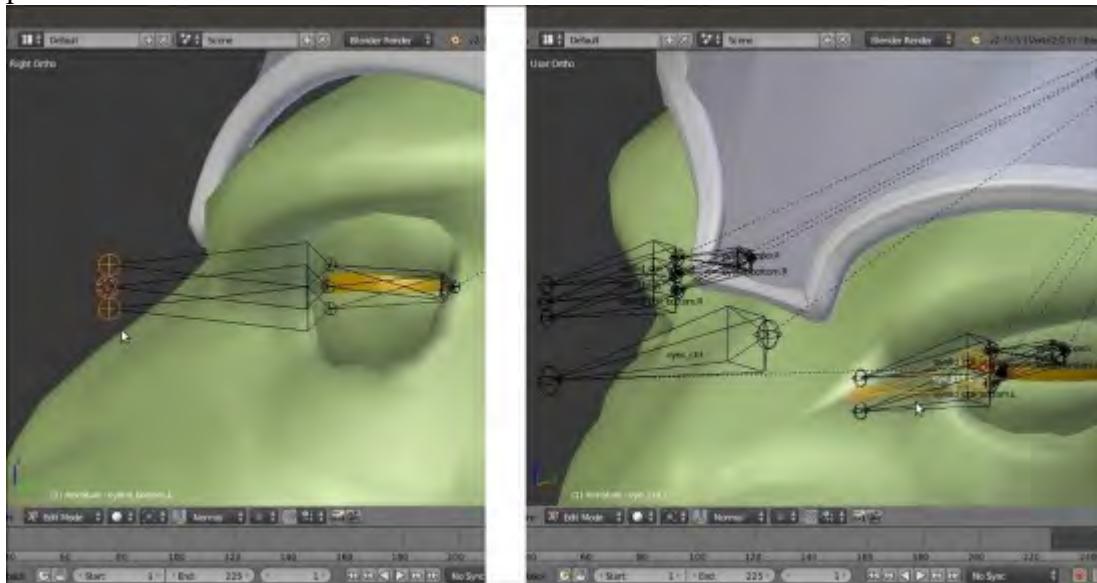
Using the Parent slot under the Relations subpanel

- Use *Shift* to select the **ctrl_foot.L** bone and the **foot.L** bone and press *Ctrl + Alt + A* to align the first one with the active one; then, select only the **ctrl_foot.L** bone, and by the toolbar widget manipulator set to **Normal** orientation, rotate it **245°** on the **x** axis:



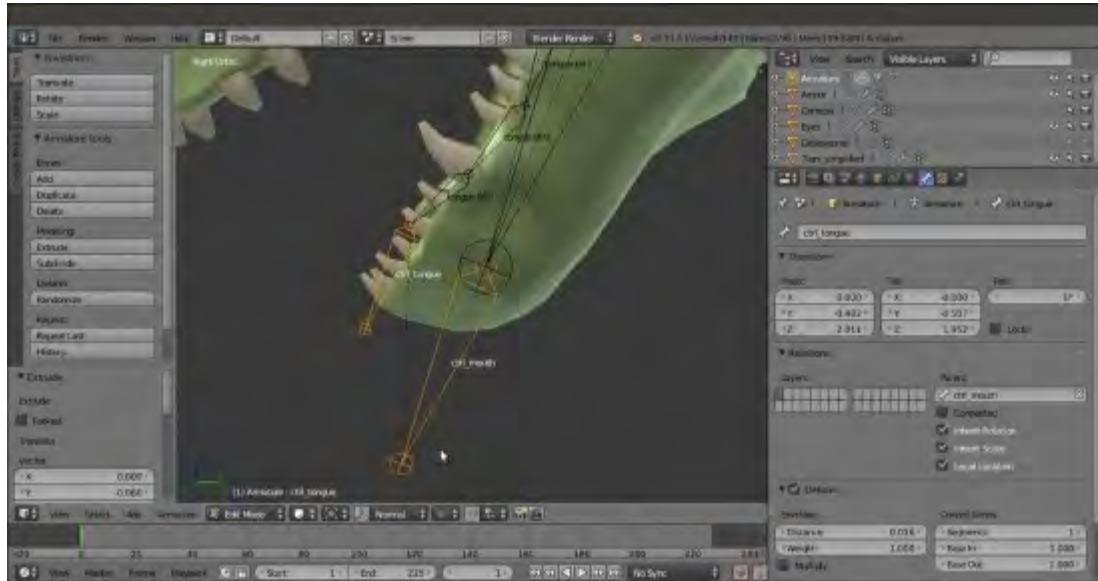
Rotating the bone on the Normal orientation by the widget

14. Go into **Pose Mode** and select the **hand.L** bone; assign a **Copy Rotation** bone constraint with **Target = Armature** and **Bone = handrot.L**, and set **Space = Pose Space** to **Pose Space**.
15. Repeat for the other **hand** bone and feet.
16. Select the **Tails** of the **eyelid_upper.L**, **eyelid_bottom.L**, and **eye.L** bones and extrude forward by **0.0600** along the **y** axis; rename them as **eyelid_ctrl_upper.L**, **eyelid_ctrl_bottom.L**, and **eye_ctrl.L** and the same names with the **.R** suffix for the mirrored ones.
17. Add a new bone in the middle front of the **eyes**, rename it **eyes_ctrl**, and parent it with offset to the **head** bone; then, select the **eye_ctrl.L** and **eye_ctrl.R** bones and parent them with offset to the **eyes_ctrl** bone.
18. Select the **eyelid_upper.L**, **eyelid_upper.R**, **eyelid_bottom.L**, and **eyelid_bottom.R** bones and parent them with offset to the **head** bone:



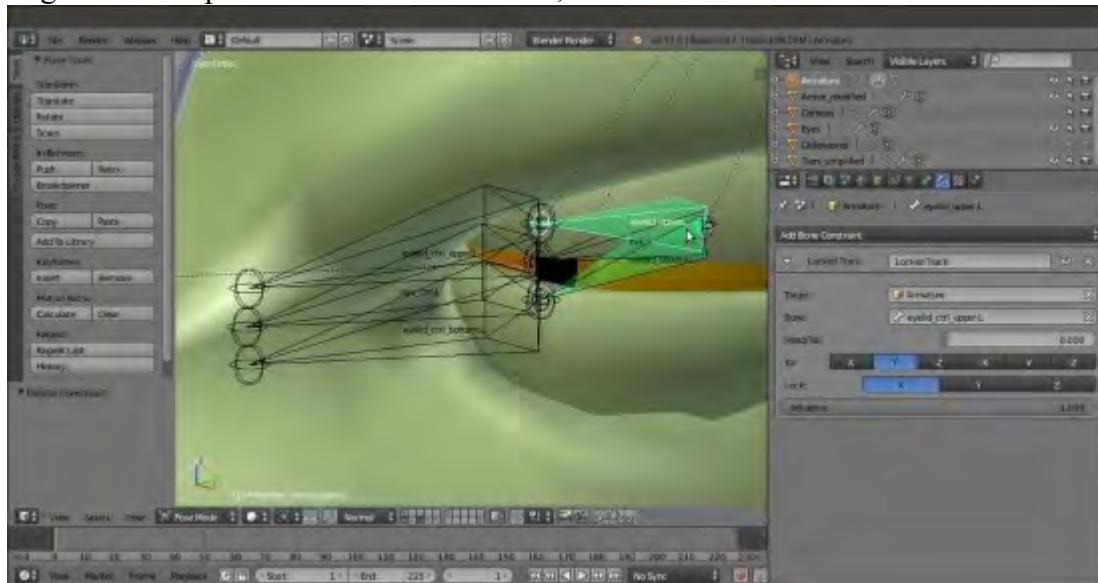
The eyes control rig

19. Select the **Tails** of the **mand** and **tongue.005** bones and extrude; rename the extruded bones as **ctrl_mouth** and **ctrl_tongue**. Parent with offset the **ctrl_tongue** bone to the **ctrl_mouth** bone and this latter bone to the **head** bone:



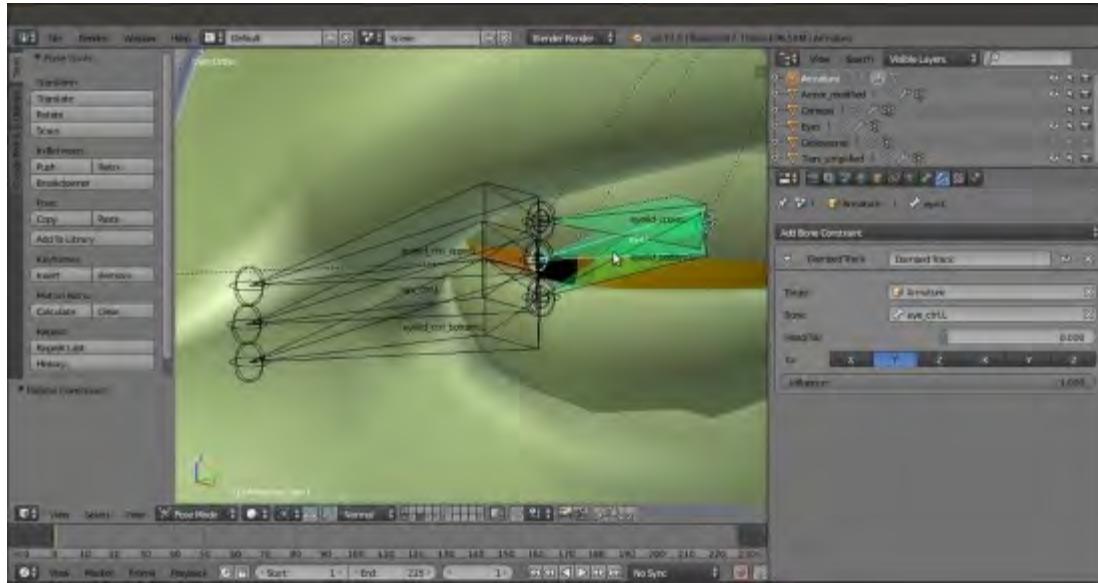
Extruding the control bones for the tongue and jaw

20. Go into **Pose Mode** and assign **Locked Track** constraints to the **eyelid_upper** and **bottom** with target to the respective extruded **ctrl** bones; set **Lock** to **X**:



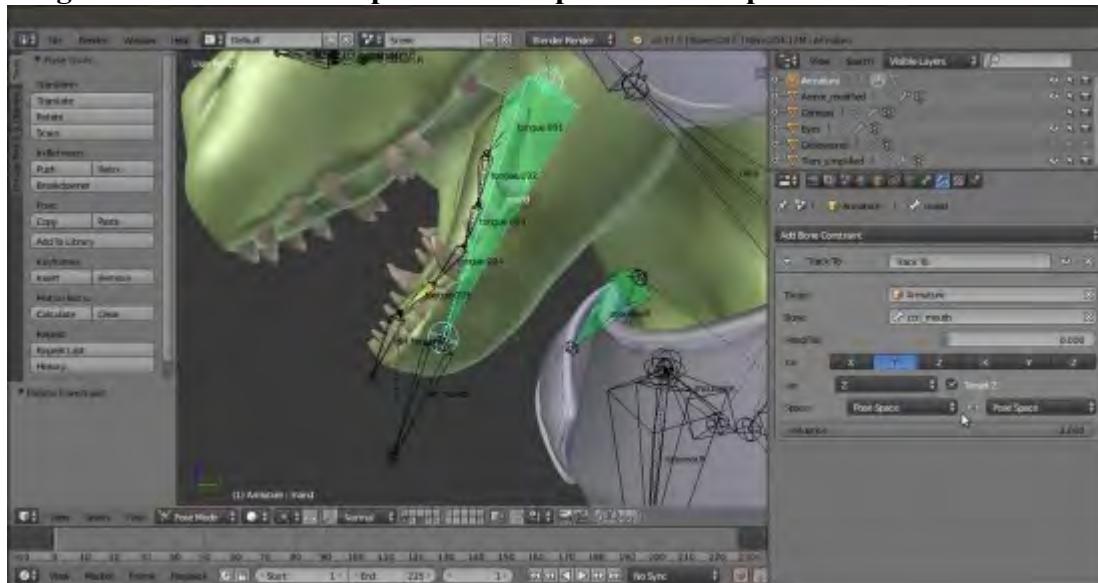
Assigning the Locked Track constraints for the eyelid's controls

21. Assign **Damped Track** constraints to the **eye.L** and **eye.R** bones, again with target to the respective extruded **ctrl** bones:



Assigning Damped Track constraints for the eye's controls

22. Assign a **Track To** constraint to the **mand** bone with target to the **ctrl_mouth** bone; check the **Target Z** item box and set **Space = Pose Space** to **Pose Space**:



Assigning a Track To constraint to the mand bone

23. Assign an **Inverse Kinematics** constraint to the **tongue.005** bone with target to the **ctrl_tongue** bone; set **Chain Length** to **5**, deselect the **Stretch** item, and then enable also the **Rotation** item;



The IK constraint for the bone's chain of the tongue

At this point, the main controls for the **Gidiosaurus** rig are made; still something is missing, for example, the controls to drive **fingers** or/and **toes** bones as a whole, and also a *muscle system* layer of bones with the **Stretch To** constraints that can be added to improve the realism of the model. However, this latter option is quite a complex matter and, for the moment, we will stop here (maybe in another book).

The very last thing to do is to assign **Custom Shapes** (usually, simple meshes located on the last scene layer) to the control and animatable bones widget, and move the rest of the bones to the third **Armature** layer to be out of view.

To see the completed rig with the **Custom Shapes** assigned to the control bones, load the **Gidiosaurus_rig_from_scratch_03.blend** file;



The rig with and without Custom Shapes and with the deformation bones hidden on the third (disabled) Armature layer

See also

- <http://www.blender.org/manual/rigging/index.html>

Generating the character's Armature by using the Rigify add-on

We have already seen that the **Human Meta-Rig** armature is part of the **Rigify** add-on. It is a tremendously useful Python script, coded by Nathan Vegdhal, that we enabled two recipes ago, and in this recipe, we are going to use that to build the final rig for the **Gidiosaurus**.

Getting ready

The preparation steps to use the **Rigify** add-on are the same as we did in the *Building the character's Armature through the Human Meta-Rig* recipe: after we have enabled the add-on in the **User Preferences** panel, we load the `Gidiosaurus_unwrap_final.blend` file, add the **Human metarig** to the scene, and then tweak the bone's position, rotation' and size in **Edit Mode** to fit the character's shape and topology.

Also, because the rig generated by the **Rigify** add-on uses some Python script, in the **User Preferences** panel, we must enable the **Auto Run Python Scripts** item (in the **File | User Preference | File** tab, click on the **Auto Run Python Scripts** checkbox).

How to do it...

At this point, in **Object Mode**, we can go to the bottom of the **Armature** window under the main **Properties** panel and click on the **Generate** button in the **Rigify Buttons** subpanel at the bottom of the **Armature** window; the add-on will automatically generate a new **rig** (simply named **rig** in the **Outliner**) using the **metarig** skeleton as an input and adding all the necessary **IK** constraints, the bone's widget controls (generated and located in the last scene layer), and also placing the different bones on different **Armature** layers that are easily accessible through the Python interface created by the script in the 3D window **Properties** sidepanel on the right-hand side (the **Rig Layers** subpanel):



The generated rig with the Rig Layers subpanel

Keep the **metarig** and move it to another layer, just in case we need to do some editing to it in the future; in fact, by testing the generated rig, sometimes you discover that something must be changed to work in a different way. In this case, it is enough to modify the **metarig** and generate the rig again by the add-on that automatically reuses the elements of any already existing rig and the bone's widgets on the last scene layer.

Keep in mind that the generated rig can (and often must) be edited later anyway; after the rig generation, save the file as `Gidiosaurus_rigify_01.blend`.

How it works...

Being conceived to build a rig for a *generic biped humanoid* character, the **Rigify** add-on doesn't generate everything you need automatically: in our case, bones for the **jaw**, **tongue**, **eyes**, and **eyelids** must be added by hands after the rig regeneration and as explained in the *Building the character's Armature from scratch* recipe.

The choice to let face-rig elements, at least initially, out of the **Rigify** add-on has been intentional by Vegdhal, who thinks that a face-rig tool would probably be better as a separate add-on. By the way, in the last Blender releases, it is available, in the **Armature** menu, a **Pitchipoy human rig** option, which is an addition to the **Rigify** script that should help in the face's rig construction (<http://pitchipoy.tv/?p=2026>).

Also, at least for the moment, the **Rigify** add-on doesn't accept custom rig parts, but only the premade parts that we can add to the **metarig** by the **Add Sample** button under the **Rigify Buttons** subpanel in **Edit Mode**; for example, the premade leg rig (**biped.leg**) has only one bone and not two for the toes, as would be necessary for the **Gidiosaurus** character, but in any case, once the final rig is generated by the script, all the necessary additions and modifications can be (quite) easily made by hand.

Obviously, to modify the generated rig, knowing how a rig works in Blender is mandatory: you can rest upon the *Building the character's Armature from scratch*, *Perfecting the Armature to also function as a rig for the Armor*, and *Building the animation controls and the Inverse Kinematic* recipes in this chapter.

In the following screenshot, you can see the **Rigify**-generated rig modified with all the additional bones for the **Armor**, **eyes**, **mouth**, and **tongue**, with the necessary added constraints and the **two toed feet** bones; the file is saved as `Gidiosaurus_rigify_02.blend`:



The final total rig

See also

- <http://blenderartists.org/forum/showthread.php?200371-Rigify-Auto-rigging-system-new-and-improved>

Chapter 7. Skinning the Low Resolution Mesh

In this chapter, we will be covering the following recipes:

- Parenting the Armature and Mesh using the Automatic Weights tool
- Assigning Weight Groups by hand
- Editing Weight Groups using the Weight Paint tool
- Using the Mesh Deform modifier to skin the character
- Using the Laplacian Deform modifier and Hooks

Introduction

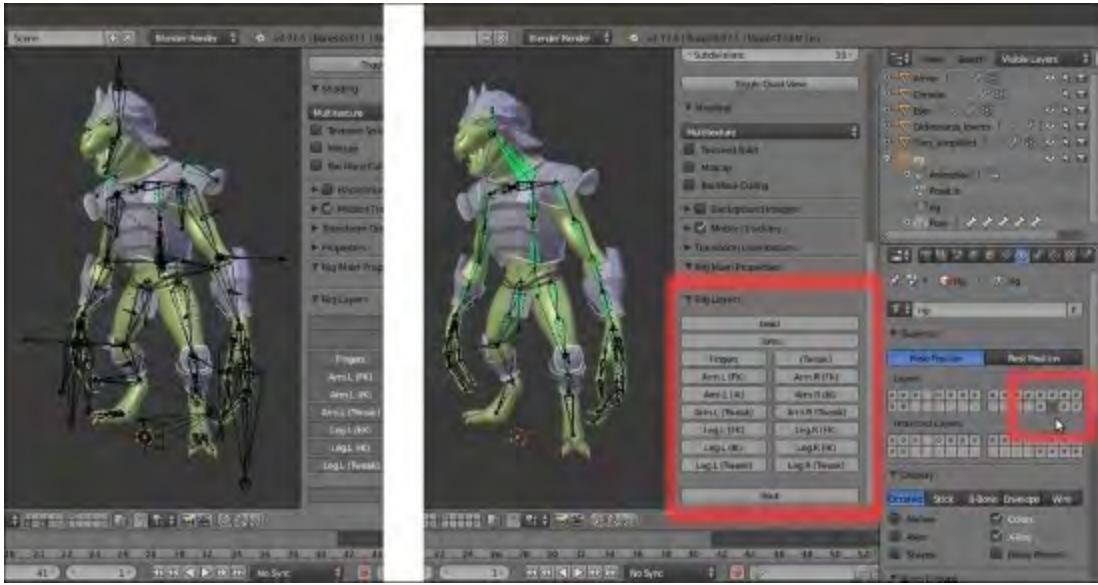
In the previous chapter, we saw the **rigging** stage, that is, how to build the character's rig (which in Blender is called an **Armature**) that will be used to deform the mesh for animations. In this chapter, instead, we are going to see quicker and more effective ways to do the **skinning** that is a necessary step to bind the bones of the **Armature** to the mesh's vertices so that they can be deformed.

To allow an **Armature** to deform a **Mesh**, they must be parented with some kind of relation; in Blender, usually you must select the **Mesh** and then *Shift* select the **Armature** and press *Ctrl + P* to parent them with different options.

This automatically makes the **Mesh** object a child of the **Armature** object and assigns the **Armature** modifier to the **Mesh**. In fact, the parenting would not be strictly necessary; it would be enough to assign an **Armature** modifier to the mesh and manually select the rig as a deforming object, but it's a good habit to use the *Ctrl + P* parenting to have the rig as a parent of the mesh, also in **Object Mode**. This way, whenever you move the **Armature** in **Object Mode**, the mesh will follow it automatically.

For the examples in these recipes, to skin the **Armature** to the **Gidiosaurus** mesh, we are going to use the final version of the rig we have built with our hands: the one saved as `Gidiosaurus_rig_from_scratch_02.blend`.

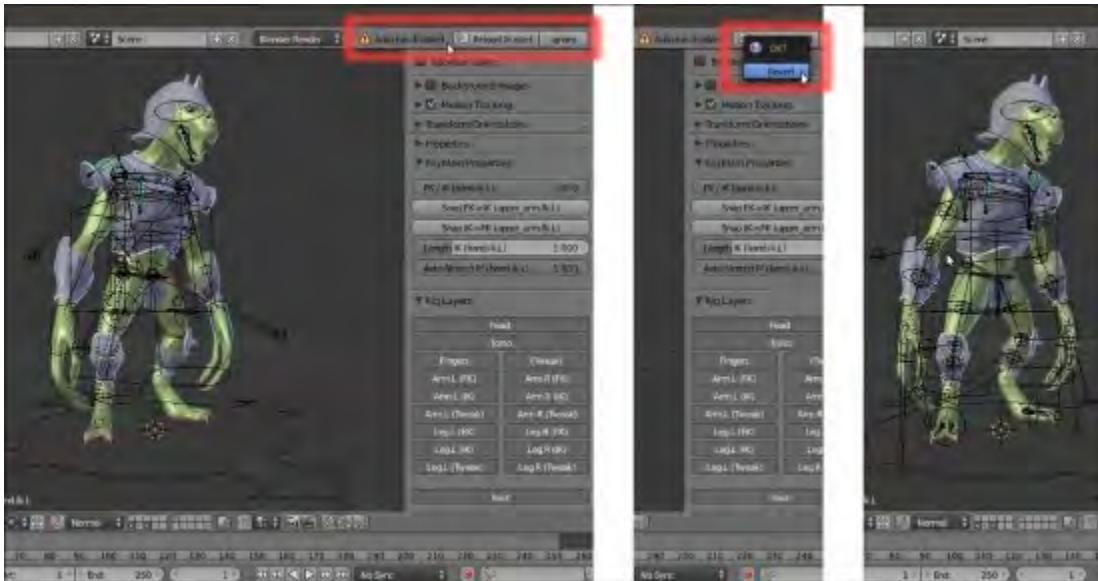
Anyway, if you want to put this to practice, in this chapter, with a more complex and complete **Rigify** armature (`Gidiosaurus_rigify_02.blend`), the procedure is exactly the same. In this case, even if not strictly necessary, remember that you can enable the **30th Armature layer** (in total there are **32**) to show the deforming bones; instead, disable the visibility of all the other bone layers also by the Python button interface in the **Rig Layers** subpanel under the 3D window **Properties** side panel:



The Rig Layers panel in the N Properties sidepanel and the Armature bone layers button in the Skeleton subpanel under the main Properties panel

Remember to check in your **User Preferences** panel (press **Ctrl + Alt + U** to call it) if you have, under the **File** tab, the **Auto Run Python Scripts** item enabled; otherwise, the rig based on Python scripts or expressions (like the rigs obtained through the **Rigify** add-on) won't work properly.

In this case, Blender will warn you through an **Auto-run disabled** message visible in the top main header; it's enough to click the **Reload Trusted** button to the right and then confirm by clicking on the **Revert** item in the pop-up menu that appears, to reload the .blend file with the scripts enabled and to have everything working as expected:



To the left, you can see several bones apparently missing in the rig because it is wrongly oriented, and the "Auto-run disabled" warning in the top main header; to the right, you can see the restored rig

Parenting the Armature and Mesh using the Automatic Weights tool

In this recipe, we are going to see one of the more commonly used parenting options: the handy **Automatic Weights** tool.

Getting ready

Start Blender and open the `Gidiosaurus_rig_from_scratch_02.blend` file.

1. Select the **Armature** item in the **Outliner** and press *Ctrl + Tab* to go out of **Pose Mode** and enter **Object Mode**.
2. Go to the **Armature** window under the **Properties** sidepanel to switch the **Display** mode from **Wire** to **Octahedral** and deselect the **Shapes** item.
3. Enable the third **Armature** layer by clicking on the **3rd** button under the **Skeleton** subpanel.
4. Disable the **13th** scene layer to hide the **Armor**.
5. Go in to **Edit Mode** and *Shift* multi-select the **MAIN** bone, the **pole** bones and the **ctrl** bones; in short, all the bones that don't have to deform anything, but are used to control the rig. Press *Shift + W* and in the **Toggle Bone Options** pop-up panel, select the **Deform** item to disable it for all of them at once:



Toggling the Deform item for all the selected bones at once

6. Now, deselect everything and select all the bones that, in the previous chapter, we had added specially to rig the **Armor** object, using the **Armature Layers** buttons: the **armor_ctrl** bone, **groinguard**, **vanbrace.L** and **.R**, **greaves.L** and **.R**, **kneeguard.L** and **.R**, **spaulder.L** and **.R**; again, press *Shift + W* | **Deform** to disable the option.



Repeating for the Armor object bones

7. Don't *deselect* the **Armor** bones, simply switch from **Edit Mode** to **Object Mode**.

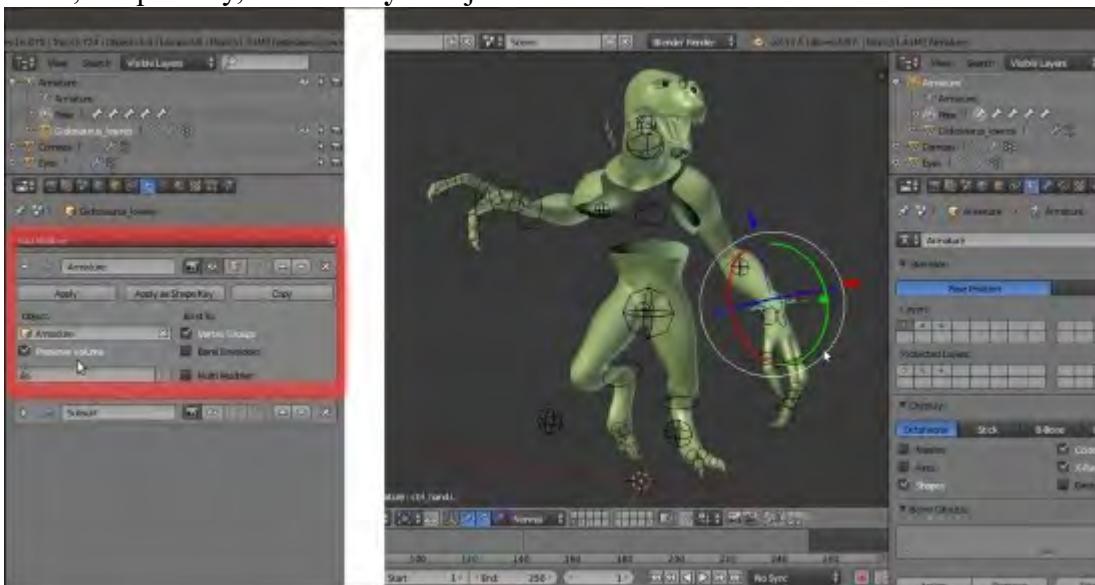
How to do it...

1. Select the **Gidiosaurus_lowres** object and then *Shift-select* the **Armature**, and press **Ctrl + P**; in the **Set Parent To** pop-up menu; select the **With Automatic Weights** item:



The Set Parent To pop-up menu

2. Reselect the **Armature**, go in to **Edit Mode**, and press **Shift + W | Deform** to re-enable the item for the still-selected **Armor** bones; then, go out of **Edit Mode**.
3. Now, reselect the **Gidiosaurus** object; go to the **Object Modifiers** window, move the newly created **Armature** modifier upwards in the stack, and enable the **Preserve Volume** item.
4. Disable the *Display modifier in viewport* button (the one with the eye icon) of the **Subdivision Surface** modifier to speed up the 3D viewport (sadly, Blender still has very bad real-time viewport performances, so even if you have a lot of RAM and a powerful workstation, it's wise to stay as light as you can).
5. Select the **Armature** and under the **Object Data** window, re-enable the **Shapes** item and hide the second and the third **Armature Layer**; press **Ctrl + Tab** to go in **Pose Mode** and try to select some of the control bones to move or rotate them and so control how they are deforming the mesh; temporarily, hide the **Eyes** object in the **Outliner**.



The Armature modifier subpanel and the posed mesh

To rotate the bones on their local axis, enable the **3D manipulator widget** in the 3D view toolbar (**Ctrl + Spacebar**), click on the **Rotate** icon, and set **Transformation Orientation** to **Normal**.

6. Save the file as `Gidiosaurus_automweights.blend`.

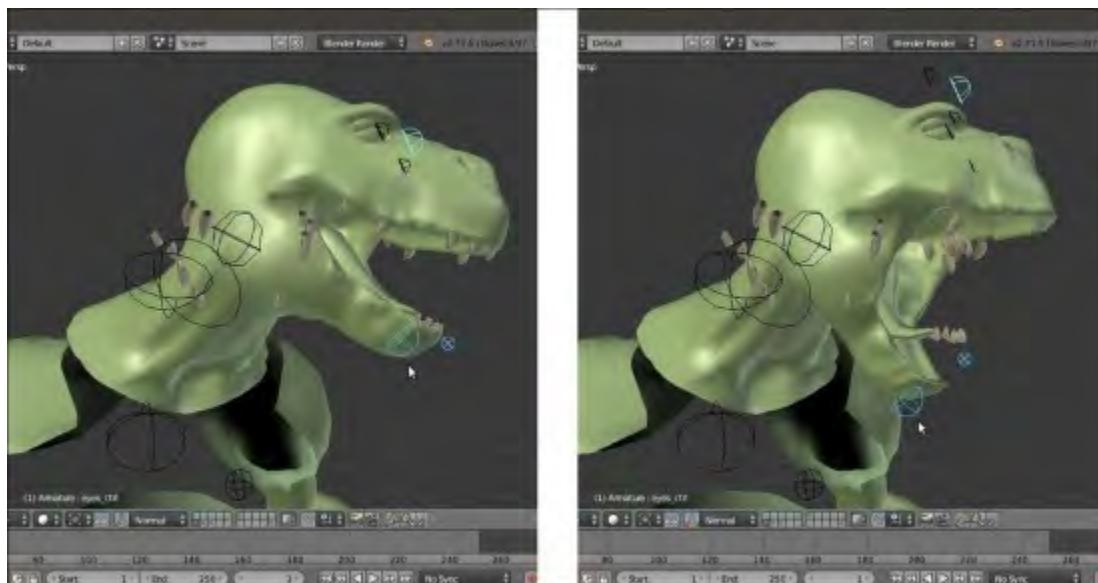
How it works...

The **Automatic Weights** tool creates the necessary **Vertex Groups** based only on the bones that have been set as **Deformers** in the subpanel under the **Bone** window. It then assigns weights inside a range from **0.000** to **1.000** to the vertices contained in these vertex groups, calculating their proximity to the bone with the same name. In short, the **arm.L** bone will deform only the vertices inside the **arm.L** vertex group, and with an intensity based on their weights.

Because we used the **Automatic Weights** tool to skin only the sole **Gidiosaurus** mesh (leaving the skinning of other objects such as the **Eyes** or the **Armor** for the next recipe and method), before the parenting we had to check for any bone erroneously left as a deformer (that is, one of the several control bones in the previous chapter), but, especially we had to temporarily disable the **Deform** item for the **Armor** bones, which otherwise would have also been evaluated by the tool for the **body**.

In most cases, the **Automatic Weights** tool can give quite good results without the need of further tweaking; however in some areas, for example the **head**, where the **head** bone length doesn't fully fit the upper part of the shape of the mesh and where there are also other deforming bones, it can easily fail.

Look at the following screenshot; at first, by rotating the **head** control, the only issue seems to be some of the **teeth** left out from the calculations but then, simply by moving the controls for the **eyes**, **tongue**, and **jaw**, it becomes evident that the tool assigned several vertices to the wrong bones merely based on their proximity to that part of the mesh:



The failure of the Automatic Weights tool parenting

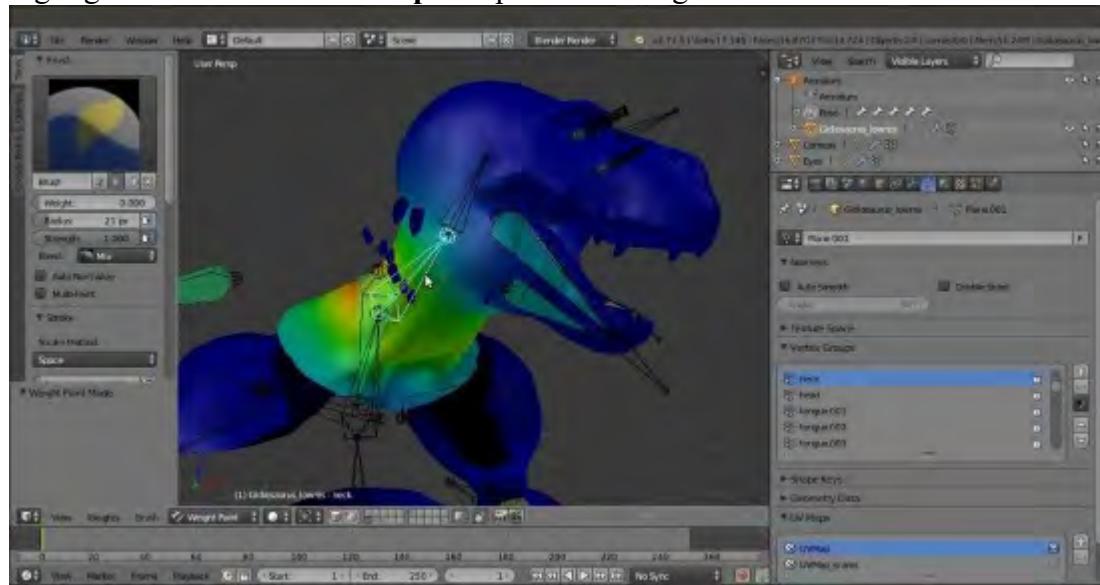
Although at first sight this can appear to be a total mess, it's usually less complex to fix than one might think.

For the moment, by selecting the **Gidiosaurus** mesh and pressing **Ctrl + Tab**, we go in to **Weight Paint** mode, and by right-clicking on a bone (the **Armature** is still in **Pose Mode**), the weights of the corresponding vertex group became visible as colored areas on the mesh; the color **red** corresponds to a weight value of **1.000** and **blue** to a value of **0.000**, with all the intermediate hues corresponding to the intermediate values. For example, **green = 0.500** and so on.

There's more...

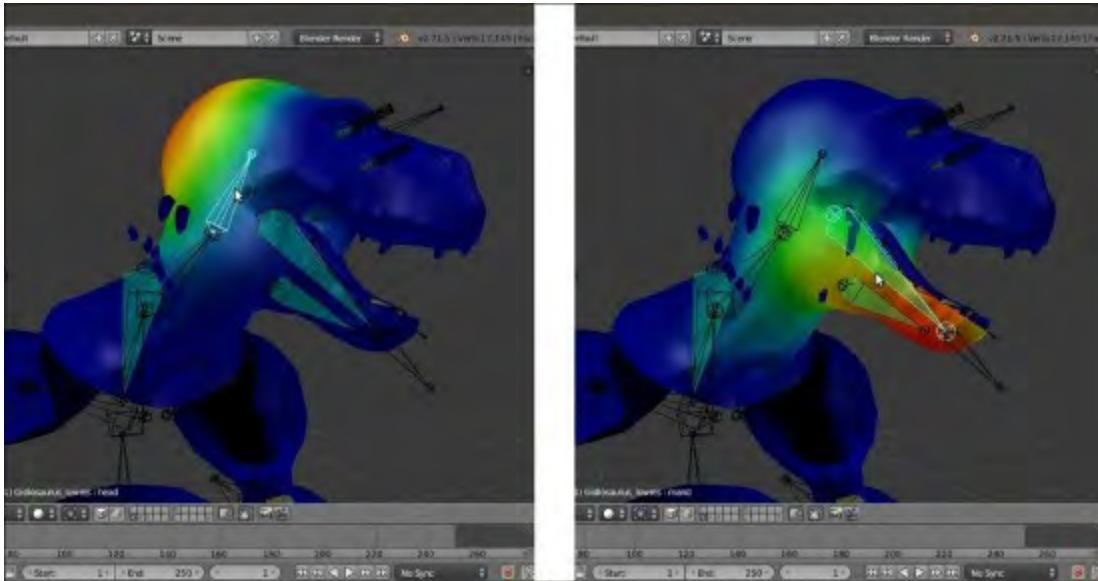
Let's see all this step by step:

1. Select the **Armature** and, while still in **Pose Mode**, enable the visibility of the **third Armature layer** (and therefore of all the deforming bones) and then disable the **Shapes** item again.
2. Select the **Gidiosaurus** mesh and by pressing **Ctrl + Tab**, enter **Weight Paint mode** (or switch to it by the *object interaction mode* button on the toolbar of the 3D view).
3. Click on any one of the deforming bones, for example the **neck** bone, and notice that while the weights appear on the mesh surface, at the same time the corresponding vertex group is highlighted in the **Vertex Groups** subpanel to the right:



Visualizing the vertex groups on the mesh

By clicking on the **head** bone and/or the **mand** bone, the reasons for the bad deformations are immediately clear: the **Automatic Weights** tool didn't assign the whole upper part of the **head** of the character to the sole **head** vertex group (and therefore to the bone with the same name) with a full value of **1.000**; instead, it assigned part of the **head** mesh to the **eyes** bones, other parts to the **tongue.005** bone, some to the **mand** bone, and so on.



Different weights of the vertex groups associated with different bones

Obviously, this isn't the tool's fault, but it is an *unavoidable issue* due to the particular arrangement of the bones in the **head** area and can be quite easily fixed anyway; we'll see how in the next and the *Editing the Weight Groups by the Weight Paint tool* recipe.

See also

- <http://www.blender.org/manual/rigging/skinning/obdata.html>

Assigning Weight Groups by hand

This technique is the oldest way to assign weights to vertices groups in Blender. Although now there are quicker ways to do the same thing, in some cases it's still one of the best approaches, which can reveal itself to be quite useful mainly because you can precisely select individual or edge-loops of vertices to be weighted inside a group.

Getting ready

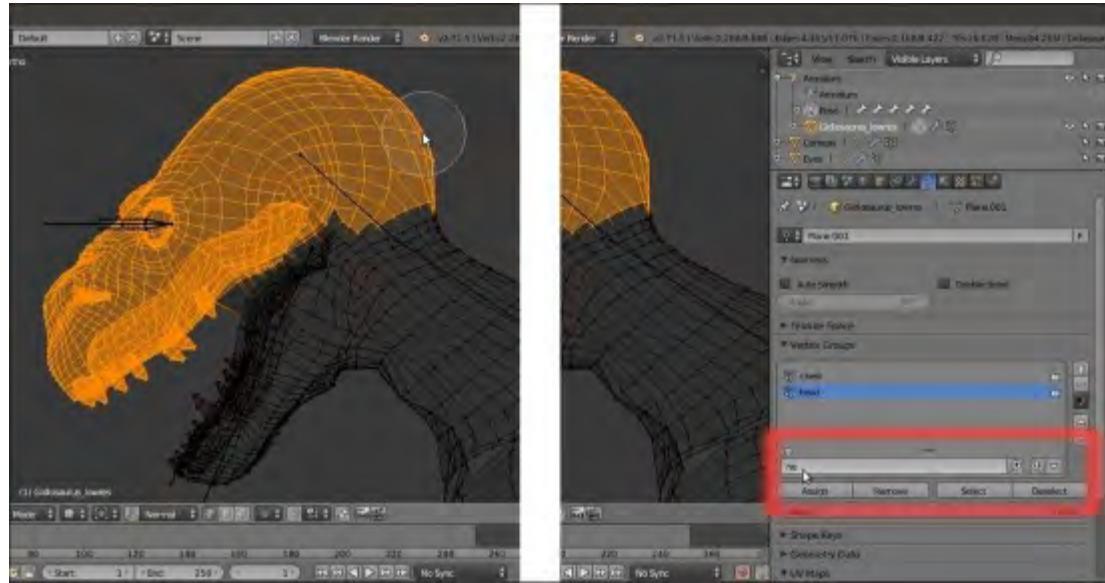
Open the `Gidiosaurus_autoweights.blend` file we saved in the previous recipe.

1. If necessary, press `Ctrl + Tab` to go out of **Weight Paint** mode.
2. Select the **Armature** (which should still be in **Pose Mode**), press the `A` key twice to deselect-select all the bones, and press `Alt + R` and `Alt + G` to clear any rotation or position and restore the default pose.
3. Press the `3` key on the numpad to go in to **Side** view; if necessary, the `5` key on the numpad to go in to **Ortho** view and the `Z` key to go in to **Wireframe** viewport shading mode.
4. Select the **Armature** and disable the **Shapes** item; switch the draw mode of the bones to **Stick** and enable the third **Armature** layer to show the deforming bones.
5. Save the file as `Gidiosaurus_skinning_01.blend`.

How to do it...

First, we are going to use this technique to fix the **head** deformation as follows:

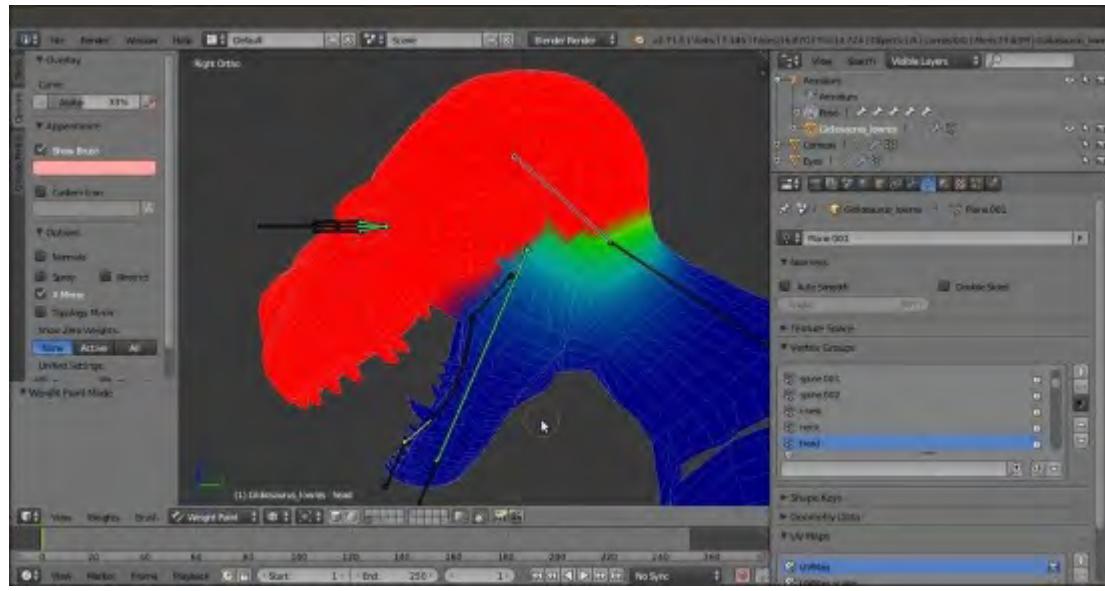
1. Press `Shift + B` to draw a box around the **head** of the **Gidiosaurus** mesh and automatically zoom to it. Select the mesh and enter **Edit Mode**.
2. Press the `C` key, through which the mouse cursor turns into a circle whose diameter can be set by scrolling the mouse wheel.
3. Start to *paint-select* the vertices you want to add to the vertex group; in this case, we must add the whole **upper head** to the **head** vertex group and also include the **upper teeth** that were missing in the group.
4. Be sure that the **head** vertex group is the selected one in the **Vertex Groups** subpanel under the **Object Data** window to the right, and that the **Weight** slider is set to **1.000**; then, click on the **Assign** button.
5. To quickly find a required vertex group, instead of slowly scrolling the list, just click on the grayed out little `+` icon at the bottom of the **Vertex Groups** window (just above the **Assign** button) to expand a blank search field and then write a few letters of the group's name followed by the `Enter` key:



The vertex group names search function

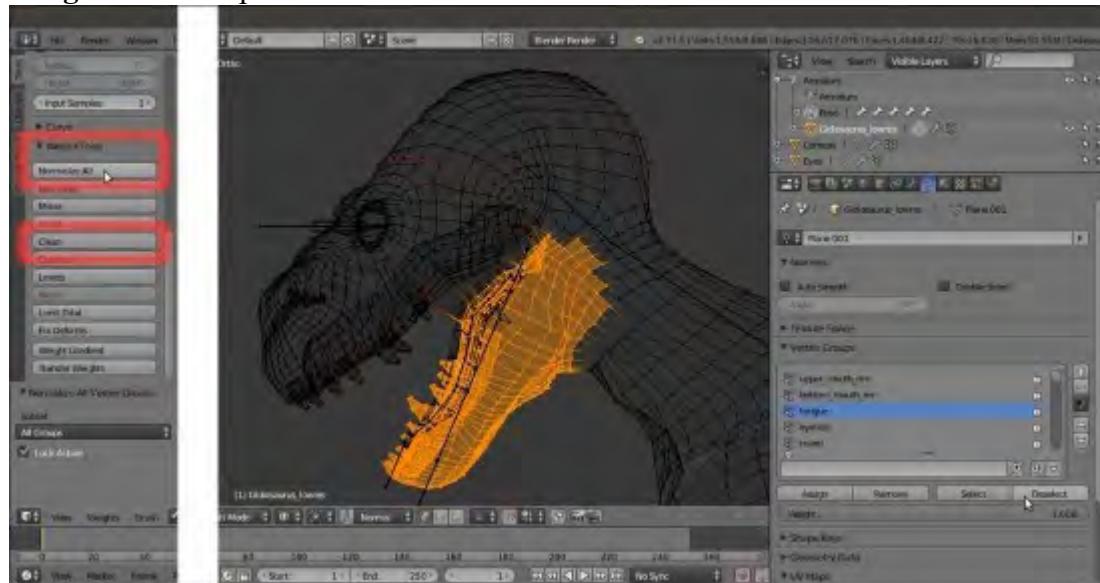
To get the complete list of vertex groups' names back, just erase the letters you wrote in the field and press *Enter*.

- Now, switch from **Edit Mode** to **Weight Paint** mode again, where the **head** vertex group colors show a lot different than before:



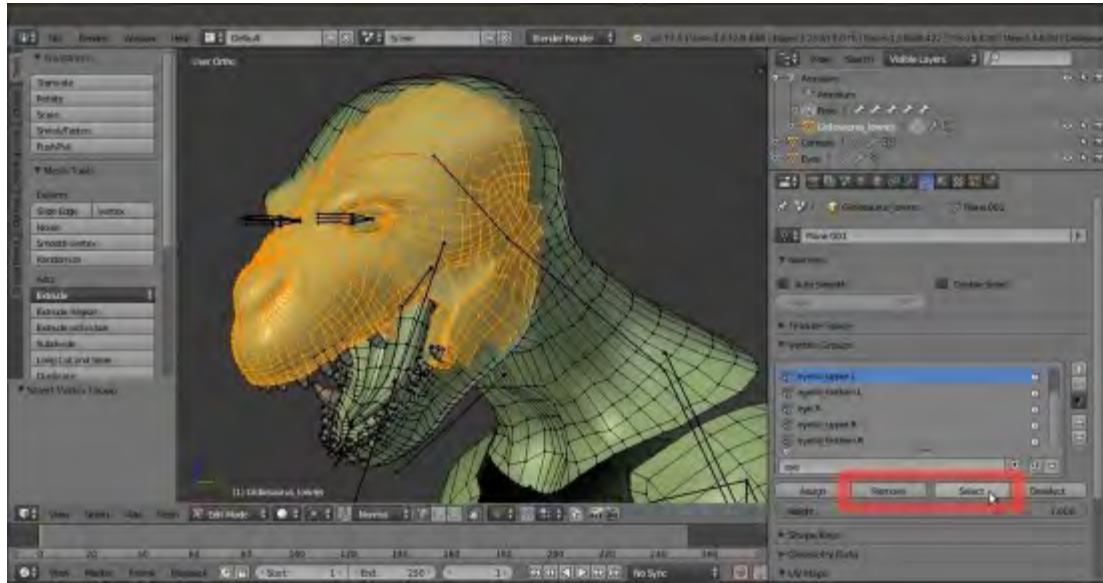
The modified "head" vertex group

7. Now, while still in **Weight Paint** mode, go to the **Tools** tab under the **Tool Shelf** to the left of the screen and, in the **Weight Tools** subpanel, first click on the **Normalize All** button and then on the **Clean** button.
8. Select the **mand** bone, go in to **Edit Mode**, and press **A** to deselect the vertices of the **head** vertex group. Select all the vertices of the **jaw**, including the **bottom teeth**; then go to the **Vertex Groups** subpanel to the right and deselect the **tongue** group by clicking on the, yes, **Deselect** button (we created the **tongue** vertex group chapters ago, during the modeling stage; otherwise, just deselect the tongue's edge-loops manually).
9. Find and select the **mand** group and click on the **Assign** button.
10. Go again in to **Weight Paint** mode and click on the **Normalize All** and **Clean** buttons under the **Weight Tools** subpanel:



The Weight Tools subpanel and the "mand" vertex group

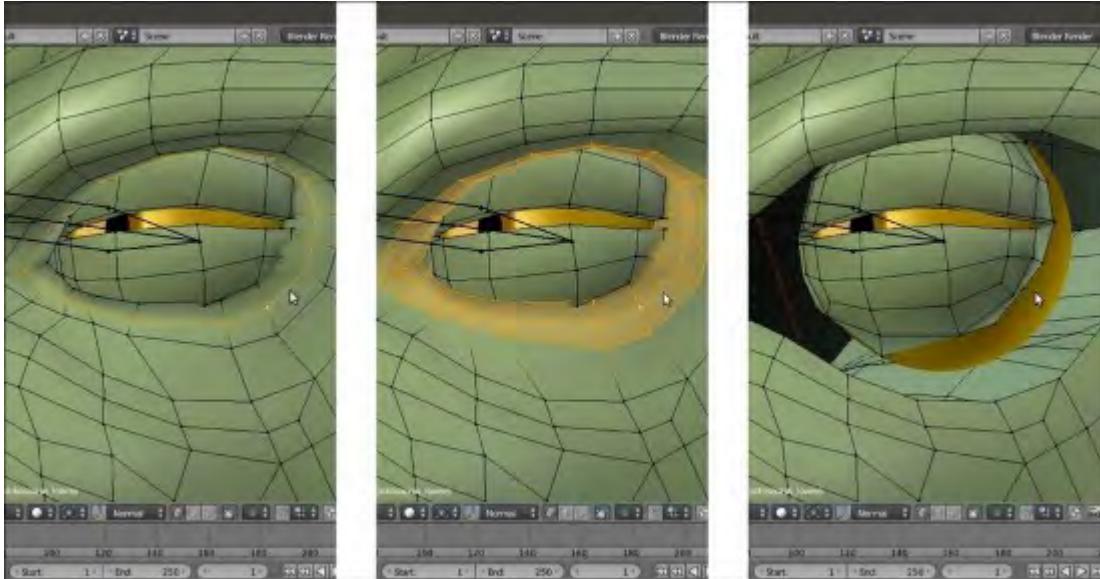
11. Now, go out of **Weight Paint** mode, select the mesh and, in the **Vertex Groups** subpanel, search for the **eyelid_upper.L** vertex group; enter **Edit Mode** and click on the **Select** button:



The selected eyelid_upper.L vertex group

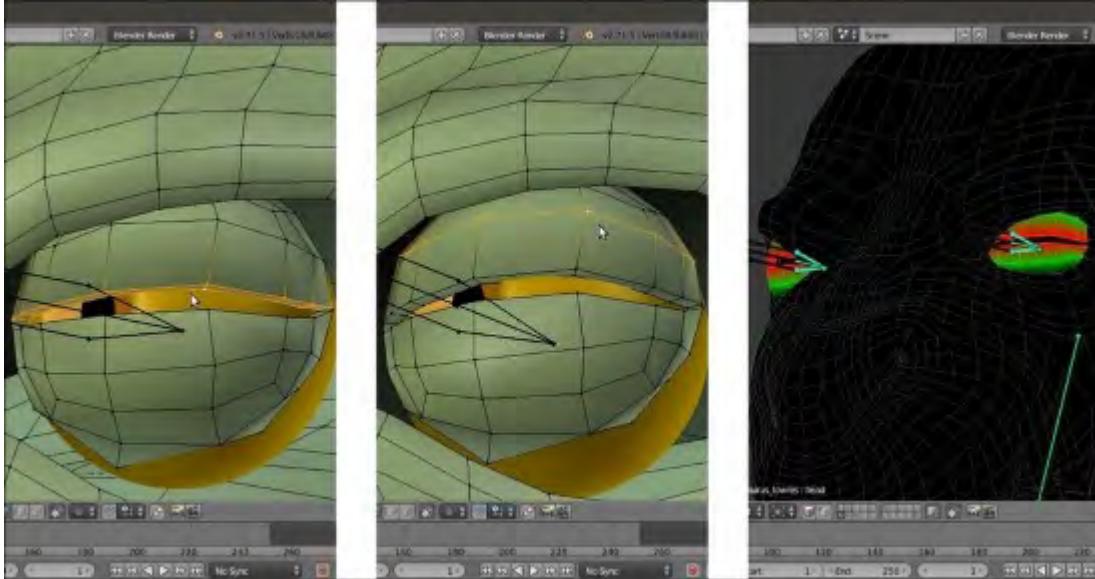
We must get rid of all these vertices erroneously assigned to the vertex group by the **Automatic Weights** tool.

12. Click on the **Remove** button and then press the **A** key to deselect everything.
13. Repeat this for the **eyelid_bottom.L**, **eyelid_upper.R**, **eyelid_bottom.R**, and also for the **eye.L** and **eye.R** vertex groups.
14. Zoom to the **eyes** area. Select an edge-loop (**Alt + right-click**) around the left **eyelids** and then press **Ctrl** and the **+** key on the numpad to extend the selection; press the **H** key to hide the selected vertices (this is simply to isolate the **eyelids** vertices for easier edge-loops selection):



Isolating the eyelids vertices

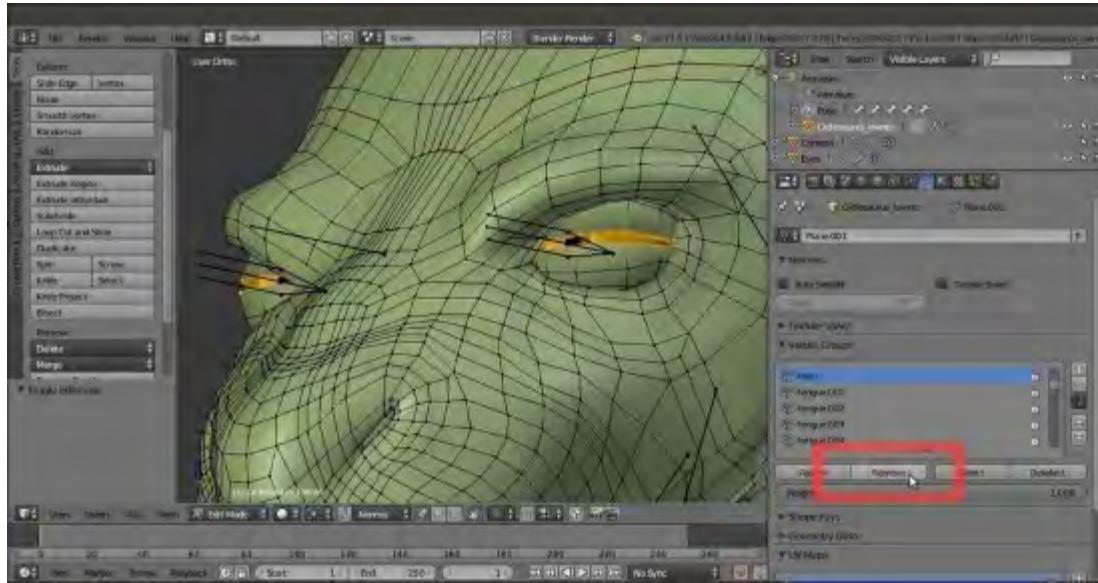
15. Select the border upper edge-loops and assign them to the **eyelid_upper.L** vertex group with a weight of **1.000**; select the second upper edge-loop and again assign it to the **eyelid_upper.L** vertex group, but with a weight of **0.500** (see the following screenshot).
16. Do the same for **eyelid_bottom.L**:



The visualization of the eyelids vertex group with different weights

In the preceding screenshot, to the right, you can see the weights of **eyelid_upper** and **eyelid_bottom** vertex groups on both sides, made visible at the same time by the **Multi-Paint** item enabled in the **Brush** subpanel under the **Tool Shelf**; here it is used only for visualization purposes.

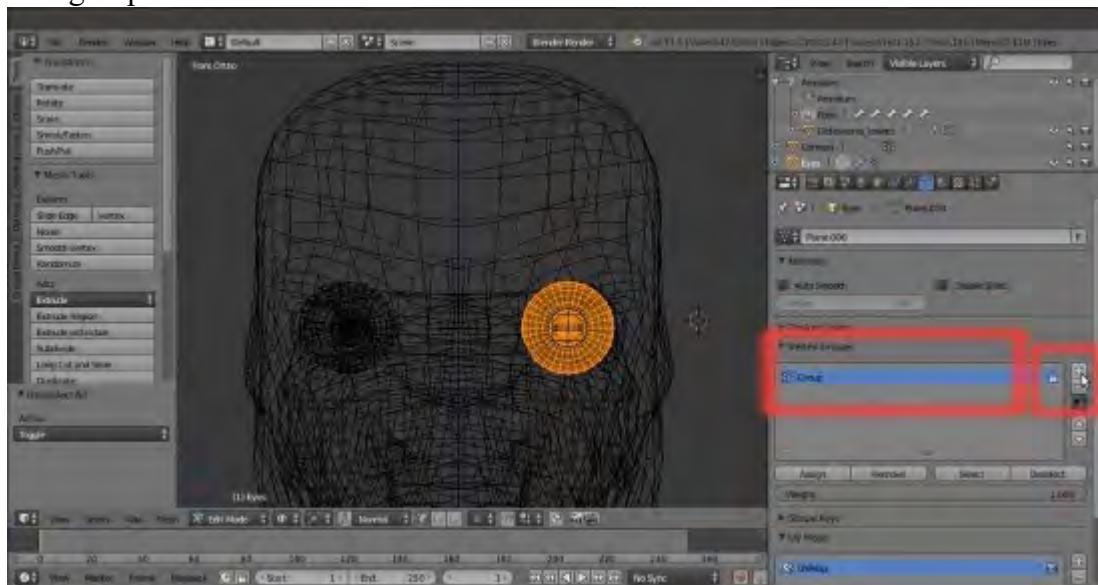
17. Repeat the procedure for the **eyelids** on the right side (**eyelid_upper.R** and **eyelid_bottom.R**).
18. In the **Vertex Groups** subpanel, select the **head** vertex group and then select the border edge-loops of both the **left** and **right eyelids**. Click on the **Remove** button to remove those vertices from the group's evaluation:



Removing the eyelids vertices from the "head" vertex group

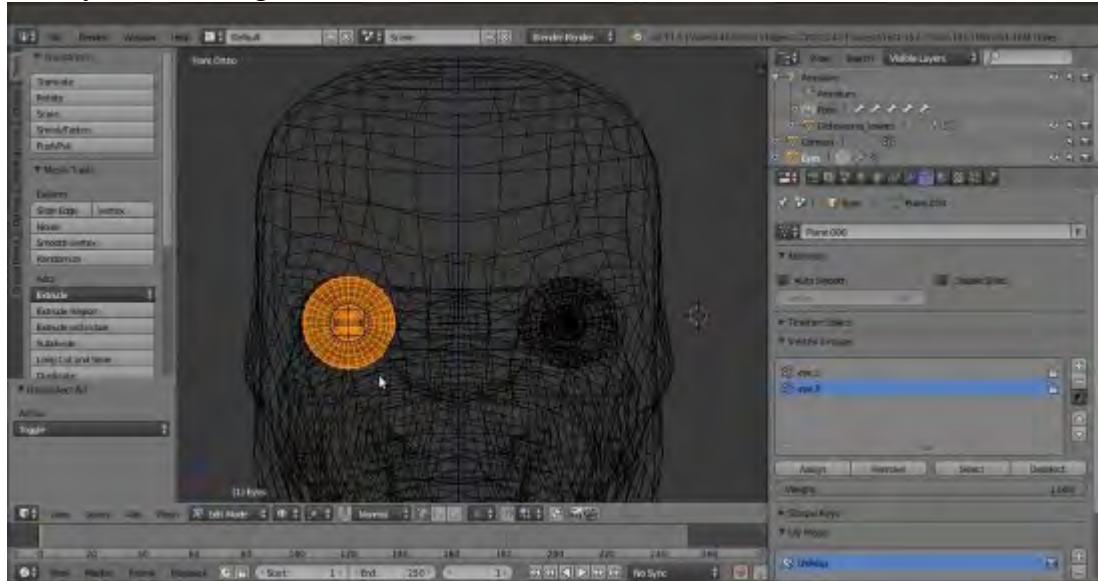
Assigning weights by hand can be a handy method also for other parts, for example, the **eyeballs**, which are separate objects from the **Gidiosaurus** mesh.

19. Go out of **Edit Mode**, and in the **Outliner** select the **Eyes** object; press *Tab* again to go in to **Edit Mode**.
20. Go in to **Front** view and box-select all the vertices of the **left eye**; then, go to the **Vertex Groups** subpanel under the **Object Data** window and click on the **+** icon to the right to create a new group:



Creating the vertex group for the eyeball

21. *Ctrl + left-click* on the name of the vertex group to rename it as **eye.L** and then click on the **Assign** button to assign all the selected vertices to the group with a value of **1.000**.
22. Deselect everything, select the vertices of the other **eye**, and create a new vertex group; rename it as **eye.R** and assign the vertices.

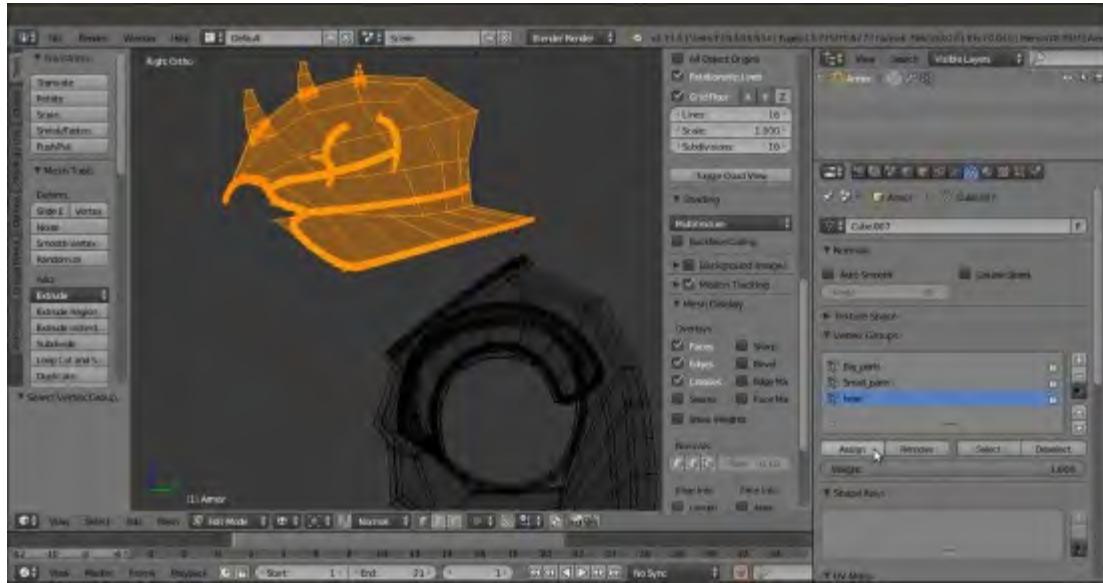


Creating the eye.R vertex group

23. Exit **Edit Mode** (*Tab*) and go to the **Object Modifier** window; assign an **Armature** modifier, move it upwards in the stack, and click on the **Object** field to select the **Armature** item as a deforming object.
24. Temporarily, unhide the **Corneas** object in the **Outliner** and repeat from step 19 to step 23, where we created the **eye.L** and **eye.R** vertex groups and assigned the appropriate mesh vertices and the **Armature** modifier.

The same process must be applied to the skinning of the **Armor** that, being a single object made of stiff elements, can be easily and ideally divided into different vertex groups; each one is skinned with the full value of **1.000**.

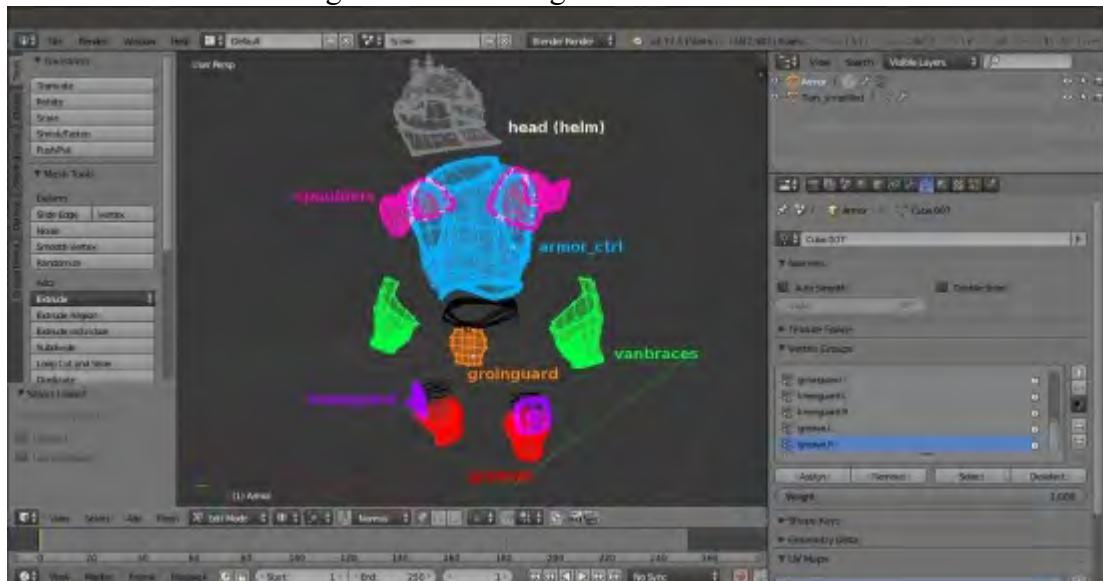
25. Click on the button to activate the **13th** scene layer and show the **Armor**; select it and enter **Edit Mode**.
26. Select all the vertices of the **helm**, including the **decorations**, create the **head** vertex group in the **Armor** mesh (remember that the name must be the one of the deforming bones), and click on the **Assign** button with a weight value of **1.000**:



The "head" vertex group for the Armor object

27. Repeat the operation with each part of the **Armor**, so creating, always with a weight of **1.000**, the vertex groups: **vanbrace.L** and **vanbrace.R** (covering the **forearms**), **greave.L** and **.R** (covering the **calves**), **groinguard** (the **front hips**), **kneeguard.L** and **.R**, **spaulder.L** and **.R**, and **armor_ctrl**.

You can use the following screenshot as a guide:



The happy colorful Armor guideline

28. Go out of **Edit Mode**; then, go to the **Object Modifier** window, assign an **Armature** modifier to the **Armor**, move it upwards in the stack, and click on the **Object** field to select the **Armature** item as a deforming object.
29. Disable the *Display modifier in viewport* button (the one with the eye icon) for the **Subdivision Surface** modifier, to speed up the 3D viewport.

For the moment, ignore the **Tiers** (which have been separated by the **Armor** object and simplified by deleting several alternate edge-loops, this we'll skin in the next recipe) and *save the file*.

How it works...

The **Normalize All** button normalizes the weights of all the vertex groups so that their sum is not superior to **1.000**; because we had assigned a weight of **1.000** to the upper head vertices, the vertices in the other groups that were interfering with the **head** deformation have been automatically set to **0.000**.

The **head** group, instead, remained the same because it was locked in the **Options** bottom panel; the **Clean** button, then, took care of removing all the unwanted vertices in the active group, restricting the inclusion of its vertices only to those with a weight greater than **0.000**.

When assigning vertices to a group, the **Weight** slider under the **Vertex Groups** subpanel can obviously be set to any value between **0.000** and **1.000**, so it's also possible to select a single edge-loop or rows of vertices and assign them at different times to the same vertex groups, but with different weight values. For example, a central edge-loop of vertices with the weight of **1.000** can be surrounded by external edge-loops with weight values of **0.750**, **0.500**, **0.250**, and so on. This is what we have done for the **eyelids** after the cleaning of the **eye sockets** area, thanks to the **Select**, **Deselect** and **Remove** buttons. Be aware that the same result can be obtained by painting and/or blurring the weights on the mesh, but we'll see this in the next recipe.

See also

- http://www.blender.org/manual/modeling/meshes/vertex_groups/index.html

Editing Weight Groups using the Weight Paint tool

Both the **Automatic Weights** parenting as well as the **Weight Groups** created and assigned by hand must, at a certain point, inevitably be edited for several reasons. As we have already seen, the parenting tool didn't do a perfect job, or maybe the transition between different weights is too sharp and must be blurred to smoothly deform the mesh. In any case, the ideal tool for this editing work is the **Weight Paint** tool.

Getting ready

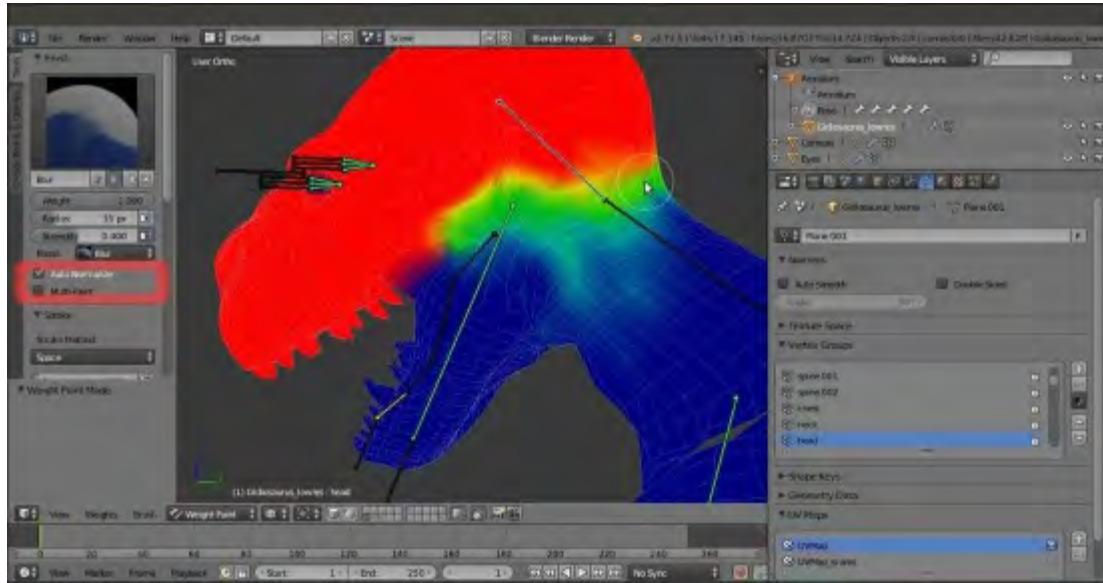
As usual, let's first prepare the scene to work on:

1. Open the `Gidiosaurus_skinning_01.blend` file and hide the **13th** scene layer.
2. Enable the **3rd Armature** layer and then deselect the **Shapes** item.
3. Press the **3** key on the numpad to go in to **Side** view; if necessary, press the **5** key on the numpad to go in to **Ortho** view and the **Z** key to go in to **Wireframe** viewport shading mode.
4. Save the file as `Gidiosaurus_skinning_02.blend`.

How to do it...

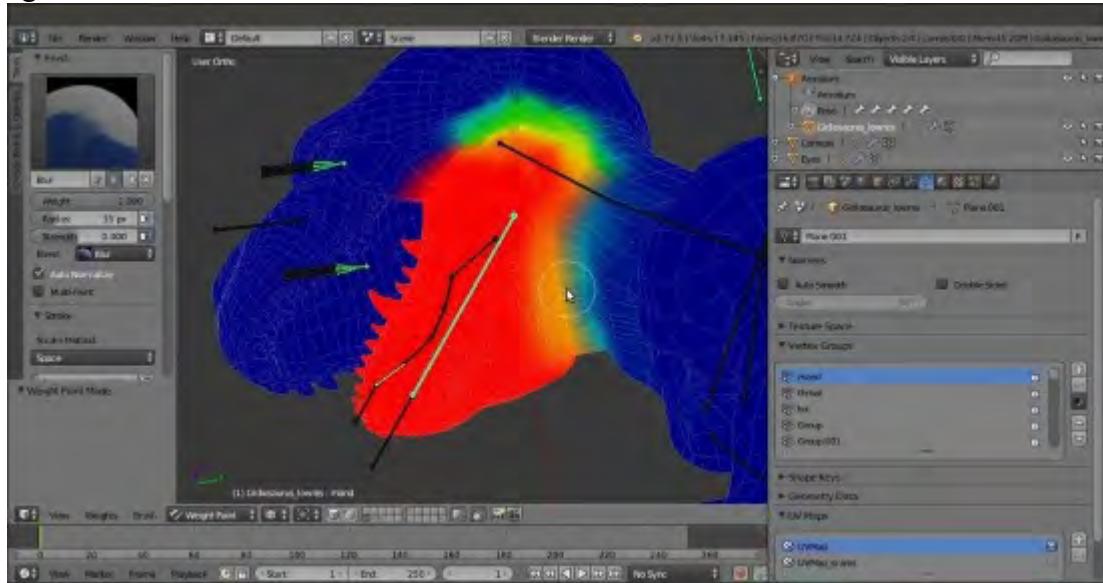
Also, now let's start with the **Weight Paint** tool itself:

1. Select the **Gidiosaurus** mesh and go in to **Weight Paint** mode; the tabs under the **Tool Shelf** on the left-hand side of the 3D window (press the **T** key in case they are not already present), change to show the **Weight Paint** tools.
2. In the viewport, right-click on the **head** bone to show the **head** vertex group on the mesh's surface.
3. Go to the **Tool Shelf** and click on the **Options** tab to verify that the **X Mirror** item in the **Options** subpanel is activated. Then, go back to the **Tools** tab and click on the big **Brush** window at the top to select a **Blur** brush; set **Weight** to **1.000** and **Strength** to **0.400**.
4. Select the **Auto Normalize** item at the bottom of the **Brush** subpanel.
5. Start to paint on the borderline of the vertex group, blurring the separation between the red and blue colors and trying to obtain, in general, a transition as smooth as possible:



Blurring the edges of the vertex group

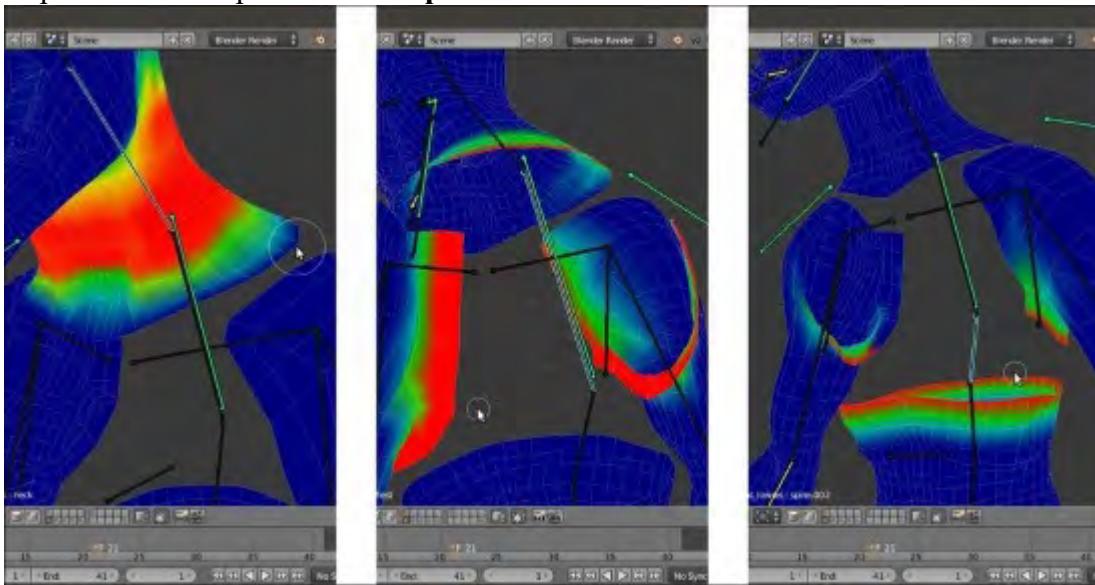
6. Switch to the **mand** vertex group by selecting the corresponding bone and smooth the transition again:



Smoothing the transition of the "mand" vertex group

7. If you need to reduce the weight of a vertex, switch the **Blur** brush with a **Subtract** one, and with a low **Strength** (**0.100** or even less) paint on it. Then, if necessary, blur the area again.
8. Alternatively, instead of using a **Subtract** brush, you can paint on the mesh with a **Mix** brush set with **Strength = 1.000** and **Weight = 0.000**.
9. Select the **neck** bone and reduce the weight of the vertices at the **neck** edges to **0.000**.

10. Select the **chest** bone and paint the vertices at the **chest** edges to **1.000**.
11. Repeat the last step also for the **spine.001** and **.002** bones:

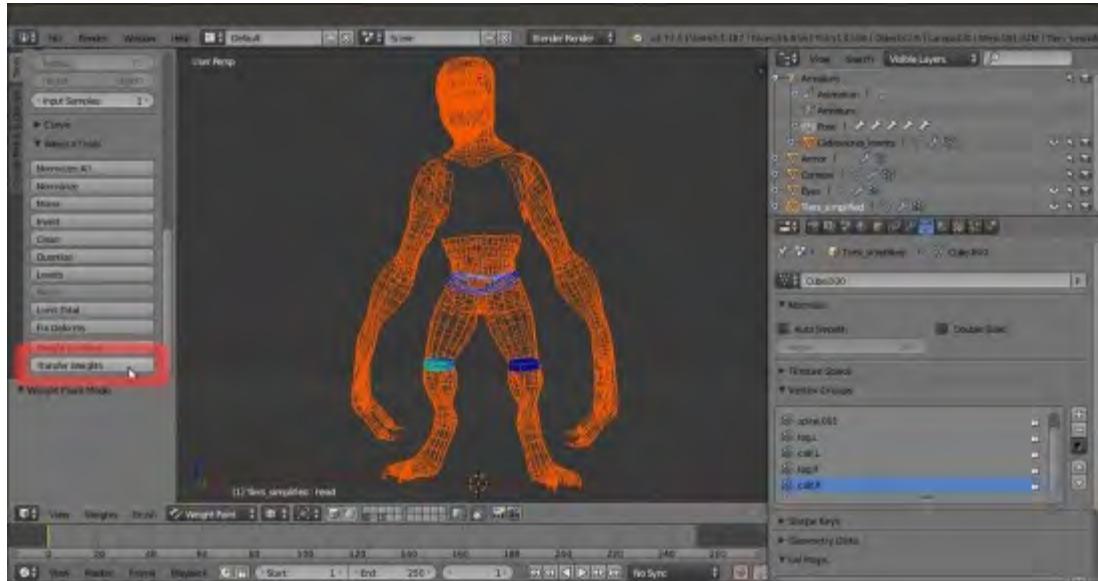


Other vertex groups

To look at exactly how the weights have been edited by the **Weight Paint** tool, open the **Gidiosaurus_skinning_03.blend** file, hide the **Armor**, select the **Gidiosaurus** mesh and press **Ctrl + Tab** to go in to **Weight Paint** mode, and then right-click to select the different bones.

One last thing still remains to be done: we must also skin the **Tiers_simplified** object.

12. Enable the **13th** scene layer to show the **Armor** and the **Tiers**; temporarily hide both the **Armature** and the **Armor** object by clicking on the respective eye icon in the **Outliner**.
13. Select the **Gidiosaurus** mesh, then, *Shift-select* the **Tiers** object and press **Ctrl + Tab** to go in to **Weight Paint** mode.
14. Go to the **Weight Tool** subpanel under the **Tool Shelf** and click on the **Transfer Weight** button, which is the last button at the bottom. After a bit of calculation, the weights of the vertices for the underlying **Gidiosaurus** mesh have been transferred to the corresponding overlaid vertices of the **Tiers** object and the vertex groups as well:



Transferring the vertex group weights from the Gidiosaurus mesh to the tiers object

15. Go out of **Weight Paint** mode and select the sole **Tiers** object. In the **Object Modifiers** window, assign an **Armature** modifier or, if you prefer, just join it to the **Armor** object (**Armor** as an active object and then press *Ctrl + J*). In both cases, just remember to enable the **Preserve Volume** item.
16. Save the file.

See also

- http://www.blender.org/manual/modeling/meshes/vertex_groups/weight_paint.html

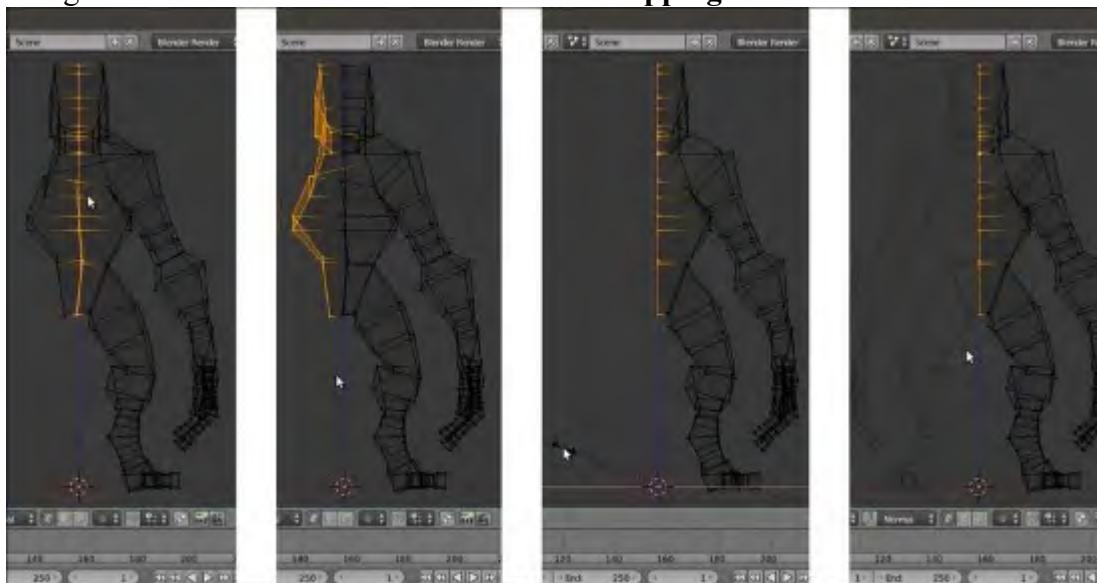
Using the Mesh Deform modifier to skin the character

The **Mesh Deform** modifier has been introduced in Blender for the production of the short open movie *Big Buck Bunny* and it's a very easy and quick way to skin medium and high resolution characters' meshes. Although the utility of this modifier really shows the skinning of fat, chubby characters, it will be useful to see the way it works even if applied to a quite skinny character such as the **Gidiosaurus**.

Getting ready

First, we must prepare the **deforming cage**, which is a simplified low poly mesh totally enveloping the character's mesh; to do this, in our case, we can start from an already made object:

1. Open the `Gidiosaurus_skinning_03.blend` file.
2. Click on the **File | Append** menu (or press `Shift + F1`), browse to the folder with all the project files, and click on the `Gidiosaurus_base_mesh_02.blend` file. Then, click on the **Object** folder and select the **Gidiosaurus** item.
3. Move the just-appended object to the first scene layer; then, go to the **Outliner** and rename it as **Gidiosaurus_cage**.
4. This is the base mesh we built in the first chapter of this book, so go to the **Object Modifier** window and apply the **Skin** modifier; delete the **Mirror** modifier and disable the **Subdivision Surface** modifier visibility in the viewport by clicking on the eye icon button.
5. Go in to **Edit Mode** and by pressing `Ctrl + R`, cut a median vertical edge-loop at the center of the mesh.
6. Select the vertices of the *missing* half and delete them; then, select the median edge-loop and, with **Pivot Point** set to **3D Cursor** (and the **3D Cursor** located at the center of the scene), scale them to **0.000** along the global **x** axis.
7. Assign a new **Mirror** modifier and enable the **Clipping** item.



Preparing the deforming cage

8. Now, enable the scene layer with the **Gidiosaurus** mesh, select it, and temporarily disable the **Armature** modifier by clicking on the *Display modifier in viewport* button (the one with the eye icon).
9. In the **Outliner**, click on the eye icon to hide the **Armature**.
10. Reselect **Gidiosaurus_cage**, enter **Edit Mode**, and start to edit. Basically, the cage must be large enough to totally include the character's mesh.
11. Select whole parts such as the **head** or a **hand** and scale the vertices on their normals (*Alt + S*) and move the vertices by hand.
12. Where necessary, add edge-loops (*Ctrl + R*) to refine the cage's shape, but try to keep it as simple and low resolution as possible.



The cage mesh in **Edit Mode**

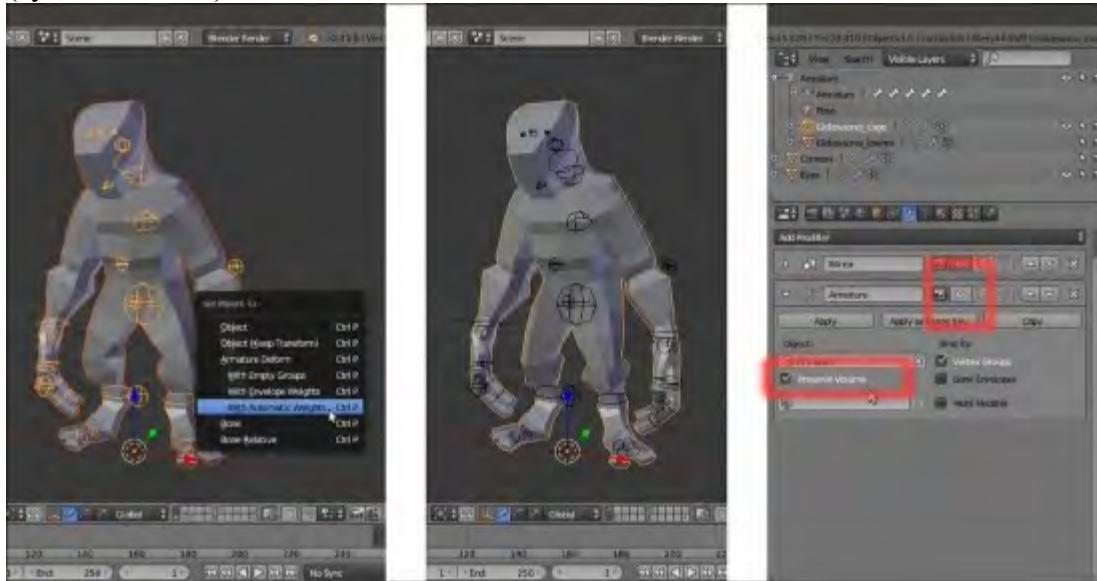
13. Once we have confirmed that the **Gidiosaurus** mesh is totally contained in the cage, we can go out of **Edit Mode**.

How to do it...

Now that the **deforming cage** is made, we can go on with the **skinning**:

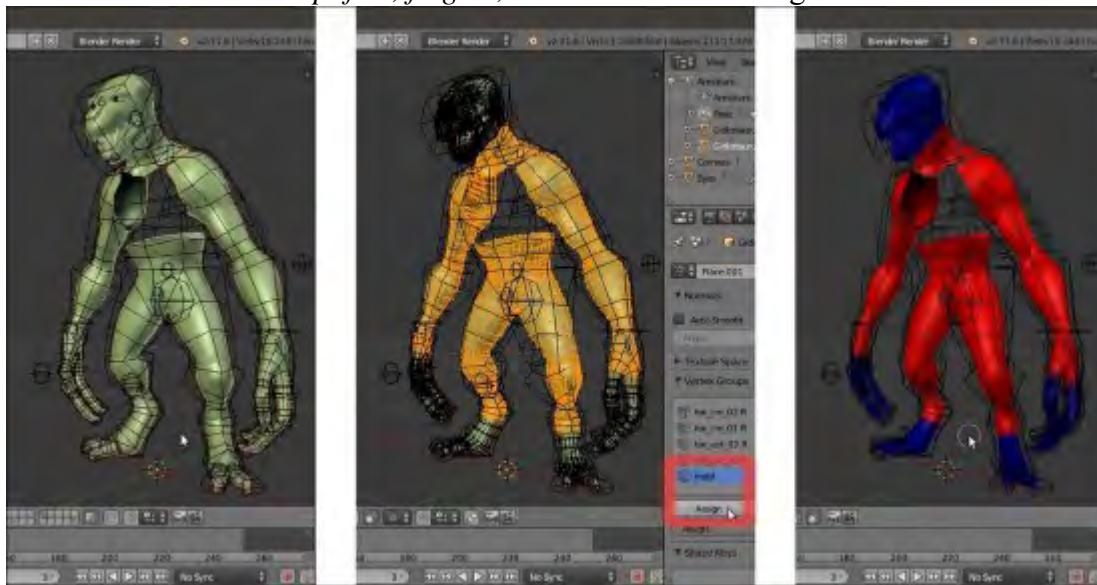
1. Unhide the **Armature** and select it. Go in to **Edit Mode**, select the bones deforming the **Armor** (see the *Parenting the Armature and Mesh using the Automatic Weights tool* recipe in this chapter for this), and press *Shift + W | Deform*. Don't deselect anything because it will be useful later on to have the bones still selected, and go straight back to **Object Mode**.
2. Now, hide the **Gidiosaurus_lowres** object; then, select the **Gidiosaurus_cage** object, *Shift-select* the **Armature**, and press *Ctrl + P | With Automatic Weights*.

3. Select the sole **Armature**, go in to **Edit Mode** and press **Shift + W | Deform**, and then switch to **Pose Mode**.
4. Reselect the **cage** and go to the **Object Modifiers** window; in the **Armature** modifier, enable the **Preserve Volume** item, but temporarily disable the visibility of the modifier in the viewport (eye icon button):



Parenting the deforming cage to the Armature

5. Go to the **Object** window and click on the **Maximum Draw Type** button under the **Display** subpanel to select the **Wire** item. Unhide the **Gidiosaurus_lowres** object.
6. Select the **Gidiosaurus** mesh and go in to **Edit Mode**. In the **Vertex Groups** subpanel under the **Object Data** window, create a new group and rename it as **mdef**; select all the vertices of the **Gidiosaurus** mesh *except feet, fingers, and the head* and assign them to the **mdef** vertex group:



The "mdef" vertex group

7. Go to the **Object Modifiers** window; in the **Armature** modifier panel, click on the vertex group empty field at the bottom (*name of Vertex Group which determines influence of modifier per point*) to select the **mdef** vertex group and then click on the *invert vertex group influence* button to the left (the one with the two arrows pointing in opposite directions). Temporarily, disable the visibility of the modifier in the viewport (eye icon button).
8. Assign a **Mesh Deform** modifier and move it upwards in the stack, before the **Subdivision Surface** modifier but after the **Armature** one.
9. In the **Object** field of the **Mesh Deform** modifier, select the **Gidiosaurus_cage** item; in **Vertex Group** again, select the **mdef** vertex group, check the **Dynamic** item box, and then click on the **Bind** button.
10. Save the file as `Gidiosaurus_mesh_deform.blend`.

How it works...

The **Gidiosaurus_cage** is a very simple mesh. Therefore, it is very easily skinned to the **Armature** (we didn't do it in our case, but obviously, when necessary, the automatic weights assigned by the parenting can be easily edited as in the *Editing the Weight Groups using the Weight Paint tool* recipe) and is therefore deforming, through the binding of the **Mesh Deform** modifier, the more subdivided **Gidiosaurus** mesh.

In fact, if everything went right, now we should have the **Gidiosaurus** body correctly deformed by the **cage** only for the vertices that belong to the **mdef** vertex group, while the **Armature**, which also deforms the **cage**, is still taking care of the vertices outside the group; to check this, just try to pose the rig and alternatively disable, in the **Object Modifiers** window, the viewport visibility of the **Armature** and **Mesh Deform** modifiers for the mesh.

Note that even we didn't apply the **Mirror** modifier to the **cage** object; the **Mesh Deform** modifier works correctly anyway, exactly like the **Armature** one.



The Gidiosaurus model posed through the Mesh Deform modifier

See also

- http://www.blender.org/manual/modifiers/deform/mesh_deform.html

Using the Laplacian Deform modifier and Hooks

One of the last modifiers introduced in Blender, the **Laplacian Deform** modifier shouldn't actually be considered as an effective tool to rig a character, but more as a tool to modify, change, or refine a default pose. Anyway, if set and used smartly it can often give interesting results, so it has been included in this chapter as well.

Getting ready

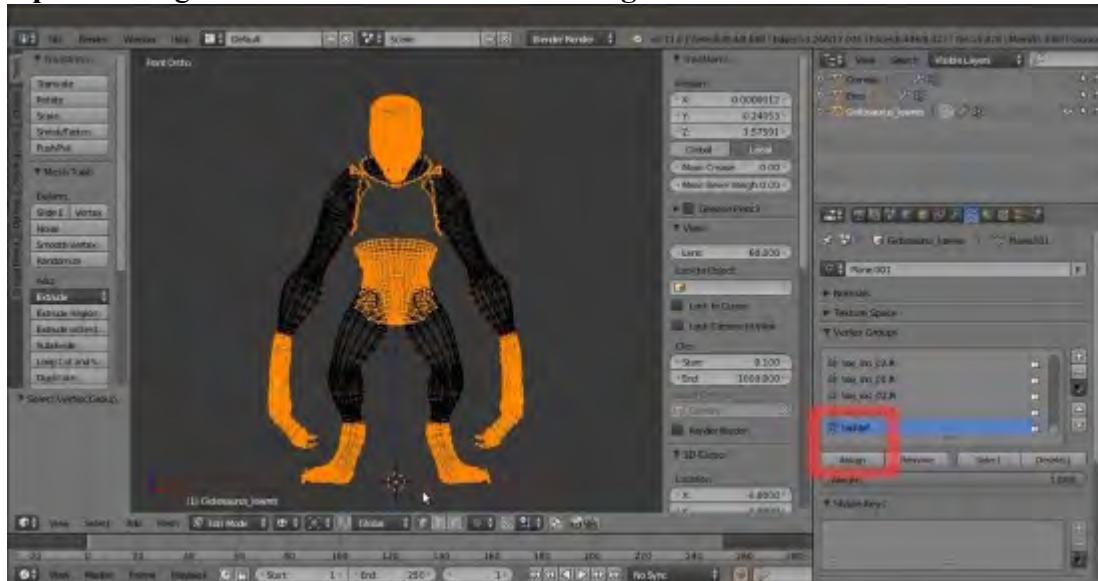
First, let's prepare the scene:

1. Open the `Gidiosaurus_rig_from_scratch_01.blend` file.
2. Select and then delete the **Armature** in **Object Mode**; then, select the **Gidiosaurus** mesh and delete the **Armature** modifier too in the **Object Modifiers** window.
3. In the **Outliner**, hide the **Eyes** object.
4. Press the Z key to go in the **Wireframe** viewport shading mode.

How to do it...

Now let's go with the **Laplacian** modifier setup:

1. With the **Gidiosaurus** mesh still in **Edit Mode**, select all the vertices of the **hands**, **feet**, **hip**, **head**, plus the **boundary edge-loops** where the mesh is missing (look at the following screenshot).
2. Go to the **Vertex Groups** subpanel and create a new group named as you wish; I named it **lapldef**. Assign the selected vertices with a **Weight** value of **1.000**:



The "lapldef" vertex group

- Now, box-deselect all the vertices, except the **head** ones; press *Ctrl + H* and in the **Hooks** pop-up menu, select the **Hook to New Object** item:



The Hooks menu

- Click on the **Select** button under the **Vertex Groups** subpanel to the right of the screen and then deselect all the vertices, except the **right hand** ones. Again, press *Ctrl + H* and in the **Hooks** pop-up menu, select the **Hook to New Object** item.
- Click on the **Select** button again, deselect all the vertices, except the **left hand** ones, and repeat the procedure:



The Hook assigned to the left hand vertices

6. Repeat the procedure separately for the left and the right **feet** and then go out of **Edit Mode**.
7. In the **Outliner**, rename the **Empties** (the **Hooks**) respectively as **Empty_head**, **Empty_hand.L** and **.R**, and **Empty_foot.L** and **.R**.



The Hooks assigned to the mesh's vertices

8. Select the **Gidiosaurus** mesh and go to the **Object Modifiers** window. Collapse all the five **Hook** modifiers for better visibility and assign a **Laplacian Deform** modifier; move it upwards in the stack, just before the **Subdivision Surface** modifier (*but always after the Hook* modifiers). Click on the **Anchors Vertex Group** to select the **lapldef** vertex group and then click on the **Bind** button.
9. For better visibility, select each **Hook** and in the **Object Data** window, set the size to **0.40**.
10. Enable the **3D manipulator widget** in the 3D view toolbar (or press **Ctrl + Spacebar**), **Shift**-click on the **Translate** and **Rotate** buttons, and set **Transform Orientation** to **Normal**.
11. Select the **Hooks** and start to move and rotate them using the **3D manipulator widget**, to pose the **Gidiosaurus** mesh:



The Gidiosaurus mesh posed through the Hooks and the Laplacian Deform modifier

12. Save the file as `Gidiosaurus_laplacian.blend`.

How it works...

Remember that because they don't work through joints, the **Laplacian Deform** modifier and the **Hooks** don't give a realistic deformation and should be used more to tweak a character pose only inside a limited range. Building a more complex rig, also with **Hooks** at the **elbows** and **knees**, is possible but probably more useful for other types of *unreal* characters' shapes.

It should also be remembered that the **Hooks**, once moved out of their location, can't be simply moved back to their original position by the **Alt + G** shortcut because this command would set them at their original **0, 0, 0** location. Instead, any rotation can be easily removed by the **Alt + R** shortcut.

In any case, the **Ctrl + Z (Undo)** shortcut can be used, but first check the number of **Steps** set in the **User Preferences** panel under the **Global Undo** item (there are only **32** by default).

See also

- <http://www.blender.org/manual/modifiers/deform/hooks.html>
- http://www.blender.org/manual/modifiers/deform/laplacian_deform.html

Chapter 8. Finalizing the Model

In this chapter, we will cover the following recipes:

- Creating shape keys
- Assigning drivers to the shape keys
- Setting movement limit constraints
- Transferring the eyeball rotation to the eyelids
- Detailing the Armor by using the Curve from Mesh tool

Introduction

In this chapter, we'll see how to create and add **shape keys** (the Blender term for **morphing**) to the model, to create facial expressions for the **Gidiosaurus** and to add shape modifications in a non-destructive way to the model.

Then, we'll see how to set a limit to the **Armature** bones' rotation using constraints and how to slightly transfer a portion of the rotation movement of the **eyeballs** to the covering **eyelids**.

Last, we'll add some detail to the **Armor** by quickly adding **rivets** through a simple and effective technique.

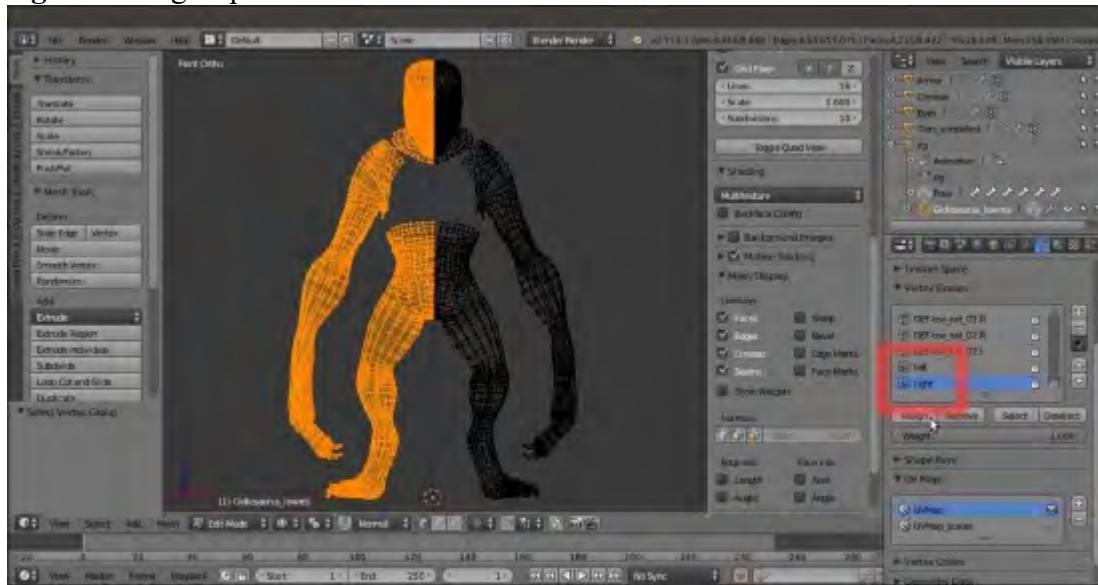
Creating shape keys

In this recipe, we'll set the **shape keys** to create (even if limited) **facial expressions** and to fake the stretching and the contracting effect of the character's **arm muscles**, and we'll add some more shape keys to fix issues in the character's shape.

Getting ready

First, let's prepare a bit the scene and the model:

1. Start Blender and load the `Gidiosaurus_skinning_rigify.blend` file, which is the same as the `Gidiosaurus_skinning_03.blend` file but with the **rig** created by the **Rigify** add-on (and later edited to add the other bones exactly as explained in the last chapter's recipes).
 2. In the **Outliner**, click on the respective eye icons to hide the **Armor**, **Eyes**, and **Tiers_simplified** objects and the **Armature**, whose name in this case is **rig**.
 3. Select the **Gidiosaurus_lowres** object and press the **1** key on the numpad to go into the **Front** view.
 4. Press **Z** to go into the **Wireframe** viewport shading mode and the **5** key on the numpad to switch to the **Ortho** view if it is not already.
 5. Enter **Edit Mode** and box-select the **left half** of the mesh vertices (including the **middle vertical edge-loop**) and in the **Vertex Groups** subpanel under the **Object Data** window, create a new vertex group; rename it **left** and assign the selected vertices a weight of **1.000**.
 6. Deselect everything and repeat this process for the **right half** of the mesh's vertices to create the **right** vertex group:



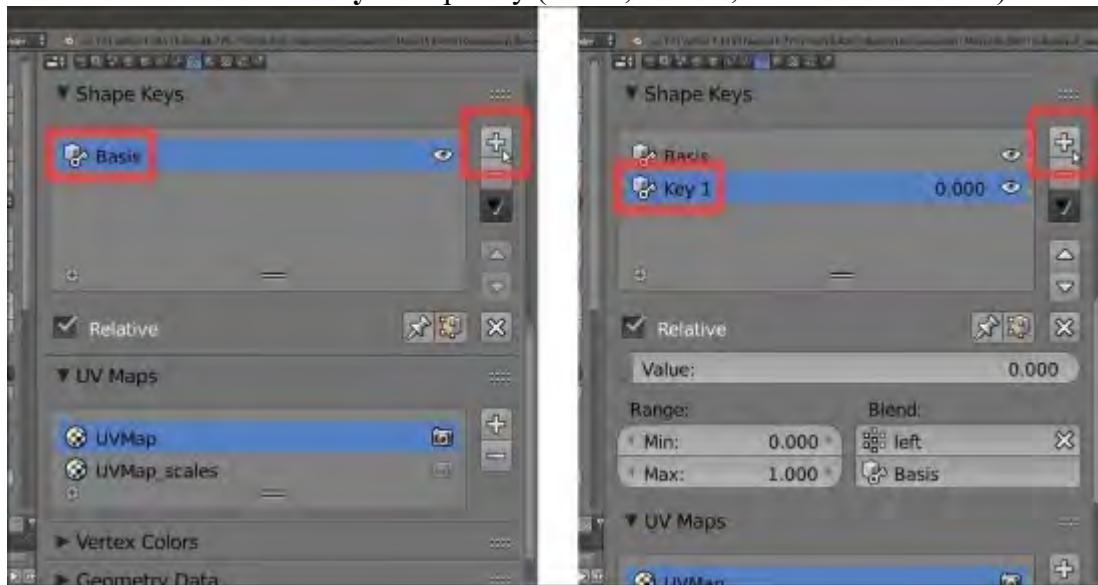
Assigning the mesh's right side vertices to the "right" vertex group

7. Save the file as `Gidiosaurus_shapekeys.blend`.

How to do it...

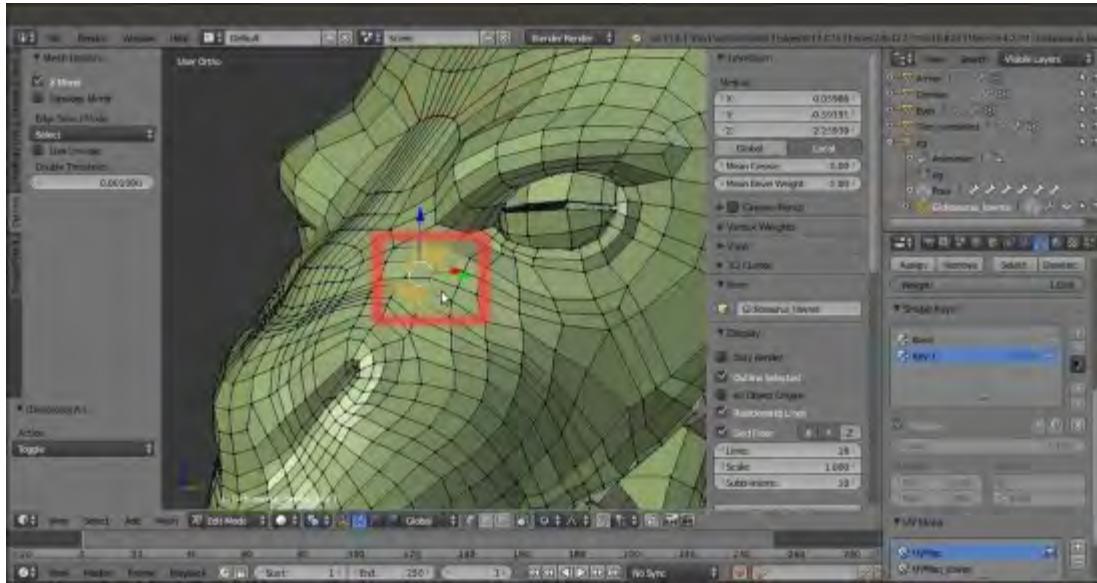
Let's start now by creating the **facial expressions** shape keys:

1. Exit **Edit Mode** and expand the **Shape Keys** subpanel under the **Object Data** window; click on the + icon button to the top left to create the **Basis** shape key (that mustn't be edited), then click once more to create the **Key 1** shape key (that is, instead, the one to be edited):



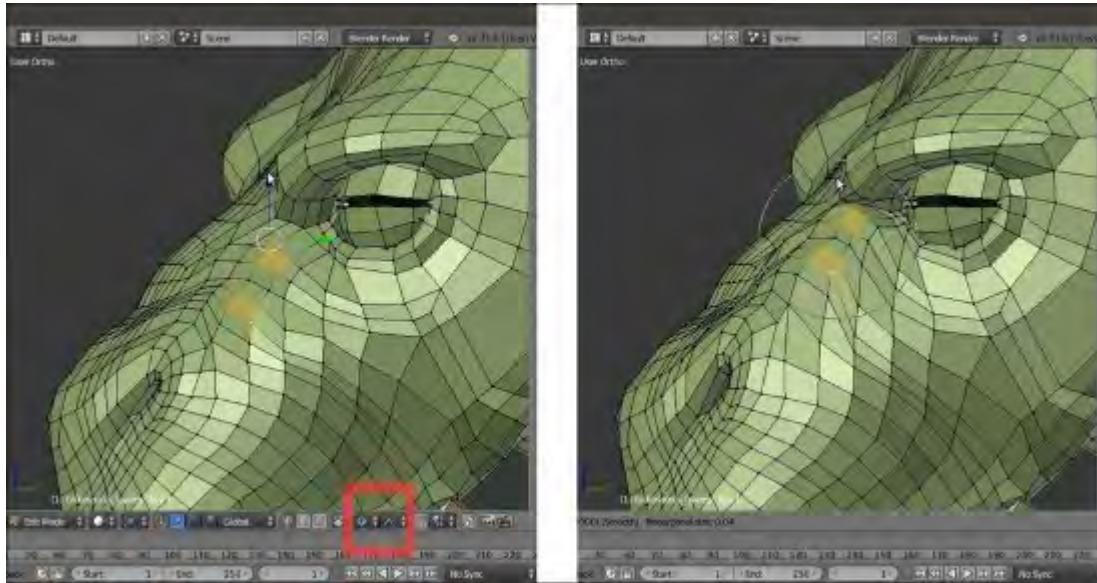
Creating the Basis and the first shape key

2. Be sure that the **X Mirror** item in the **Mesh Options** tab under the **Tool Shelf** is activated and click on the **PET (Proportional Editing Tool)** button in the 3D viewport toolbar (or activate it by pressing the **O** key).
3. Zoom to the **Gidiosaurus** head and again in **Edit Mode**, select some of the vertices at the center of the **snout** area, as indicated in the following screenshot:



Selecting the vertices

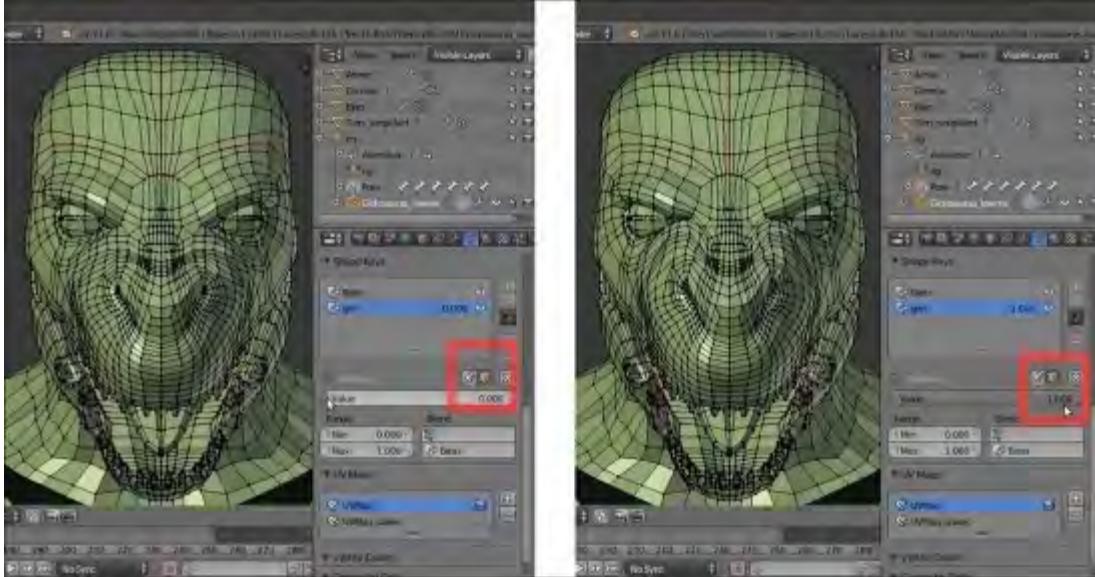
- Move them upwards and towards the eye, set the amount for the **PET** smoothing by scrolling the mouse wheel:



Moving the selected vertices slightly backwards and upwards

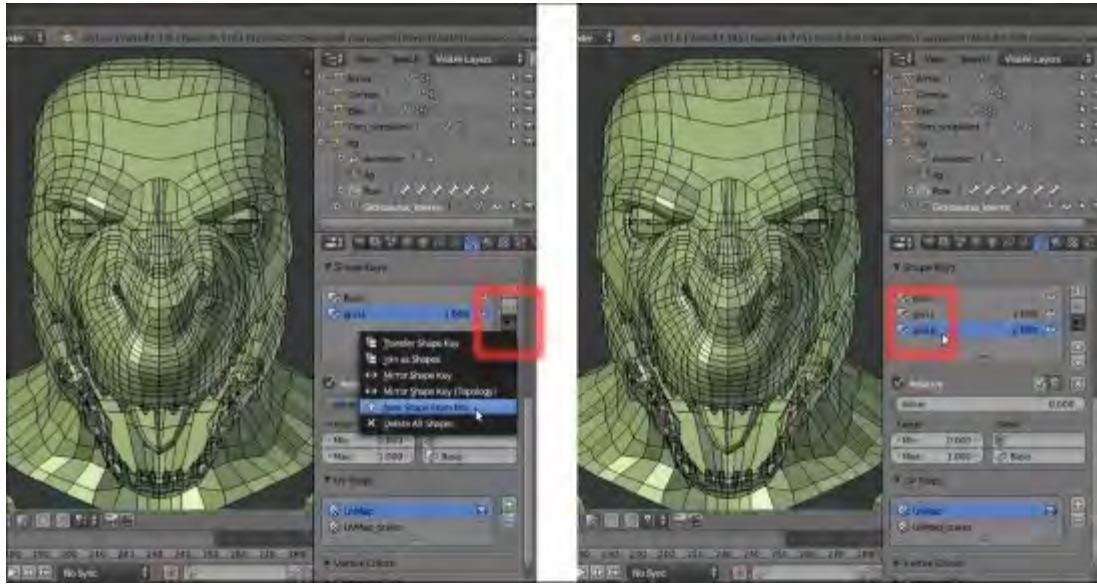
Note that we selected two faces instead of the folder edges, because the muscular *scrunching* involves both movement of the skin and a slight folding of the skin as well; the middle edge does not move as much as the selected faces due to **PET** falloff, hence creating a very slight scrunch and a more *naturalistic* skin sliding.

5. If necessary, adjust the position of the single vertices, maybe also disabling the **PET**.
6. Go to the **Shape Keys** subpanel and rename the **Key 1** shape key as **grin**.
7. Click on the *Apply shape key in edit mode* button located right above the **Value** slider to enable it; go into the **Front** view and by moving the **Value** slider from **0.000** to **1.000** and back, check for the correct working of the shape key on both the sides of the character:



Checking how the "grin" shape key works

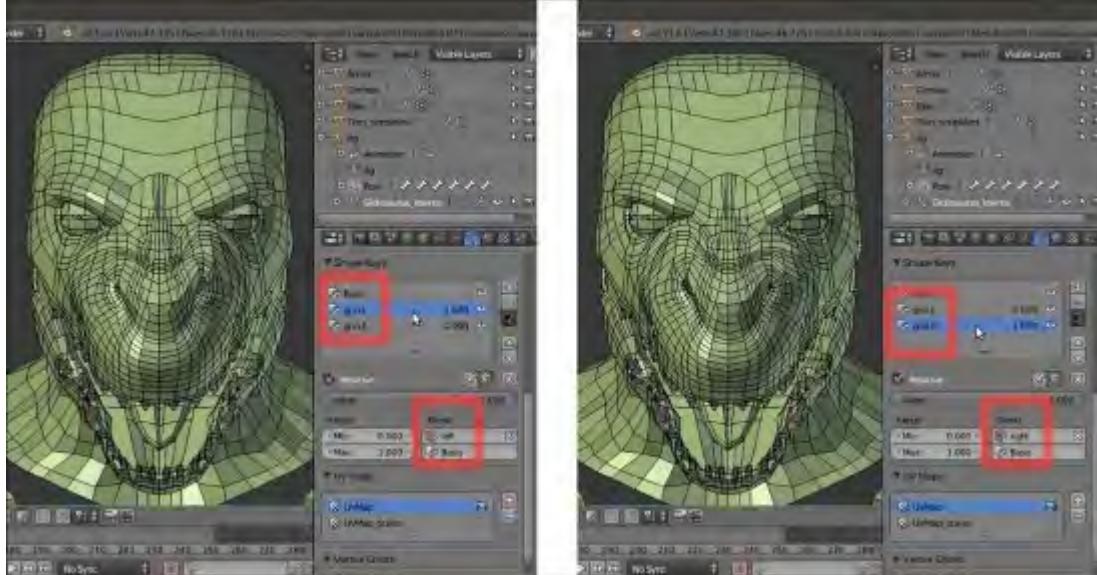
8. Go out of **Edit Mode** and just to have better visibility of the shape key modifications, go to the **Object** window to enable the **Wire** item in the **Display** subpanel.
9. Go back to the **Object Data** window and click on the button that has an icon of a downward pointing arrow (*Shape keys specials*); from the pop-up menu select the **New Shape from Mix** item: this adds a new shape key made by the sum of all the active shape keys. In this case, it is just a perfect copy of the sole **grin** shape key (the **Basis** shape key is, well, just the base starting position of the vertices in the mesh). Rename the two shape keys as **grin.L** and **grin.R**.



Copying the "grin" shape key to a new one

10. Click on the **grin.L** shape key, set the **Value** slider back to **0.000**, and then click on the **Vertex weight group** slot, the one under the **Blend** item and above the **Basis** one, and select the **left** vertex group.
11. Repeat for the **grin.R** shape key by selecting the **right** vertex group, and then once again set the **Value** slider to **1.000** and ensure it works correctly.

Each shape key now works only on the respective side, according to the selected vertex group:

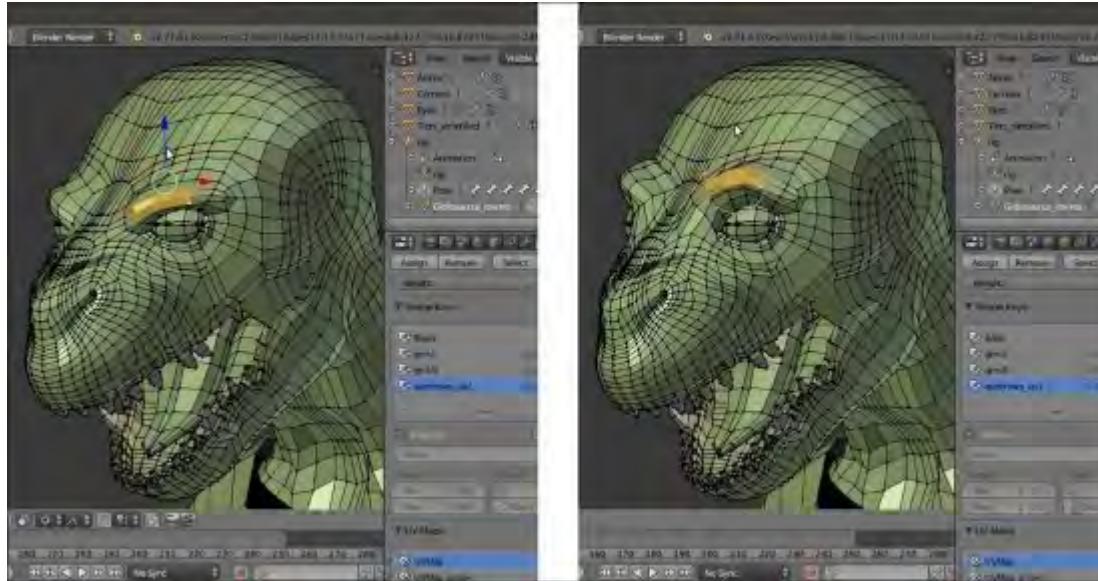


The two "grin" shape keys for the right and the left sides

This was for the **grin** expression; now we need to add at least two or three more kinds of shape keys, namely: two for the **eyebrows** (up and down) and one for the **nostrils**, multiplied for each side.

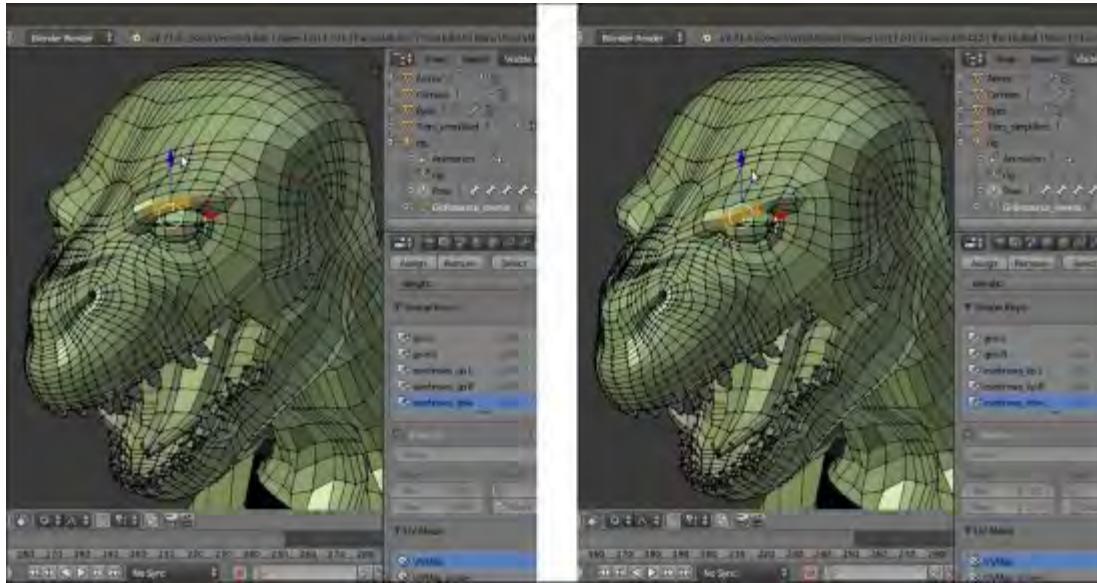
This means **six** more shape keys in total, but as you have seen, the procedure is quite quick and simple.

12. Set the **Value** slider for the **grin.L** and **grin.R** shape keys back to **0.000** and click on the **+** icon button to add a new shape key.
13. Rename it **eyebrow_up.L** and enter **Edit Mode**; grab some vertices on the left eyebrow and, still with the **PET** activated, move them upward; you can use the mouse wheel to set the influence of the **PET**:



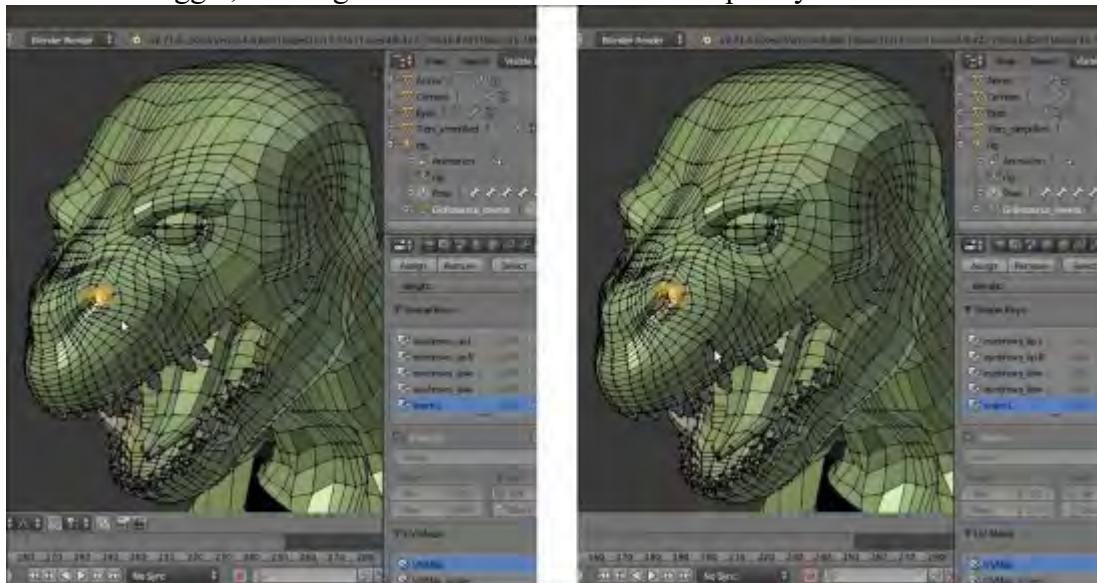
Moving the eyebrow upward

14. Repeat the steps from 9 to 11 to create the **eyebrow_up.R** shape key.
15. Repeat the steps from 3 to 11 to create the **eyebrow_down.L** and **eyebrow_down.R** shape keys:



Moving the eyebrow downward

16. Finally repeat step 3 to step 11, this time selecting the vertices around the **nostrils** and scaling them to be bigger, creating the **snare.L** and **snare.R** shape keys:

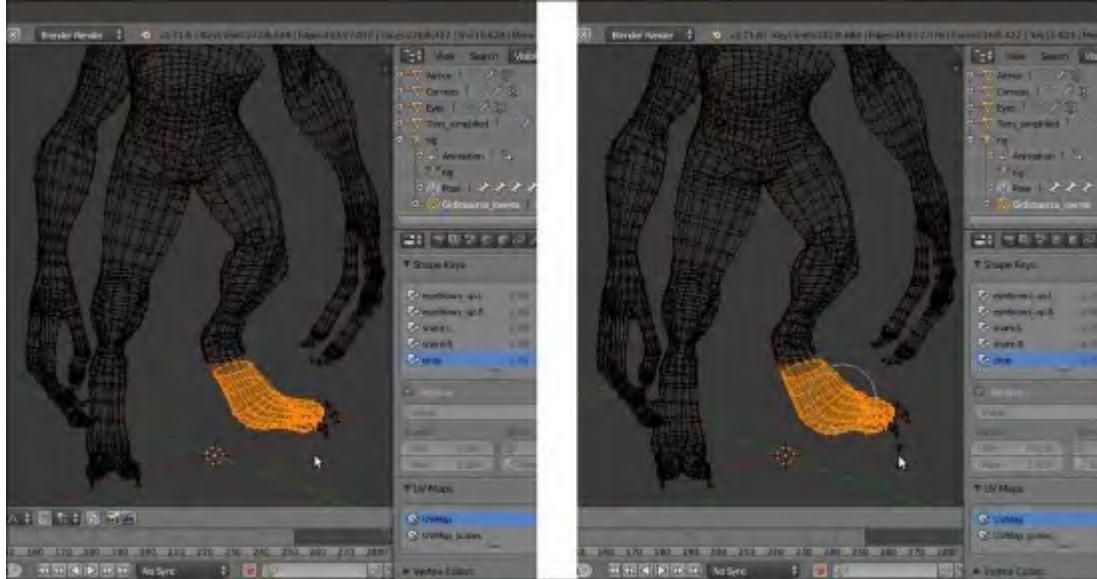


The nostril flaring

Note that, when naming the shape key for the enlargement of the nostrils, I erroneously wrote **snare**; it should have been something like *snarl* or *flaring*, but in the end it's just a naming convention and therefore, this little mistake doesn't pose a real problem.

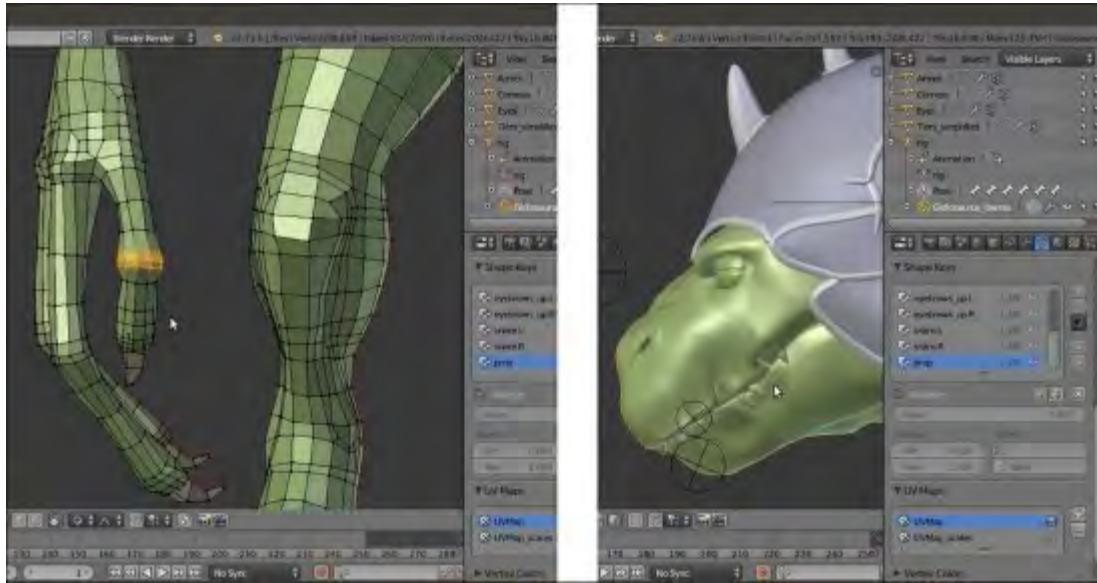
We are done with the facial expressions; now let's add one more shape key to enhance some of the body features of the **Gidiosaurus** a bit; these are not meant to be animated during the animation, but are simply a way to apply non-destructive modifications to the model.

17. Add a new shape key and rename it **prop** (for *proportions*).
18. In **Edit Mode**, select the vertices of the **left foot**, excluding the **feet talons**, and press **Alt + S** to scale them *on their normals*; if you are using the **PET**, just be sure to be in the **Connected** mode (so that the **PET** has influence only on the vertices connected to the selected ones, otherwise the unselected **feet talon** vertices will also be modified):



Modifying the feet proportions

19. Disable the **PET** and adjust the transition between the scaled vertices and the surrounding ones, the area between the two **toes**, and so on.
20. If you wish, you may also make additional modifications; in my case, besides the bigger **feet**, I simply enhanced the **knuckles** on the **hands** and at the **fingers'** joints. Also, I tweaked the rim shape of the upper and bottom borders of the **mandibles** a bit:

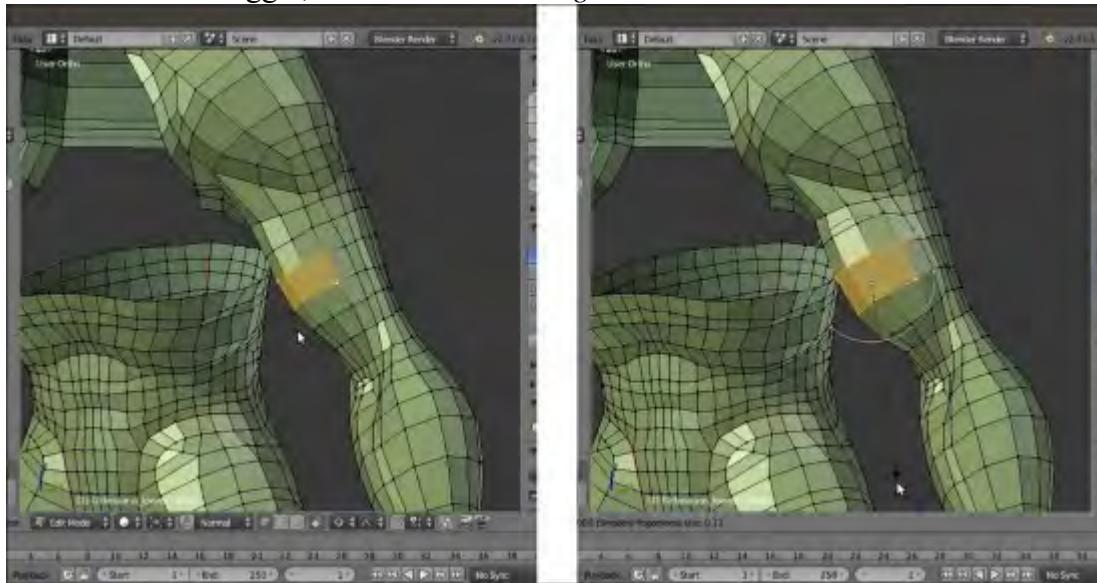


Enhancing some of the character's features

21. When you are done, set the **Value** slider of the **prop** shape key to **1.000**.

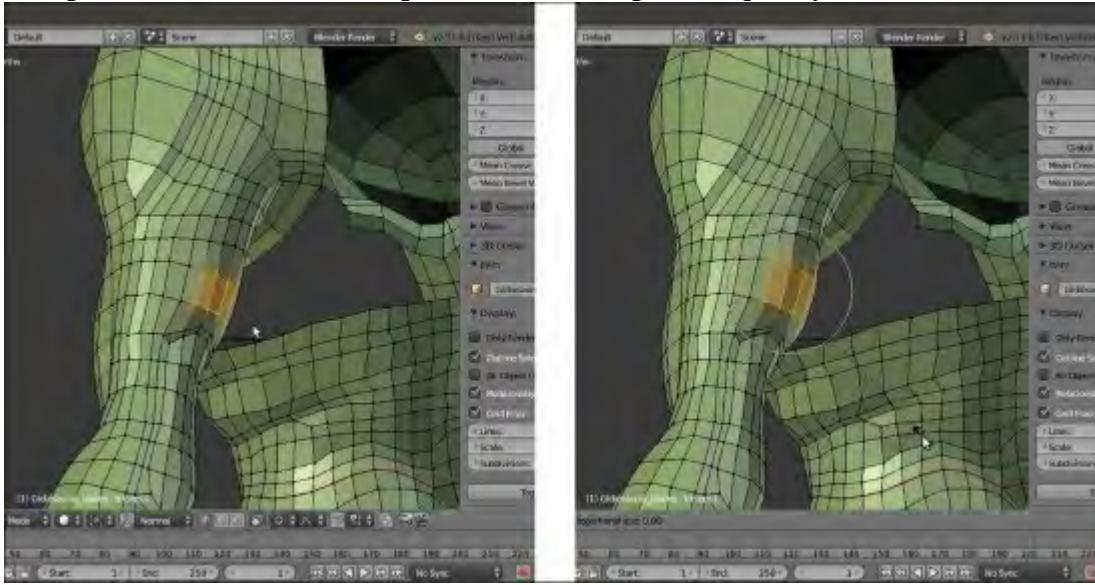
Now, let's add a couple of shape keys to mimic the movement of the main muscles of the arms, specifically of the **biceps** and of the **triceps** muscles:

22. Add a new shape key, then go to the **Gidiosaurus** mesh; select some of the vertices in the middle area of the **bicep** muscle and after enabling the **PET** again, move them forward and also scale them to be bigger, to make the muscle grow:



Making the bicep muscle grow

23. Rename the shape key **bicep.L**, and then repeat the steps from 9 to 11 to create the **bicep.R** shape key.
24. Add a new shape key and repeat everything by selecting vertices on the back of the arm, in the **triceps** area, to create the **triceps.L** and the **triceps.R** shape keys.



Making the triceps muscle grow

25. Leave the values of these last four shape keys as **0.000** and exit **Edit Mode**.

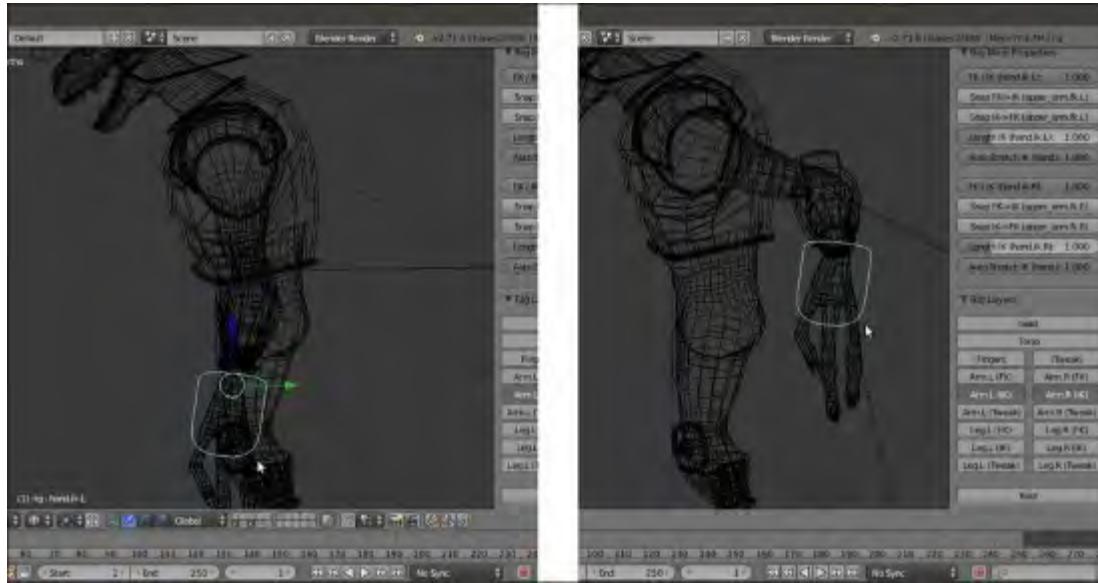
More shape keys could be added to simulate a complete muscle system, but in our case we stop here with the **Gidiosaurus** mesh; now let's concentrate on the **Armor**.

26. Go to the **Outliner** and unhide both **Armor** and **rig**.
27. Select the **rig** and then press **N** to call the **Properties** 3D view sidepanel; scroll to the bottom and first go to the **Rig Main Properties** subpanel to set both the slider for **FK/IK (hand.ik.L)** and **(hand.ik.R)** to **1.000**, then go down to the **Rig Layers** subpanel and deselect everything except for the **Arm.L (IK)** and **Arm.R (IK)** buttons.

Note

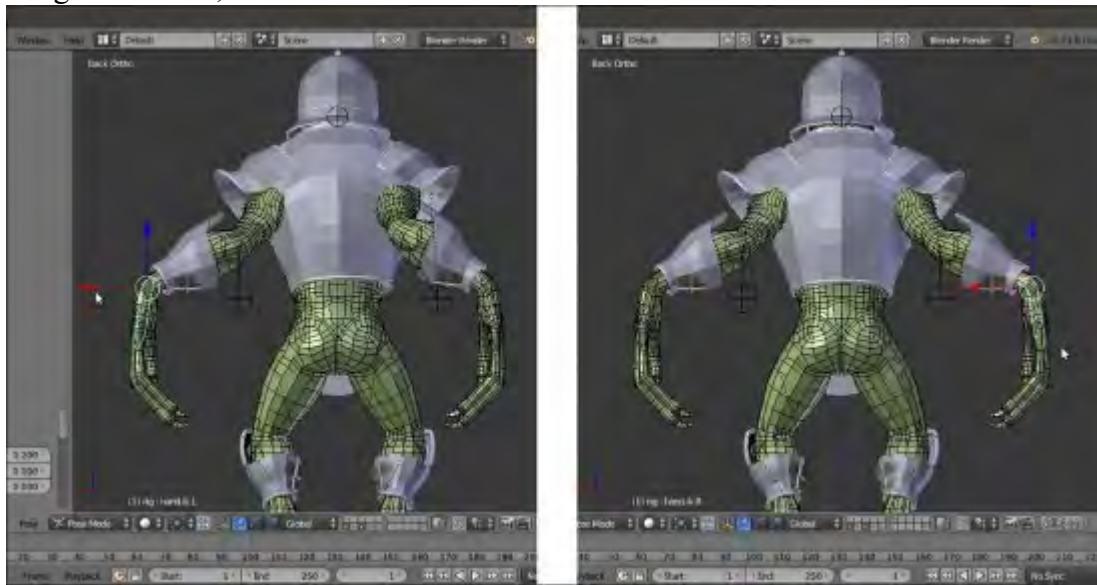
Note that if the **FK/IK sliders** don't appear, it's because you have to select one of the hand (**ik** or **fk**) bones in the viewport first.

28. Go to the 3D viewport and select the **hand.ik.L** and **hand.ik.R** handle bones, go into the **Side** view and move them towards the upper back.



Moving the arm control bones backward using the Inverse Kinematics

29. Now press **Ctrl + numpad 1** to go in **Back view** and move the **hand.ik.L** bone to **0.200** along the global **x** axis; select the **hand.ik.R** bone and move it to **-0.200**:

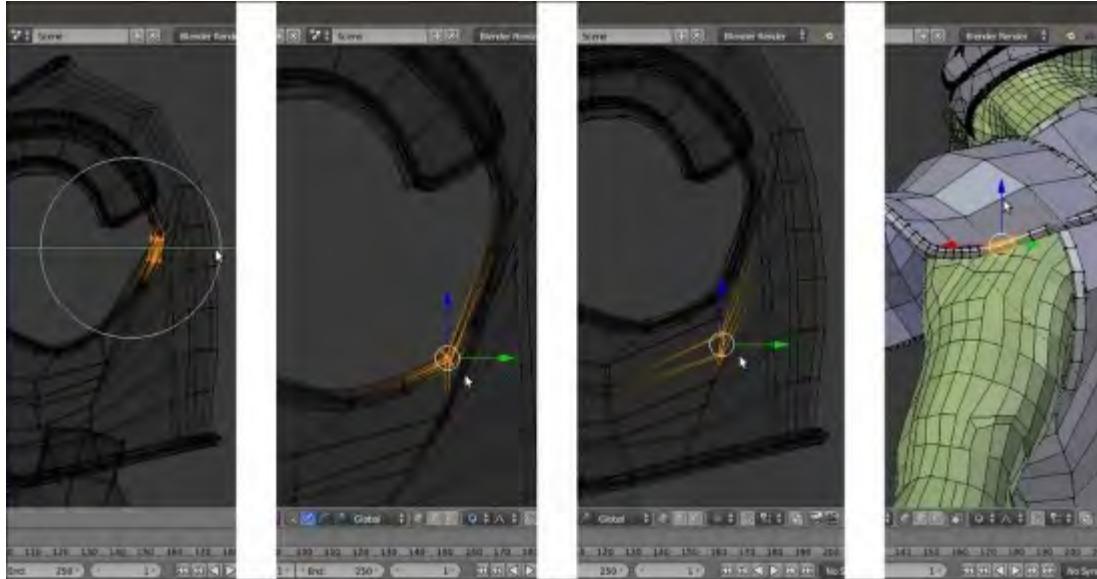


Adjusting the lateral position of the arms

30. Now select the **Armor** object and add the **Basis** shape key in the **Shape Keys** subpanel under the **Object Data** window, and then add **Key 1**.
31. Enter **Edit Mode** and press the slash key (/) on the numpad to go in **Local** view; this way only the selected objects are visible in the viewport, in our case only the **Armor**. Using the **Proportional Editing** mode, start to enlarge the back section of the opening for the **arms**,

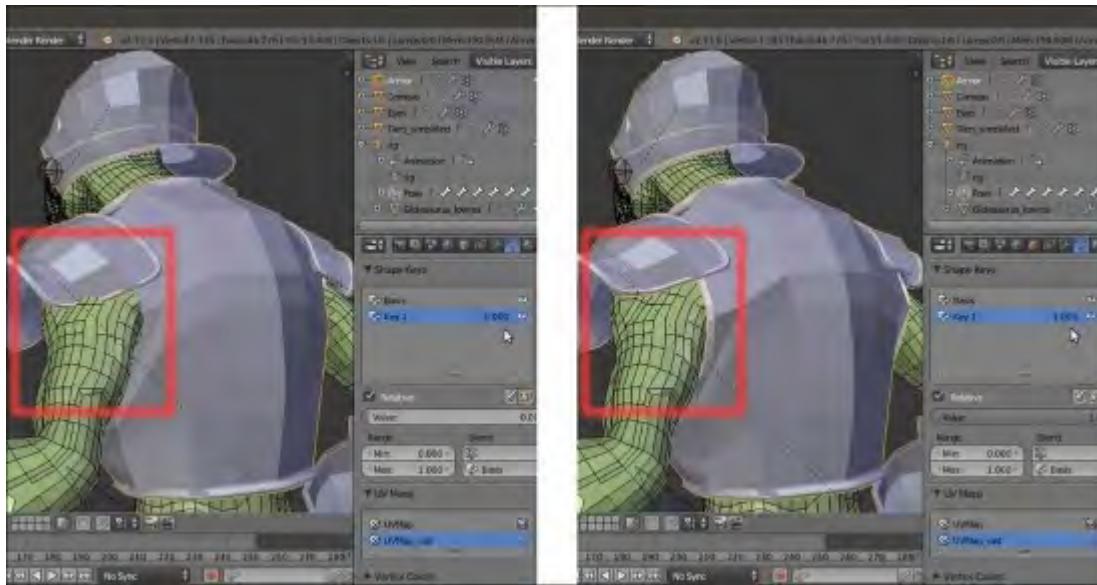
adjust the position of the surrounding vertices as required, and also raise the back vertices of the **spaulders** a bit, to avoid interpenetration with the borders of the **chest plate** and the **shoulder** as well.

32. Press the numpad slash (/) key again to go out of the **Local** view and check for the correction with the **Gidiosaurus** mesh:



Editing the position of the Armor vertices for a new shape key

33. When you are done, just exit **Edit Mode** and set the **Value** slider of the **Key 1** shape key for the **Armor** to **1.000**:



The shape key working as a fix for the Armor

34. Rename the **Key 1** shape key as **Armor_fix** and save the file.

How it works...

Although technically there are no differences, we could say that we created three different types of **shape key**:

- One type to fix shape errors or make improvements in the mesh, for example, with the **prop** and the **Armor_fix** shape keys
- The second type to modify the mesh only at certain established moments during the animation process, in our case just to animate facial expressions
- The third type to simulate muscle movements in the character

Shape keys work in **linear space**; that means that it's not possible to make vertices rotate around a pivot through a shape key, but only to move them *from point A to point B*. That's why we didn't use shape keys for stuff like the **eyelid** movements, for example, or the opening/closing of the **jaw**, but only for actions including muscles sliding above the bones such as the **eyebrows**, the **grin**, and the **nostrils**, as well as the **bicep/triceps** movements.

Thanks to shape keys, we also made *last minute* modifications and improvements to the **Gidiosaurus** mesh and to the **Armor**; being included inside a shape key, all these modifications are *non-destructive* and can be turned on or off at will, or their influence can be set at an intermediate strength value.

When modifying a mesh using a shape key, be careful not to change too much of the mutual proportions of articulated parts of a *to-be-deformed* mesh; for example, it's usually problematic to scale a whole part such as the **hands** or the **head** of a character, both smaller or bigger, unless you also scale the corresponding bones of the rig and the joints' position accordingly as well.

Beyond a certain threshold, the bones of the **fingers**, or of the **eyes** and **jaw**, start to be *out of register* compared to the respective mesh's edge-loops and you'll have to fix this by re-positioning the joints of the **Armature's** bones (just in case, remember: always do this in **Edit Mode**).

In our example, with the **prop** shape key, we just restricted ourselves to enhance the hands' **knuckles** and to make stronger **feet** by simply making the vertices' positions grow in the direction of their normals.

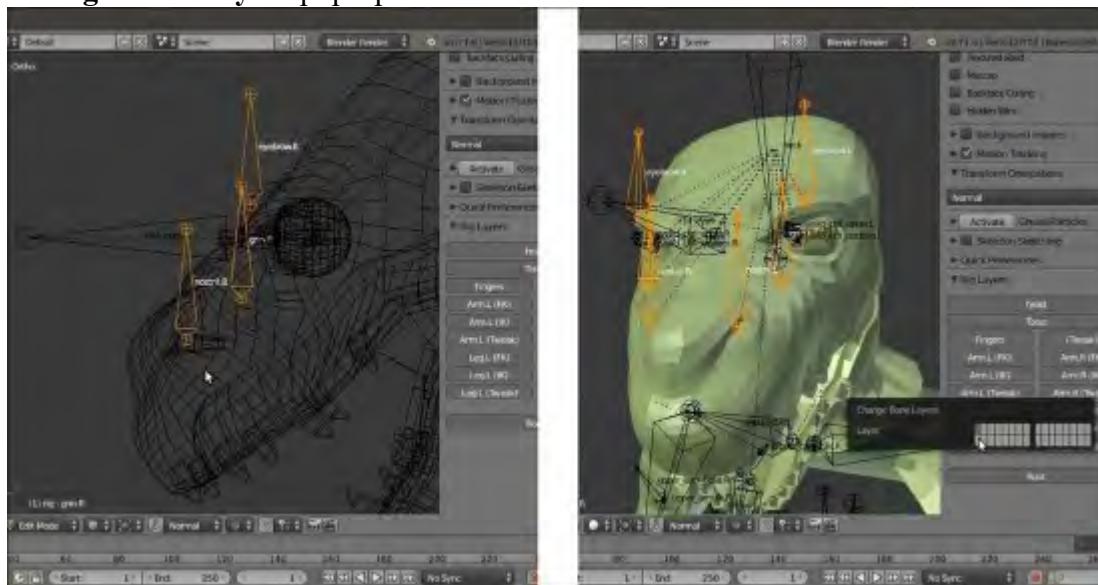
Assigning drivers to the shape keys

In the previous recipe, we created three different types of **shape keys**. Besides the *fixing* shape keys, that have a fixed value (no pun intended), we now need a way to set the amount of influence of the other two types of shape keys, facial expressions, and the muscle movements during the animation. This is accomplished by setting **drivers**, though with different kinds of controls.

Getting ready

Start from the previously saved `Gidiosaurus_shapekeys.blend` file:

1. Go to the **Outliner** and hide the **Armor** object.
2. Select the **Armature** rig, switch to the **Octahedral** bones draw mode, and press *Tab* to enter **Edit Mode**; zoom to the character **head** and add **six** bones located as follows: **two** bones close to both the right and the left **eyebrows**, **two** bones close to both the sides of the **grin snout** area, and **two** bones close to the **nostrils**. Enable the **Names** item in the **Skeleton** subpanel under the **Object Data** window and rename the bones accordingly and with the correct **.L** or **.R** suffix, then be sure to have them located on the first bone layer by pressing the *M* key to call the **Change Bone Layers** pop-up.



The new bones for the shape key drivers

3. Exit **Edit Mode** and select the **Gidiosaurus** mesh.
4. Go to the **Shape Keys** subpanel under the **Object Data** window and expand the list window by left-clicking on the = icon at the bottom and dragging it downward.
5. Now right-click on the value (**0.000**) at the right side of the name of the first shape key (**grin.L**) and from the pop-up panel, select the **Add Driver** item; the value is enhanced, in violet, to show that now it has a **driver** associated.

- Repeat the same for all the shape keys in the list except for the **prop** one, which has a fixed value of **1.000**:



The shape keys list showing they have drivers

- Split the 3D viewport horizontally into two parts, change the upper part into a **Graph Editor** window, or simply switch the screen to the **Animation** layer (in the **two** files provided with the cookbook, there are actually two prepared animation screens, **Animation1** and **Animation2**). Click on the *Editing context being displayed* button in the toolbar of the **Graph Editor** window and change it from **F-Curves** to **Drivers**.

How to do it...

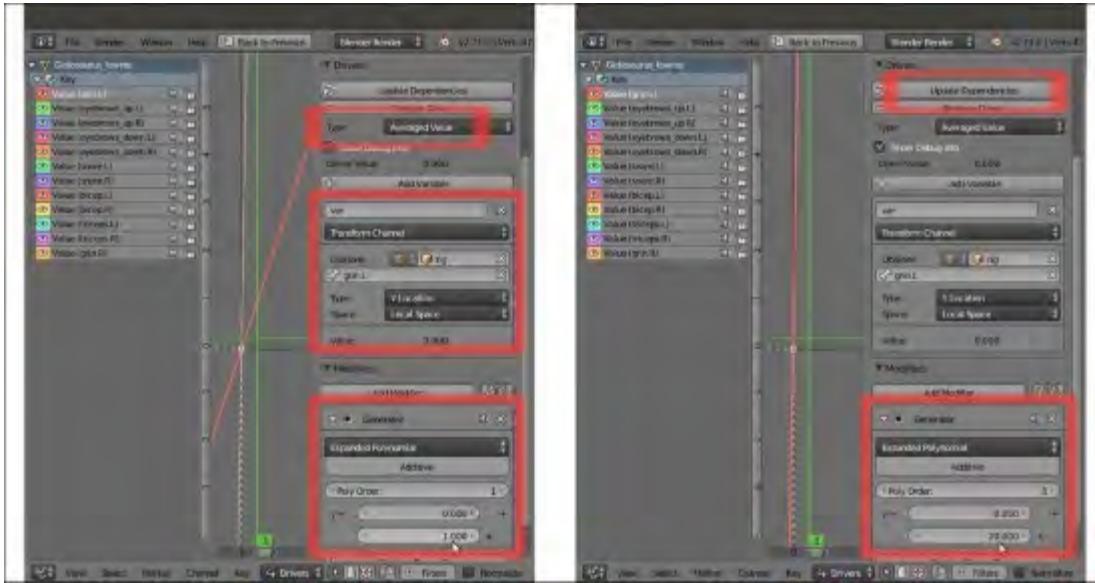
Let's start with the **expressions shape keys**:

- If not already present, press the **N** key to open the **Properties** sidepanel of the **Graph Editor** window, then click on the **Value (grin.L)** top item in the drivers list at the top-left of the screen:



The Graph Editor window, at the top of the screen, with the driver f-curve

2. Go to the **Properties** panel of the **Graph Editor** and, by scrolling down, find the **Drivers** subpanel. In the **Driver Type** slot, switch from the **Scripted Expression** item to the **Averaged Value** one.
3. In the **Ob/Bone** slot, select **rig** and in the under slot (*Name of PoseBone to use as target*), select the **grin.L** bone.
4. Going downward, in the **Variable Type** slot, select the **Y Location** item and in **Space**, select **Local Space**.
5. Click on the **Update Dependencies** button at the top of the **Drivers** subpanel (the **Update Dependencies** function works particularly for **Scripted Expression**; it is quite important to use it to refresh the new setups each time).
6. Go even further down and click on the **Add Modifier** button in the **Modifier** subpanel; from the **Add F-Curve Modifier** pop-up menu, select the **Generator** item.
7. In the **Coefficient for polynomial – x** slot, change the value **1.000** to the value **20.000** (this is to re-map the declivity of the **f-curve** and therefore the speed of the corresponding shape key):



The N Properties Graph Editor sidepanel for the selected driver

8. Now select the **grin.L** bone and in **Pose Mode**, move it upward to see the **grin.L** shape keys being animated on the character's **snout**.
9. Go to the **Shape Keys** subpanel and right-click on the value to the right side of the **grin.L** shape; from the pop-up menu, select the **Copy Driver** item.
10. Select the **grin.R** shape key and right-click on the value to the right; from the pop-up menu, select the **Paste Driver** item.
11. Go to the **Animation** screen and switch the **grin.L** to the **grin.R** bone in the **Ob/Bone** field under the **Drivers** subpanel.
12. Copy and paste the drivers for the **eyebrows_up.L** and **eyebrows_up.R** shape keys, then replace the driver bones names in the **Ob/Bone** field under the **Drivers** subpanel.
13. Go to the **Shape Keys** subpanel under the **Object Data** window and set the **Max** value under the **Range** item to **0.600** for both the **eyebrows_up.L** and **eyebrows_up.R** shape keys; this is to limit the movement of the shape keys to avoid any intersection with the character's **helm**.
14. Copy and paste the drivers for the **eyebrows_down.L** and **eyebrows_down.R** shape keys. This time, leave the same driver bone names and instead change the value of the *Coefficient for polynomial – x* to negative and **-20.000** to *invert* the direction of the **f-curve**.
15. Repeat the procedure for the **snare.L** and **snare.R** shape keys, this time switching the **Variable Type** from **Y Location** to **X Location** and assigning a negative **-20.000** value to the **snare.L** driver and a positive **20.000** value to the **snare.R** one.

At this point, all that is left is to assign automatic drivers for the *shape keys to stretch and grow muscles* we created for the character's **arms**.

16. Click on the **Value (bicep.L)** item in the drivers window at the left top of the **Graph Editor** and then go to the **Properties** panel on the right and then to the **Drivers** subpanel. In the **Driver Type** slot, select the **Averaged Value** item again; in the **Variable Type** slot, switch to **Rotational Difference**.

17. In the **Bone1** slot, select **rig** and in the slot below (*Name of PoseBone to use as target*), select the **DEF-forearm.01.L** bone; in the **Bone2** slot, select **rig** again, and then **DEF-upperarm.02.L**.
18. In the **Graph Editor**, click on a point of the **f-curve** to select it and then press the **L** key to select all the points of the **f-curve**; move them downward, on the **y** axis, by **-1.400** (**G | Y | -1.4 | Enter**).



The triceps Rotational Difference driver

19. Copy and paste the driver to the **bicep.R** shape key, then change the **.L** suffixes of the bones to the **.R** ones.
20. Copy the **bicep.L** driver and paste it to the **triceps.L** shape key; click on the **Add Modifier** button under the **Modifier** subpanel; and from the **Add F-Curve Modifier** pop-up menu, select the **Generator** item.
21. In the *Coefficient for polynomial – y* slot, write the value **2.300** and in the *x* slot, write the value **-1.000** (remember that all these values in the recipe are not universal and are valid just for this **Gidiosaurus** model in this particular setup; the drivers values could change from character to character, so always test them on your model).
22. Copy and paste to the **triceps.R** shape key, and change the suffixes of the bones.
23. Click on the **Update Dependencies** button at the top of the **Drivers** subpanel and save the file.

How it works...

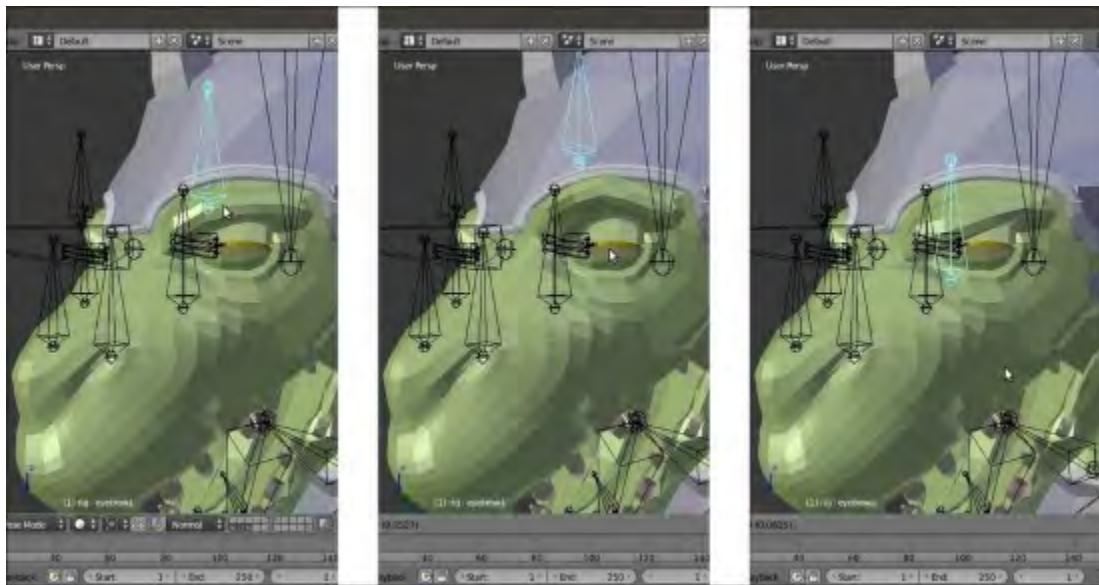
Drivers assigned to bones as controllers for the shape keys are not only an effective way to create a device for animation but also a mandatory technique in the Blender pipeline workflow, where a character is usually linked into the scene from a different file and the rig gets **proxified** (we'll see how to do this in the next chapter). The only possible way to have access to the shape keys in a linked character is through the **drivers** and the **rig**.

As you probably already know, shape keys are often used not only for **facial expressions** but also to mimic the stretching and the growing of the body's **muscles** according to the movement of a character's **limbs**. In this case, their influence is automatically driven by the rotation of the respective bones through the **Rotational Difference** drivers that, as the name itself says, base their influence on the difference of rotation between two bones; more precisely, on the **angle** between them.

The **Generator** modifier we added is a multiplier we used to *virtually* modify the slope inclination of the **f-curves** of the drivers. The default inclination of the **f-curve** wasn't enough to fully map the curve itself to a driver bone movement of (almost) just one or two Blender Units (it was too slow, resulting in a required driver movement of several units to have an appreciable effect), so we increased the declivity by a factor of **20.000** to have a faster correspondence.

However, the same modifier was also used to reverse the direction of the **f-curve**, by using a negative value of **-20.000**, for example to drive the downward movement of the **eyebrows**, or to change the location of the curve along the y axis so as to tweak the timing of the driver influence, like in the **triceps** shape keys.

Therefore, by copying and pasting a driver and giving an opposite declivity at the slope of the copied one, it is possible to drive two opposite shape keys through the same bone, as for the **eyebrows** shape keys:



The same bone moving in two opposite directions to drive two opposite shape keys

There's more...

To add shape keys and the respective drivers to the **Gidiosaurus** model, we used the **Gidiosaurus_skinning_rigify.blend** file, with the **rig** created by the **Rigify** addon. The control bones of a **Rigify** rig have pre-made **Custom Shapes** to make their identification and selection easier and are usually located in the last scene layer.

So, for the last step, I just modeled a new simple custom shape, a small **Circle** mesh with **16** vertices. I named it **Widget_generic4** and I assigned it to all the *driver* bones:



The driver bones with the new Custom Shape

See also

- <http://www.blender.org/manual/animation/basics/drivers.html>

Setting movement limit constraints

Often, it is very useful to put movement limitations on the bones of a rig, for several reasons—usually, to make them easier to work with, but also to establish a maximum range for the rotation of the **limbs** or other parts like in the **mandible** or the **eyelids**.

Two types of limits for the bones are: by the **Transform** locks, and by **bone constraints**.

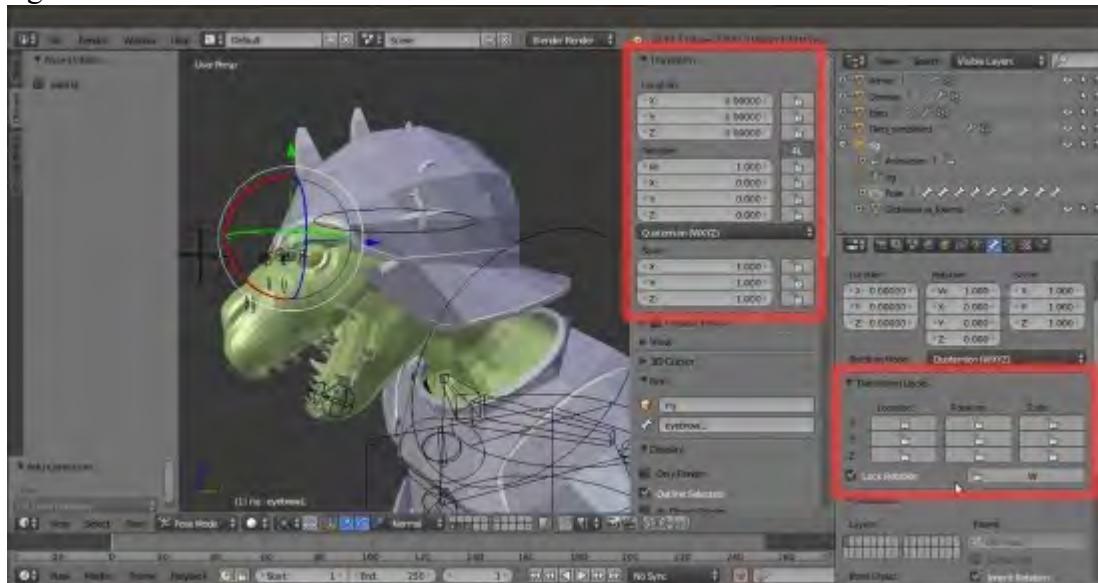
Getting ready

Load the `Gidiosaurus_shapekeys.blend` file, select the **Armature**, and go in **Pose Mode**.

How to do it...

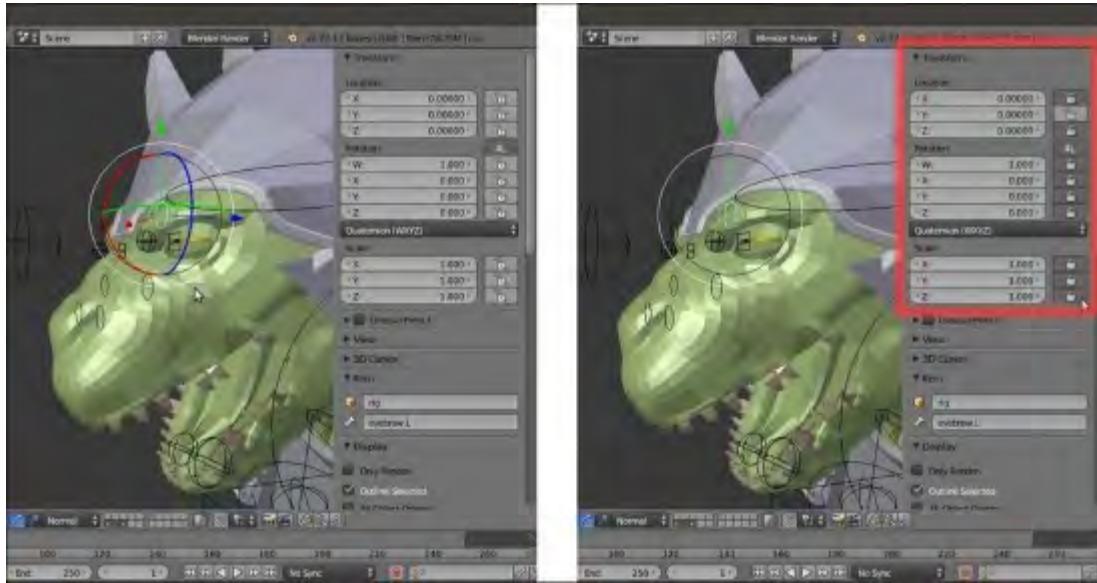
Let's start with the **Transform** locks:

1. Select the **eyebrow.L** bone and if not already present, press the **N** key to call the 3D viewport **Properties** sidepanel. Go to the **Transform** subpanel, which is the first entry at the top, or also to the **Transform Locks** subpanel under the **Bone** window in the main **Properties** panel to the right of the screen:



The **Transform** subpanel in the **N Properties** sidepanel and the corresponding **Transform Locks** subpanel under the main **Properties** panel

2. Click on the lock icon to the right side of the properties; for this bone (which, if you recall, is the driver control object for the left up and down **eyebrow** shape keys), we want to lock all the possible transformations *except* for the movement on its **y** axis, so the **Location Y** lock button is the only one that should remain untouched:



Setting the axis Transform locks for the Location, Rotation, and Scale

If you now try to move the **eyebrow.L** bone, you will notice its movement is constrained only to its local y axis; the movement is directed by the **Roll** orientation of the bone in **Edit Mode** (and not by the **Normal** item enabled in the **Transform Orientation** button on the viewport toolbar); enable the **Axes** item in the **Display** subpanel under the **Object Data** window to see this.

Having locked the other two axes, it's no longer necessary to use the widget arrow to move the bone on its local y axis but it's enough to simply press the **G** key and then move the mouse instead.

And now, let's see limits by **constraints**.

3. Go to the **Bone Constraints** window and assign a **Limit Location** constraint to the **eyebrow.L** bone.
4. Check the **Minimum X** and **Maximum X**, **Minimum Y**, and **Maximum Y**, and **Minimum Z** and **Maximum Z** items. Leave the values for the *x* and *z* axes as they are, change **Minimum Y** to negative **-0.050**, and change **Maximum Y** to positive **0.050** (again, remember that these values are valid just for this file).
5. In the **Convert** slot, change the **Owner Space** item to **Local Space**:



The assigned Limit Location constraint subpanel under the main Properties panel

In [Chapter 1, Modeling the Character's Base Mesh](#), we enabled the **Copy Attributes Menu** add-on in **User Preferences** and then we saved the **User Settings**, so I'm taking for granted that you have the script still enabled.

Therefore, we do the following:

6. Select the **eyebrow.R** bone and then *Shift*-select the **eyebrow.L** bone. Press ***Ctrl + C*** and from the **Copy Attributes** pop-up menu, select the **Copy Bone Constraints** item.
7. Select the **grin.L** and **grin.R** bones and then *Shift*-select the **eyebrow.L** bone. Once again, press ***Ctrl + C | Copy Bone Constraints***, and in the two copied constraints, set the **Minimum Y** value to **0.000**.
8. Select the **nostril.L** and **.R** bones and *Shift*-select the **grin.L** bone, then press ***Ctrl + C | Copy Bone Constraints***. This time, set both the **Y** values to **0.000** and **Minimum X** for the **nostril.L** bone to negative **-0.050** and the **Maximum X** for the **nostril.R** bone to positive **0.050**.
9. Save the file.

Several other movement constraints have been added to different bones in the rig, for example the **jaw** bone, or the **eyelid** controllers, but especially to the **eye** bones, to limit the range of possible rotations. To have a look at the various settings, just open the **Gidiosaurus_limits.blend** file.

See also

- <http://www.blender.org/manual/animation/techs/object/constraint.html>

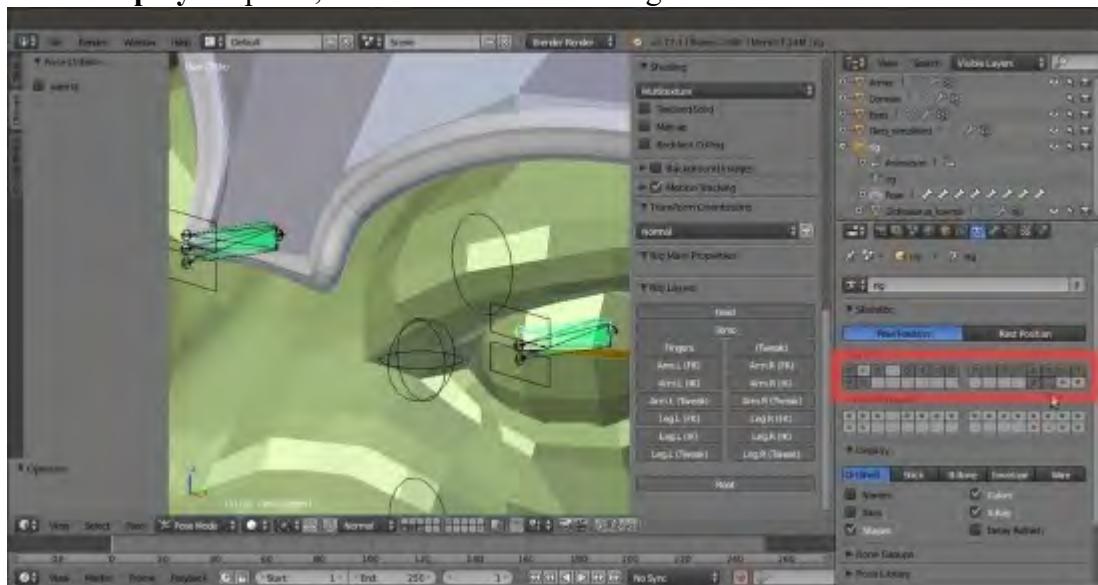
Transferring the eyeball rotation to the eyelids

This is a really simple trick that can add a lot of life to the facial expressions of an animated model, making the **eyelids** follow some of the movement of the **eyeballs**.

Getting ready

Following on from the previous recipes, open the `Gidiosaurus_limits.blend` file:

1. If not already selected, select the **Armature** and enter **Pose Mode**.
2. In the **Object Data** window, go to the **Skeleton** subpanel and enable the **30th** bone layer, to show the deforming bones.
3. In the **Display** subpanel, switch the bones' drawing mode from **Wire** to **Octahedral**:



The Skeleton subpanel with the bone layers

How to do it...

Now zoom to the character's **head** and continue with the following steps:

1. Select the **eyelid_upper.L** bone and go to the **Bone Constraints** window; assign a **Copy Rotation** constraint.
2. In the **Target** field, select the **rig** item, and in the **Bone** field, select the **eye.L** bone item. Set **Space = Pose Space to Pose Space**.
3. Set the **Influence** slider value to **0.300**.
4. Select the **eyelid_bottom.L** bone and **Shift-select** the **eyelid_upper.L** bone, then press **Ctrl + C** | **Copy Bone Constraint**.
5. Select the **eyelid_upper.R** bone and repeat the procedure but with **eye.R** as the target bone; copy the constraint to the **eyelid_bottom.R** bone:



The eyelids slightly following the eye movements

6. Save the file.

Detailing the Armor by using the Curve from Mesh tool

In [Chapter 3, Polygonal Modeling of the Character's Accessories](#), in the *Using the Mesh to Curve technique to add details* recipe, you already saw how to use this technique as a modeling tool. In this recipe, we'll use the same technique but in the opposite direction—to add **rivets** around the perimeter of the borders of the different **Armor** parts.

Getting ready

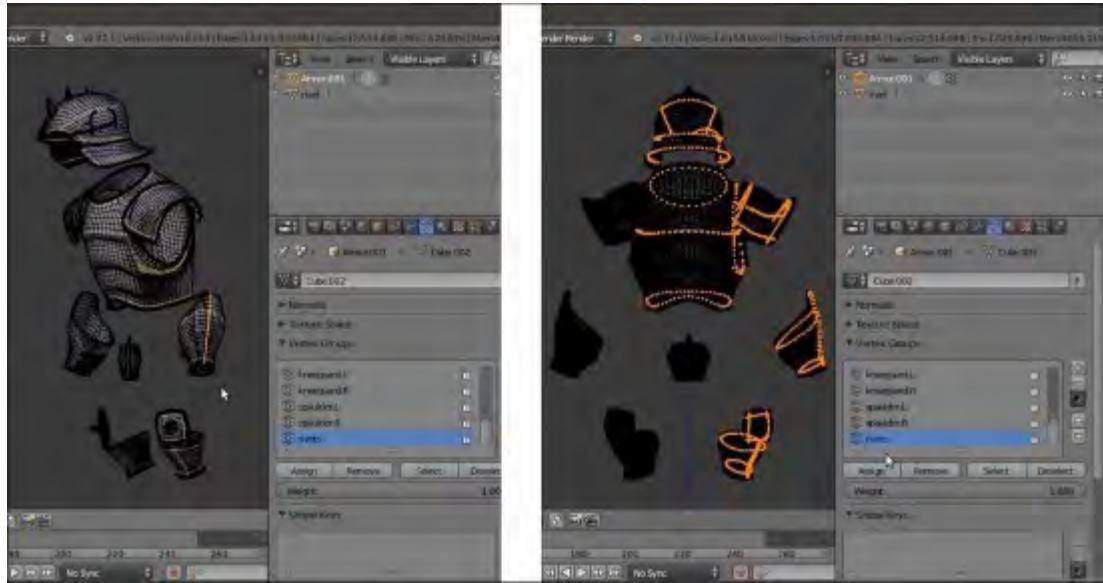
Re-open the `Gidiosaurus_limits.blend` file; the first thing to do is to model a very lowpoly **rivet** object to be duplicated on the **Armor** surface:

1. Switch to an empty scene layer, press **Shift + C** to place the **3D Cursor** at the center of the grid, and add a **Cube** primitive mesh. Enter **Edit Mode** and delete the bottom face, then scale the remaining faces by a value of **0.100** twice, then one last time by **0.500**. Move the top face downward to flatten the overall shape a bit and scale the same face by **0.700**.
2. Press **A** to select all the vertices and **W** to choose the **Subdivide Smooth** item from the **Specials** pop-up menu, then delete the middle horizontal edgeloop.
3. Put the pivot on the **3D Cursor** and while still in **Edit Mode**, rotate all the vertices by **90°** on the **x** axis.
4. Select the bottom edgeloop and press **Shift + S | Cursor to Selected**. Exit **Edit Mode** and click on the **Set Origin** button under the **Tool** tab to select the **Origin to 3D Cursor** item.
5. Click on the **Smooth** button under the **Shading** item and in the **Outliner**, rename the **rivet** object. Once again, place the **3D Cursor** at the center of the grid and the **rivet** at the **Cursor** location; press **Ctrl + A** to apply the **Rotation & Scale** option.
6. Enable the scene layer with the **Armor** on it, and in the **Outliner**, hide the **rig**.

How to do it...

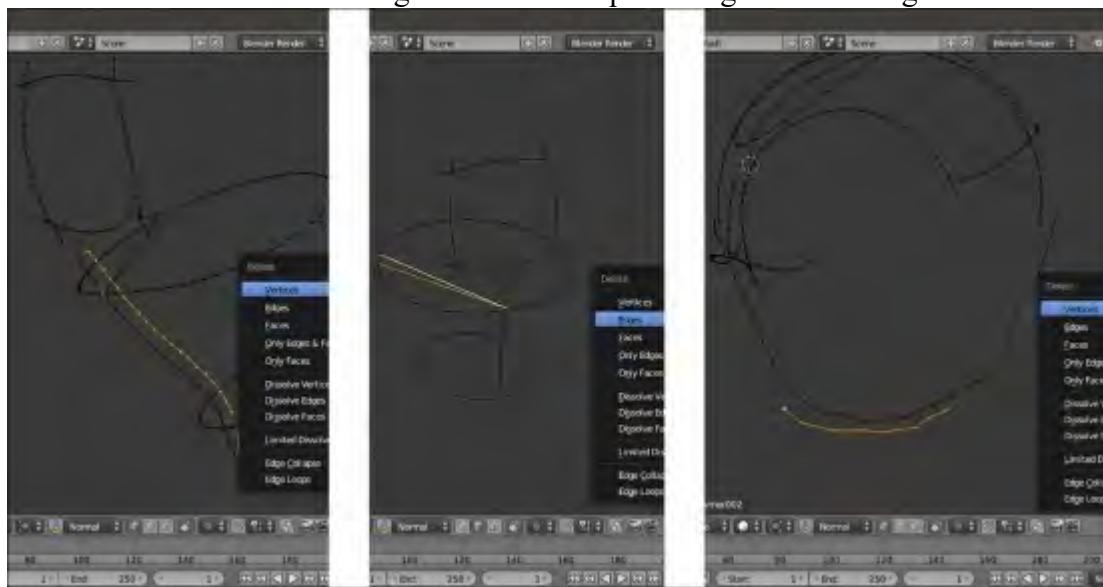
Now, let's create the *guides* to duplicate the rivets on:

1. Select the **Armor** object and press **Shift + D** to duplicate it, then place the duplicate **Armor.001** object on the scene layer of the **rivet**. Go to the **Shape Keys** sidepanel under the **Object Data** window and delete the **Armor_fix** first and then the **Basis** shape keys.
2. Go to the **Object Modifiers** window, remove the **Armature** modifier, and apply the **Subdivision Surface** modifier with a **Subdivision** level of **2**.
3. Enter **Edit Mode** and start to select the edgeloops on the different **Armor** parts in areas where you want to add the **rivet** rows (**Alt + right-click** for the first one, then **Alt + Shift + right-click**). As usual, it's enough to work only on one half of the mesh:



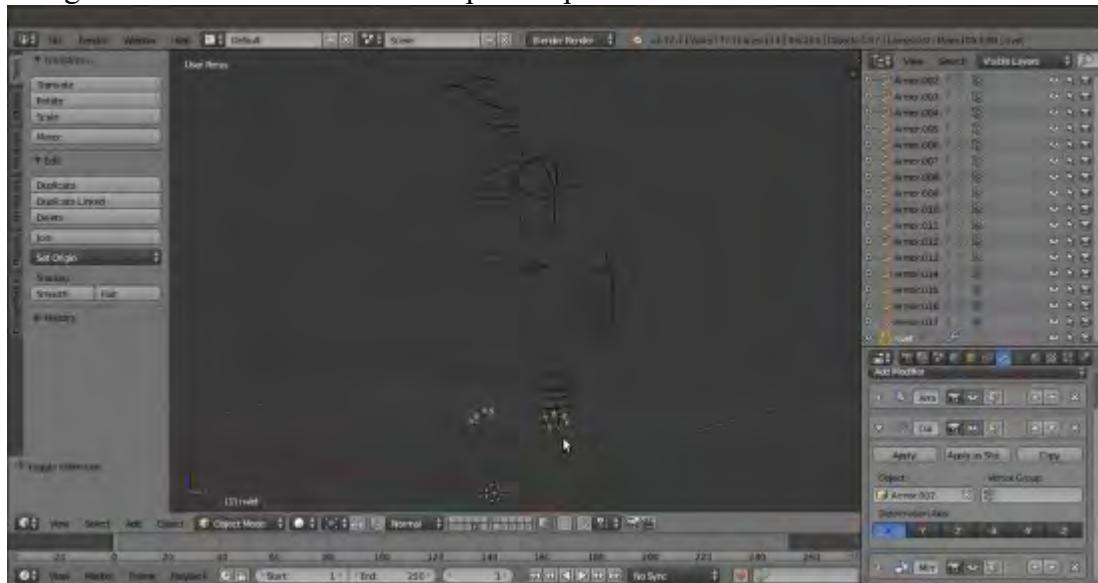
The Armor mesh in Edit Mode with the selected edge-loops

4. Press *Shift + D* and soon after, click the right mouse button to duplicate the selected edgeloops without moving them, then press the *P* key to separate them from the **Armor.001** object (in the **Separate** pop-up menu, choose the **Selection** item).
5. Exit **Edit Mode** and delete the **Armor.001** object, or if you don't have problems with big file sizes, move it to a different scene layer to keep it for future refinements. In this case, you can save the edge-loops selection as a vertex group named **rivets**.
6. Select the **Armor.002** object (the duplicated and separated edgeloops) and enter **Edit Mode**; make the necessary adjustments to the edgeloops by deleting the unnecessary vertices, for example the backsides of the plates, and disconnect the welded edgeloops by deleting the common vertices or connecting them where required edges are missing:



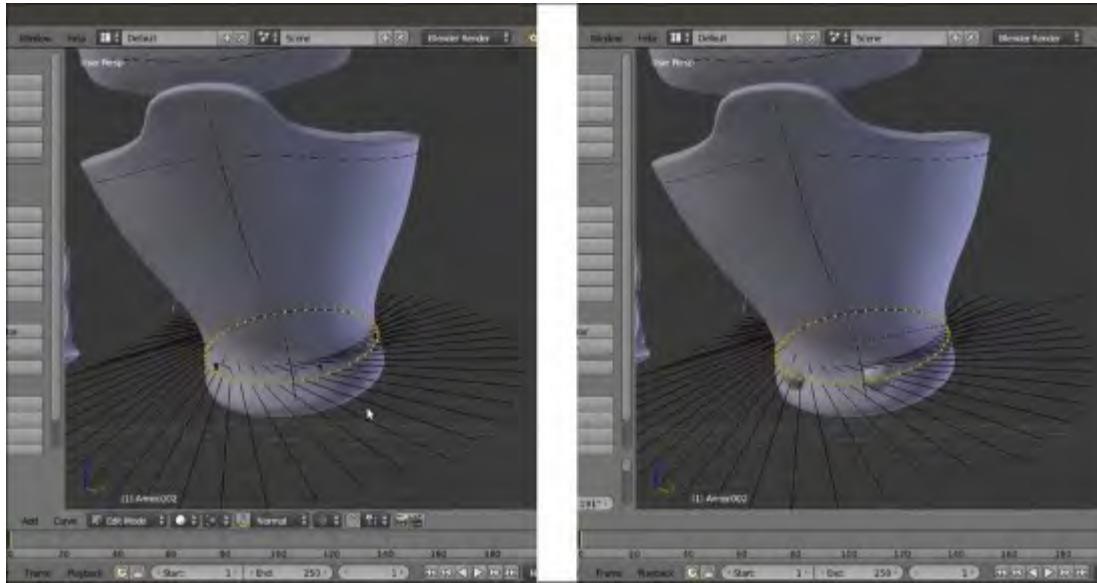
Cleaning the edge-loops of the duplicated Armor.002 mesh

7. Press **A** to select all the vertices and then go to the **Tools** tab under the **Tool Shelf**. Go to the **LoopTools** subpanel and press the **Space** button to evenly space the vertices along the edgeloops.
8. Exit **Edit Mode** and press **Alt + C**; in the **Convert to** pop-up menu, select the first item, **Curve from Mesh/Text**. The mesh edgeloops actually get converted into a **Curve** object, as you can see in the **Object Data** window under the main **Properties** panel to the right of the UI. Click on the **Fill** slot to select the **Full** item.
9. Now the tedious part (but not difficult, just a little tedious); in **Edit Mode** again, put the mouse on one of the points and by pressing the **L** key, select each separate part of the **Curve**, then press **P** to separate the whole selected part. This way, you are going to obtain **16** separated **Curve** objects.
10. Select the **rivet** object and go to the **Object Modifiers** window; assign an **Array** modifier with **Fit Type = Fit Length, Length = 0.50**, and **Relative Offset X = 3.000**. Collapse the panel.
11. Assign a **Curve** modifier, then in the **Object** field select the **Armor.002** curve. Leave the panel expanded.
12. Assign a **Mirror** modifier and collapse the panel:



The rivet object instanced on the mirrored curve object

13. In the viewport area, zoom to each curve to check for the correct tilting of the points; if necessary, select the curve, enter **Edit Mode**, select all the points, press **Ctrl + T**, and move the mouse to rotate the tilting of the curve's points until the instanced **rivets** are correctly rotated/aligned with the surface of the main **Armor** mesh:



Tilting the curve's points

If necessary, you can also select individual points of the curve to tweak the orientation of only a part of the instanced rivets, even of single rivets at once; this has been done for part of the **helm** and for the **spaulders**, especially.

14. In the **Outliner**, re-select the **rivet** and press **Shift + D** to duplicate it, then in the **Object Modifiers** window, under the **Curve** modifier panel, select the **Armor.003** item in the **Object** field.
15. Once again, zoom to the curve and if necessary, fix the curve tilting and also adjust the **Length** value of the **Array** modifier (for the **Armor.003** curve it has been raised to **0.59**) and the **Relative Offset** value. By selecting all the points and pressing **W**, you can also select the **Switch Direction** item in the **Specials** menu.



The rivets on the helm object

- Duplicate the rivet and repeat the procedure changing the curve name in the modifier for each curve object and so on. At the end, you should have **16** copies of the rivet as well.

At this point, if required, we can still make some modification to the rivet mesh; in my case, I just subdivided it a bit more, then deleted some useless edgeloop, made it rounder, and extruded the open side a bit more.

Now that the rivet is ready, select all the rivet copies (so that the *modified one* is the active object, that is *the last selected*) and press **Ctrl + L | Object Data** to share the modifications between them.

Leaving everything selected, press **U | Object & Data** to make them single users again (this is necessary for the next step with the modifiers).

- When you are done, select all the rivets *one at a time* in the **Outliner** and apply all the **Array** and the **Curve** modifiers.
- Join all the rivets into a single object (select all and press **Ctrl + J**) and in **Edit Mode**, delete the unnecessary or overlapping ones, keeping only the rivets that really add to the **Armor** look.

Then, apply all the **Mirror** modifiers:



The completed rivets

- Select the **Armor** object and then *Shift-select* the rivets object, press **Ctrl + Tab** to go in **Weight Paint** mode and click on the **Transfer Weights** button under the **Tools** tab.
- Exit **Weight Paint** mode and assign an **Armature** modifier to the **rivets** object, select **rig** in the **Object** field.
- Save the file as **Gidiosaurus_final_detailing.blend**.

There's more...

At this point, the **Gidiosaurus** model is ready to be animated, but some minor adjustments are still missing and can be added.

I won't go into the details about these additions, they are all processes you have already seen in the previous chapters and recipes, so this is simply a showcase:



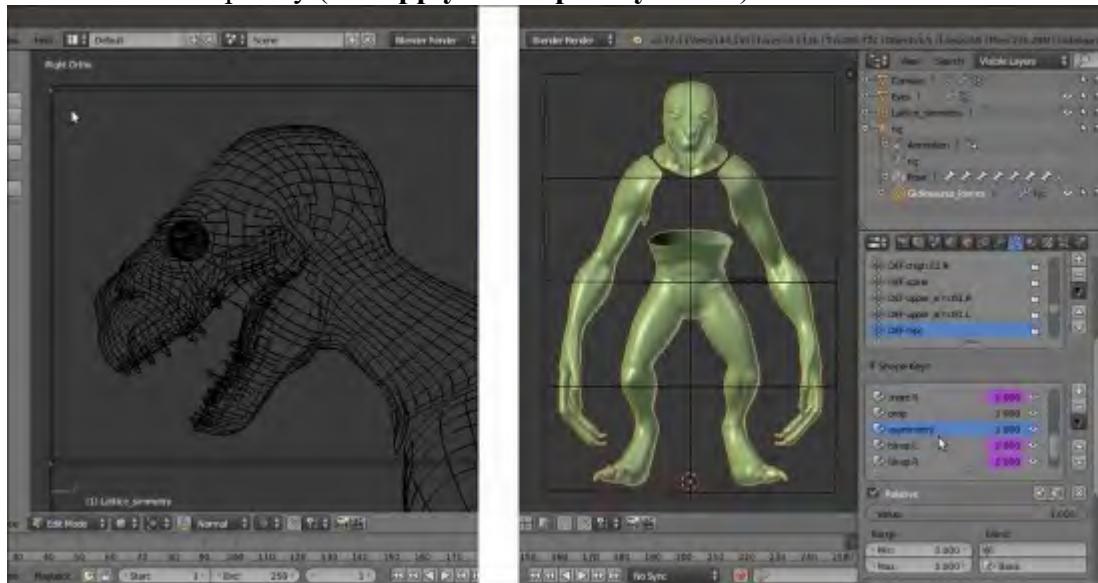
The modeled tiers and the rivets

1. The tier attachments on the **Armor's vambraces** and on the **greaves** have been refined by adding smaller **rivets**, and new tiers have been added to the sides of the **Armor chest plate**. Also, the opening seams in the **Armor** parts have been modeled under each tier location.
2. The **Armor** decorations have been separated as a new object (the **Armor_decorations** item in the **Outliner**) and simplified by deleting as many edgeloops as possible without altering their basic shape:



The simplified decorations in Edit Mode

3. The **Armor**'s shape also has been tweaked even further through the **Armor_fix** shape key to adjust some overlaps that were occurring during the movements of the **spaulders** and in the **stomach** area too. The same shape key has been repeated also on the **decorations** and on the **rivets** objects for the areas of interest.
4. A bit of asymmetry has been introduced in the **Gidiosaurus** mesh by assigning a **Lattice** modifier to the character, and slightly modifying the shape on the left side, then applying the modifier as a shape key (the **Apply as Shape Key** button):



The asymmetry lattice

- Finally, after some test renders, I realized that the **teeth** and the inside of the **mouth** of the **Gidiosaurus** still needed refinements, so I made some more adjustments to the **prop** shape key by making the **teeth** bigger and bolder, and the **inner mouth** more organic-looking and smooth:



The modified teeth and inner mouth

Be aware that almost in every modeled object there is still room for improvement, and that's okay, it's not a sign of a bad job! This sort of improvement is done all the time and is simply part of the working experience.

See also

- <http://www.blender.org/manual/modeling/curves/index.html>

Chapter 9. Animating the Character

In this chapter, we will cover the following recipes:

- Linking the character and making a proxy
- Creating a simple walk cycle for the character by assigning keys to the bones
- Tweaking the actions in Graph Editor
- Using the Non Linear Action Editor to mix different actions

Introduction

There are literally a *plethora* of tutorials and manuals about animation principles in general, and in Blender in particular, on the Web and in bookstores, so this one is going to be just a very *easy* chapter, mainly about the **technical aspects** of creating a simple animation with the rigged **Gidiosaurus** character, following the most usual pipeline commonly used in Blender (at least for the **open movies**).

Linking the character and making a proxy

The habit of linking assets from library files is the most useful and used, I would say, not only in a Blender based workflow, but also in the *industry*. A linked asset, in our case a creature character, can be placed and animated even if not already completed in all its parts, thus it allows a team to work almost *at the same time* on the different aspects. In our case, the **Gidiosaurus** is still missing **texturing** and **shaders**, but can already be placed *on stage* and animated anyway.

To link an asset in Blender and keep the possibility of animating it through a rig, we must make a proxy of the **rig** itself. A **proxy object** overrides the animation controls of a linked object in a non-destructive way, so that an animator can animate it locally to the .blend file the rigged character has been linked to. This way, the linked character object retains all its original information and is *only locally* altered by the proxy object scene.

Getting ready

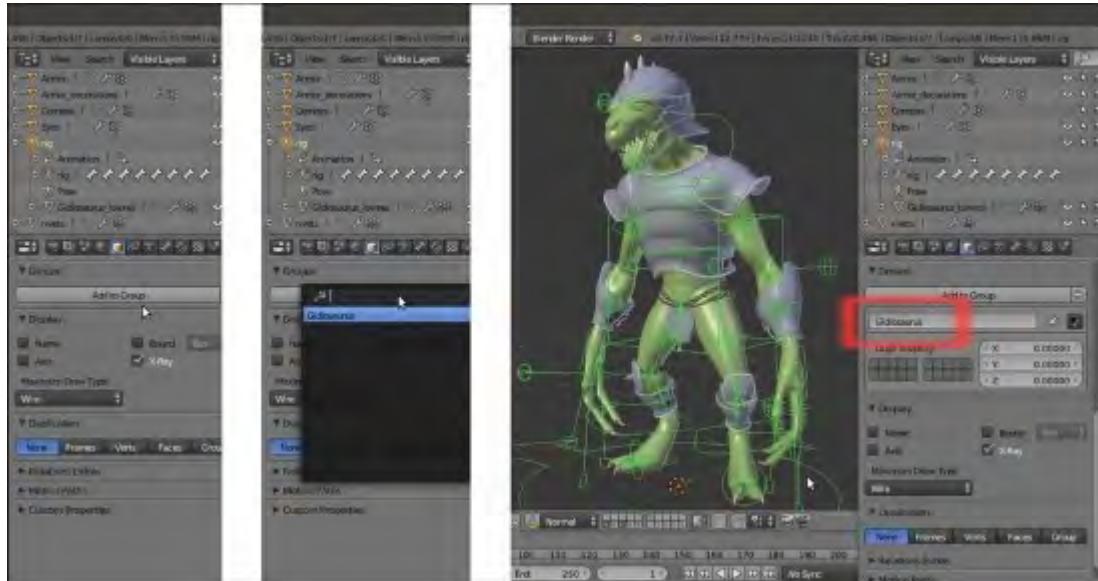
As the first thing, we must prepare the **library**, so open the **Gidiosaurus_final_detailing.blend** file:

1. Go to the **Outliner** and select the **Gidiosaurus_lowres** mesh, then also *Shift-select* the **Armor**, the **Armor_decorations**, the **rivets**, the **Eyes**, and the **Corneas** objects.
2. Press *Ctrl + G*, and all the selected objects are outlined in green to show that now they belong to a **group**, in this case, to the same group we created just now.
3. Go to the **Object** window and in the **Groups** subpanel, change the generic default **Group** name to **Gidiosaurus**.



Creating a Group and assigning all the selected objects to it

4. Go to the **Outliner** and click on the eye icon to the side of the **rig** item to make it visible again, and then click on the **rig** item itself to select it.
5. Press **Ctrl + Tab** to go out of **Pose Mode** and go to the **Groups** subpanel under the **Object** window again. Click on the **Add to Group** button and in the pop-up menu, select the **Gidiosaurus** item (in this case, the only group already created). The **rig** is outlined in green as well:



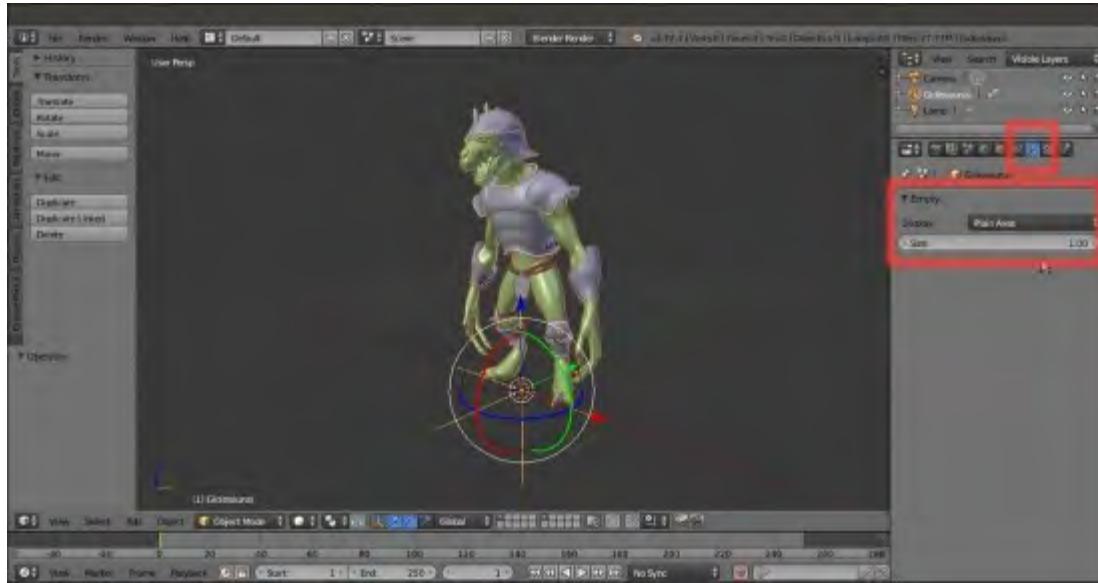
The rig assigned to the group as well

6. Click again on the *Restrict view-port visibility* button (the one with the eye icon) to the side of the **rig** item to hide it and save the file as **Gidiosaurus_library.blend**.

How to do it...

1. Click on the **File** item in the main menu bar, select the **New** item, and confirm by clicking on the **Reload Start-Up File** pop-up (or just press **Ctrl + N**).
2. Select the default **Cube** and delete it, then go to **File | Link** (or press **Ctrl + Alt + O**). Browse and click on the **Gidiosaurus_library.blend** file, then click on the **Group** folder item, and finally click on the **Gidiosaurus** item. Click on the **Link from Library** button to the top right of the screen.

A new object has appeared at the **3D Cursor** location (that should be placed at the center of the scene), and what we have got at this point is the **linked Gidiosaurus group**; this means that the character and any other object inside the **Gidiosaurus** group in the library file are now linked and *instanced* on an **Empty** that is named **Gidiosaurus** as well:



The Gidiosaurus group linked and instanced on the Empty

Remember that in the library file, inside the **Gidiosaurus** group we put also the **rig**, which for the moment is not visible in the linked group because it is *hidden in the library file*.

3. Press *Ctrl + Alt + P*, and a new pop-up appears where we can select the item we want to *proxify* (although all the objects inside the group appear in the list, at the moment only an **Armature** can be proxified). Click on the **rig** item:



The proxified rig

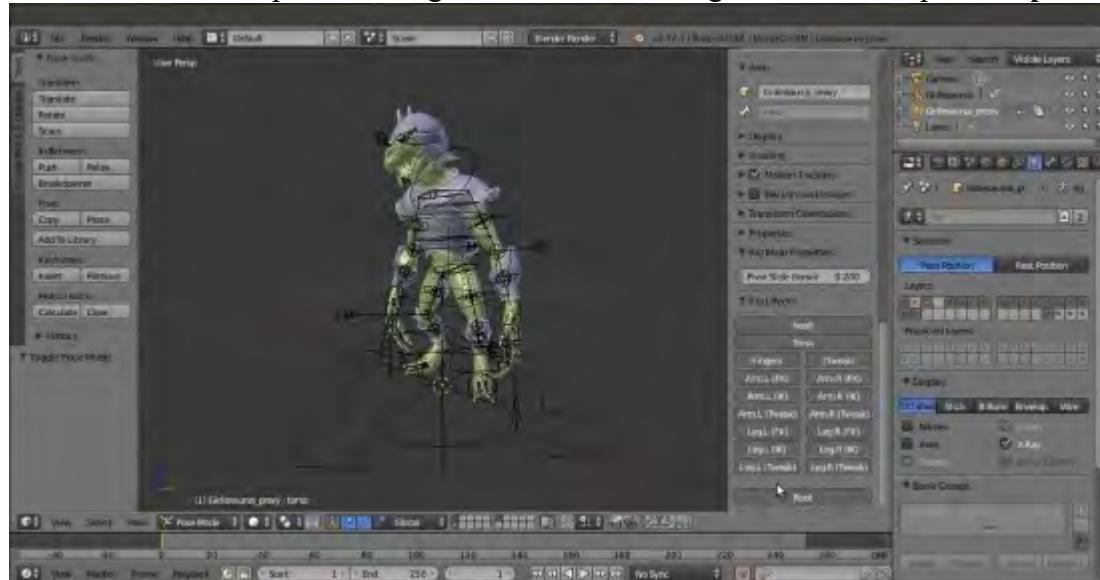
The **rig** appears as a separate object in the **Outliner**, identified by the name **Gidiosaurus_proxy**; at this point, it is possible to only select the **rig** (which is still in **Object Mode**) and move it to a different layer.

4. Select the **Gidiosaurus_proxy** object and move it to the **11th** scene layer (use the *M* key). *Shift*-click to enable the layer and then go to the **Display** subpanel, under the **Object Data** window, to enable the **X-Ray** item.
5. Press *Ctrl + Tab* to go into **Pose Mode** and the *N* key to call the viewport **Properties** sidepanel.
6. Save the file as **Gidiosaurus_proxy.blend**.

At this point, looking at the viewport **Properties** sidepanel, we will see the **Rig Layers** interface usually created by the **Rigify** addon, *but* if we save the file and reopen it, the interface is gone.

This is because, at least for the moment, the Python script that draws the rig interface doesn't get automatically linked with the rig, so it's something we must do by hand. This is not a big issue, and by the way, the procedure is incredibly simple:

7. Click again on **File | Link** in the main header menu (or press *Ctrl + Alt + O*).
8. Browse to the **Gidiosaurus_library.blend** file, click on it, and then click on the **Text** item. Click on the **rig_ui.py** item (the Python script for the interface) and then on the **Link from Library** button.
9. Save the file and reopen it; the rig interface is visible again on the viewport **Properties** sidebar:



The rig interface at the bottom of the Properties sidebar

See also

- http://wiki.blender.org/index.php/Template:Release_Notes/2.43/Animation/Proxy_Objects
- http://www.blender.org/manual/data_system/linked_libraries.html?highlight=proxy#proxy-objects

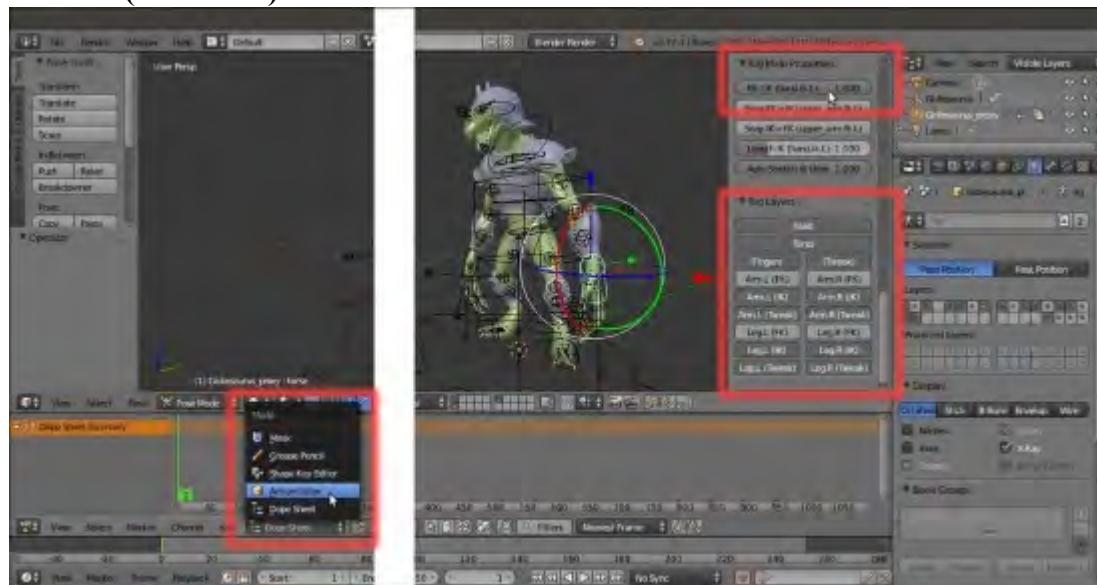
Creating a simple walk cycle for the character by assigning keys to the bones

We are now going to create a simple **walk cycle** for the **Gidiosaurus** character by assigning **position** and **rotation** (and in some cases, also **scaling**) keys to the **control bones** of the **rig**.

Getting ready

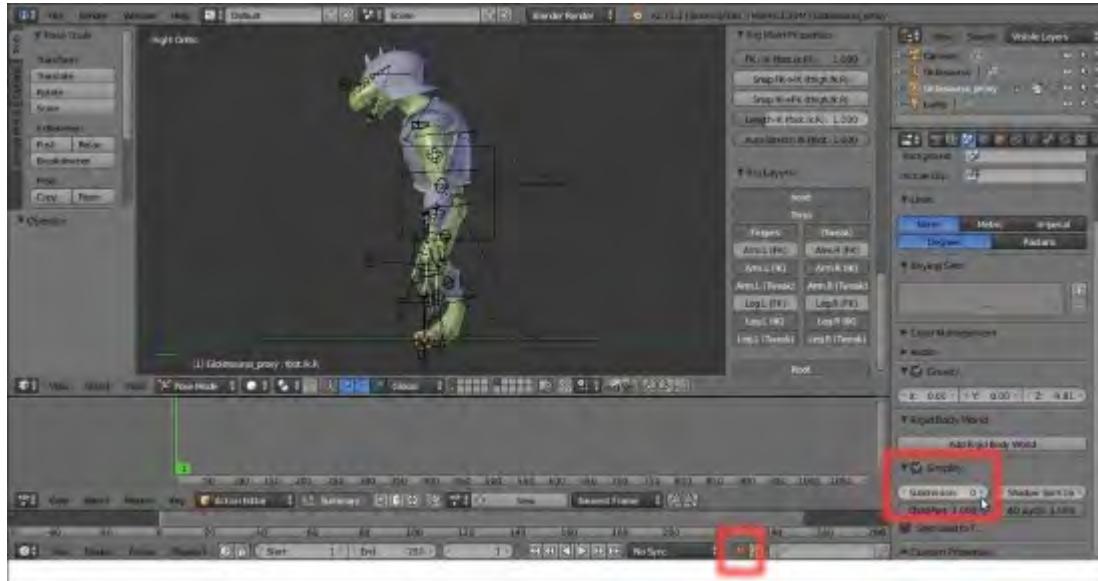
In Blender, there is already a **preset screen** layout named **Animation** that you can switch to and start animating. By the way, I usually prefer to set up my screen layout for the required task, and animating is no exception, so let's first prepare the scene and the screen for the job:

1. Open the **Gidiosaurus_proxy.blend** file.
2. If necessary, enable the **3D manipulator widget** in the toolbar of the 3D view (press **Ctrl + Spacebar**), click on the **Translate** icon button, and set **Transform Orientation** to (just for the moment) **Global**.
3. Split the 3D view horizontally into two windows and change the bottom one into a **Dope Sheet** window. Click on the *Editing context being displayed* button on its toolbar to switch from **Dope Sheet** to the **Action Editor** context **Mode**.
4. Go to the **Properties** sidepanel of the 3D viewport (use the **N** key to make it appear if necessary) and under the **Rig Layers** subpanel, disable the **Arm.L (FK)**, **Arm.R (FK)**, **Leg.L (FK)**, and **Leg.R (FK)** buttons.
5. Select the **Gidiosaurus_proxy** rig, making sure you're in **Pose Mode**, and select the **hand.ik.L** control bone. Go to the **Rig Main Properties** subpanel under the **Properties** panel and set the **FK / IK (hand.ik.L)** slider to **1.000**:



Switching from the Graph Editor to the Action Editor and setting the Inverse Kinematics in the Rig Layers subpanel

6. Repeat for the **hand.ik.R** bone and for the **foot.ik.L** and **foot.ik.R** control bones as well.
7. Go to the **Scene** window, enable the **Simplify** subpanel, and set the **Subdivision** level to **0** (or, if you have a more powerful machine than my laptop, also to **1**).
8. Go into the **Side** view and press the **5** key on the numpad to go into the **Ortho** view.
9. Click on the **red button icon** (*Automatic keyframe insertion for Objects and Bones*) in the **Timeline** toolbar.



The red button icon and the Subdivision Surface modifier subpanel

10. Save the file as **Gidiosaurus_walkcycle.blend**.

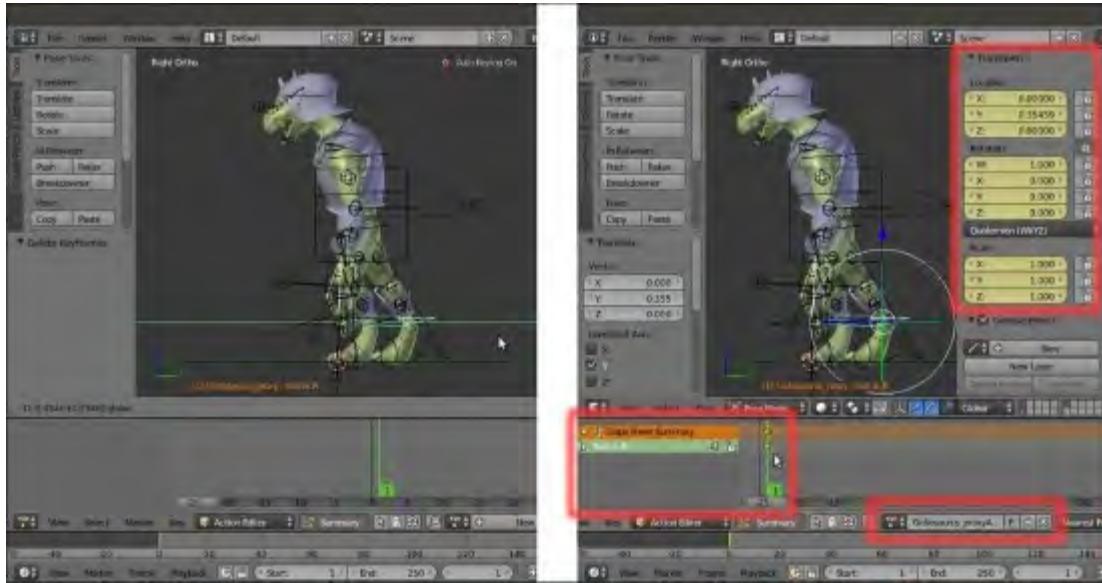
How to do it...

To create a walk cycle, it's important to first establish the start and the end poses of the walk, so let's pose our character for his first step:

1. Be sure to be in the first frame (which in Blender is frame **1** and not **0**), both by clicking on the *Jump to first/last frame in frame range* left button on the **Timeline** toolbar or by pressing the **Shift + Left Arrow** keys.
2. Select the **foot_ik.R** control bone and, by using the widget, move it backward on the global y axis to around **0.350**.

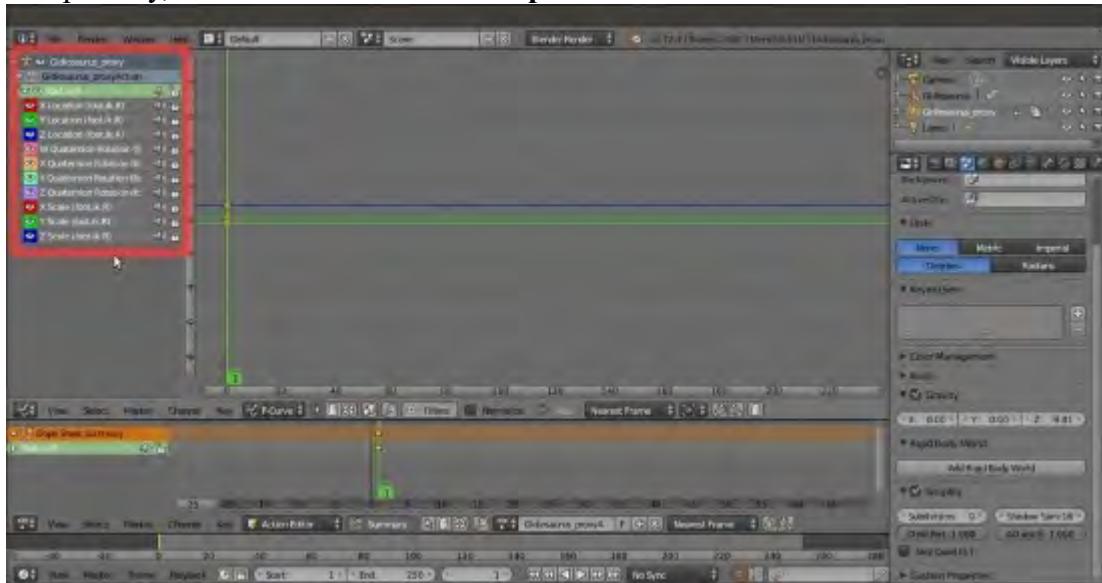
As you release the mouse button, an **Action** datablock, automatically named **Gidiosaurus_proxyAction**, is created and a keyframe for the **foot_ik.R** bone is automatically added in the first frame in the **Action Editor** window. We can also see the value for the movement on the y axis in the **Transform** subpanel.

Note that all the transformation value slots turned yellow; this is to show that at the current frame, an animation keyframe exists for all those values:



Setting the first key at frame 1

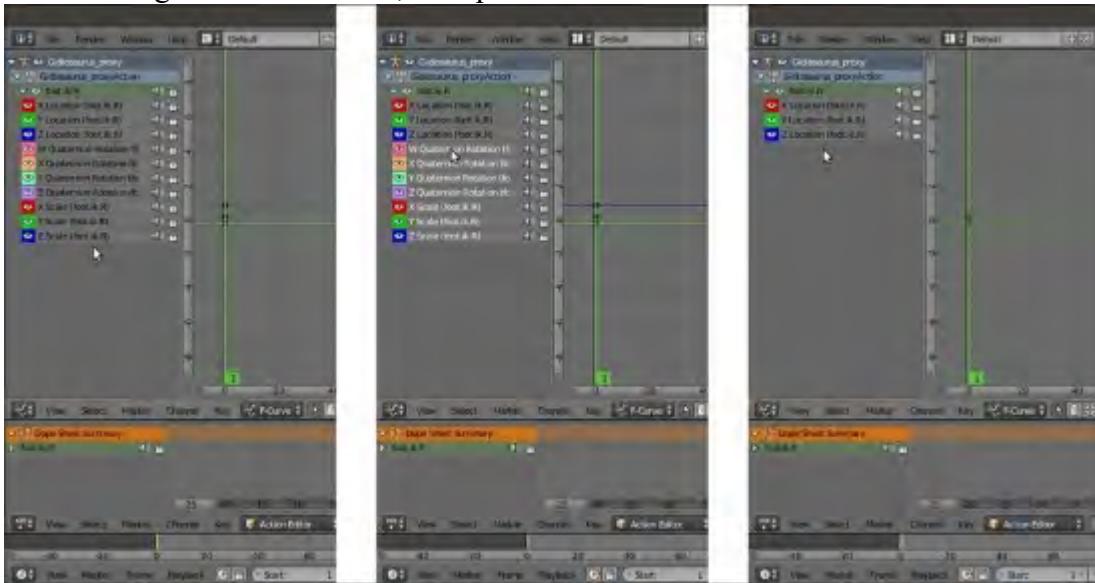
3. Temporarily, switch 3D View to the Graph Editor window:



The Graph Editor window

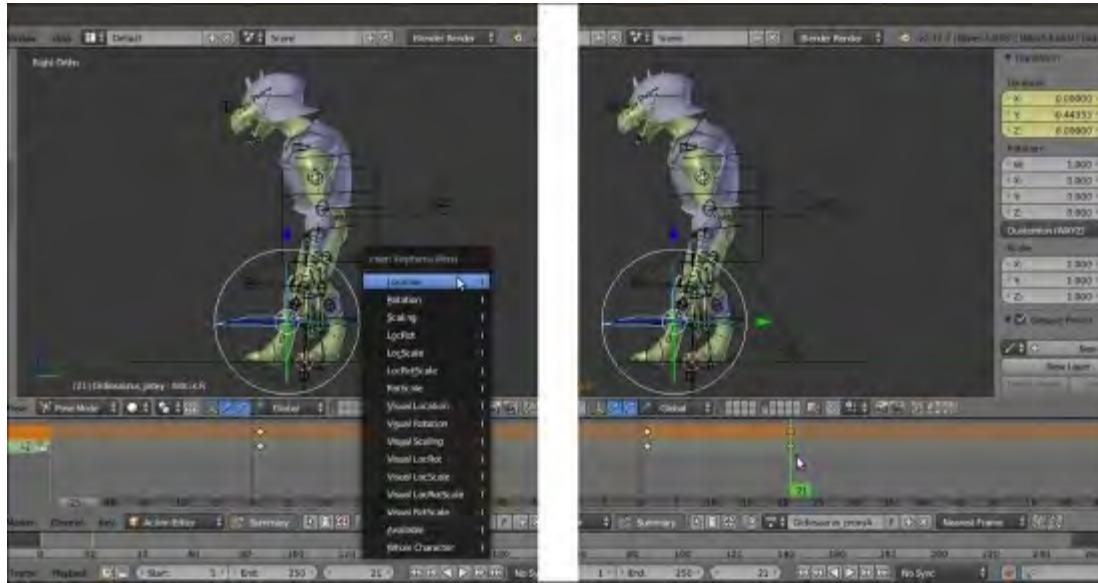
As you can see, because we enabled the **red button icon** (*Automatic keyframe insertion for Objects and Bones*) in the **Timeline** toolbar, every time we move, rotate, or scale a bone, a keyframe for **Location**, **Rotation**, and **Scaling** is automatically added to the **Action**. This can be handy, but also results in a lot of useless keyframes, for example, for most of the rig bones, we need to set keys for the **Location** and/or the **Rotation**, but very rarely for the **Scaling**.

4. Put the mouse cursor inside the **Curve Editor** area of the **Graph Editor** and press the **A** key to deselect everything.
5. **Shift + left-click** on the **Scale** and **Quaternion Rotation** items in the **Gidiosaurus_proxy** Channel Region to select them, then press **X** to delete them:



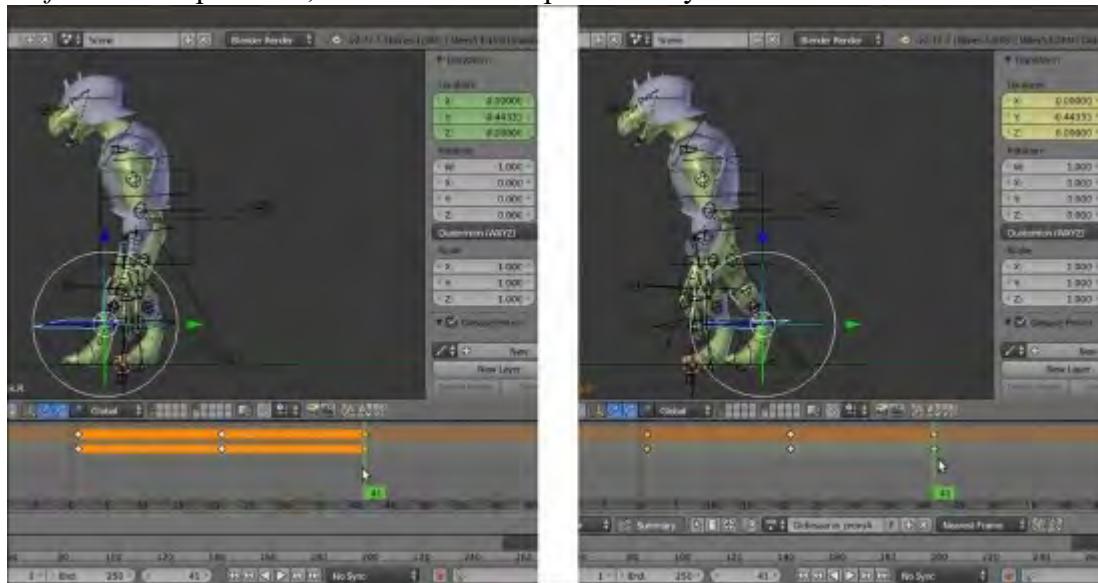
Deleting the useless transformation channels

6. Switch back to the **3D View**, and in the **Transform** subpanel in the **Properties** sidebar (and in the **Transform** subpanel under the **Bone** window in the main **Properties** panel), now only the **Location** slots are highlighted in yellow.
7. Disable the **red button icon** (*Automatic keyframe insertion for Objects and Bones*) in the **Timeline** toolbar.
8. Go to frame **21** by grabbing and moving the **Time Cursor** inside the **Timeline** window or the **Action Editor** window, or by typing the frame number inside the *Current Frame* button on the **Timeline** toolbar.
9. Select the **foot_ik.R** control bone and by using the widget, move it forward on the global y axis for around **-0.440**.
10. Press **I** and in the **Insert Keyframe Menu**, select the **Location** item; this adds a second key to the **foot_ik.R** bone at frame **21**, but this time only for **Location**:



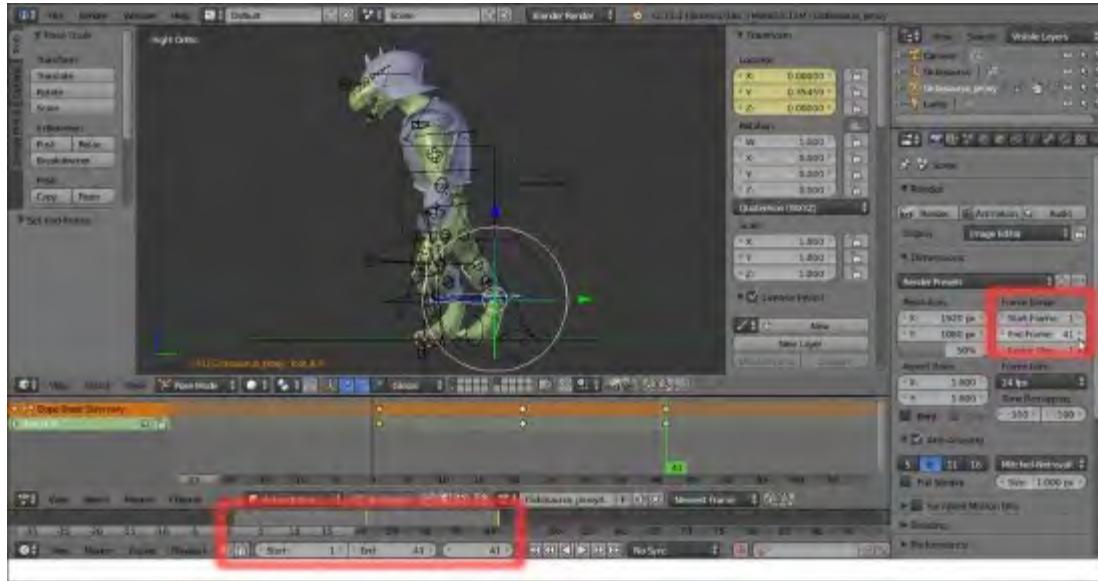
Setting a Location only key through the Insert Keyframe Menu pop-up

11. Go to frame **41**, right-click to select the key at frame **1** in the **Action Editor** window, and press **Shift + D** to duplicate it, then move the duplicated key to frame **41**.



Creating poses at different frames by duplicating keys

12. Put the mouse cursor in the **Timeline** and press the **E** key to set the total length of the animation to the current frame position:



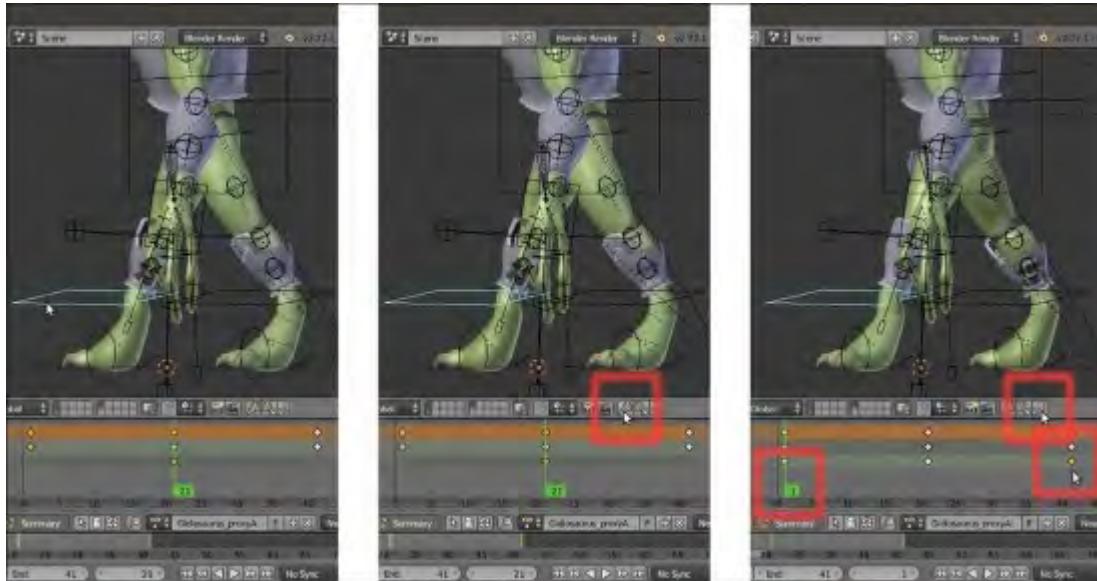
Setting the action total length in frames

13. Still at frame **41** (but this being a cycle, frame **1** could also be fine) and with the **foot_ik.R** bone selected, click on the *Copy the current pose of the selected bone to copy/paste buffer* button on the 3D viewport toolbar.
14. Go to frame **21** and select the **foot_ik.L** bone, then click on the *Paste the stored pose on to the current pose* button at the extreme right side of the 3D viewport toolbar to paste a **mirrored pose**.
15. Press the **I** key and in the pop-up menu, click on the **Location** item to add a new key:



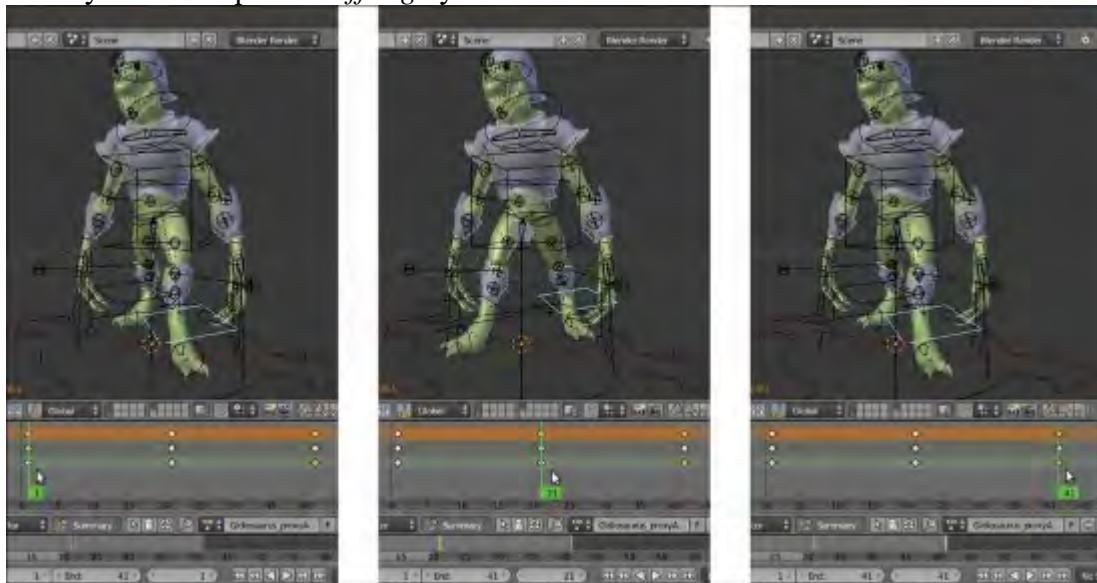
Copying a pose and pasting it reversed

16. Now, still at frame **21**, select the **foot_ik.R** bone and click on the *Copy the current pose of the selected bone to copy/paste buffer* button on the 3D viewport toolbar.
17. Go to frame **1**, select the **foot_ik.L** bone, and again click on the *Paste the stored pose on to the current pose* button to paste the reversed pose, then press **I** and insert a **Location** key.
18. Select and duplicate the new key at frame **1** for the **foot_ik.L** bone and move the duplicated one to frame **41**:



Creating new keyframes by copying, pasting, and duplicating pose keys

At this point, by scrolling the **Time Cursor** in the **Timeline**, in the **Action Editor** window, or by clicking on the **Play Animation** button in the **Player Control** on the **Timeline** toolbar, we can already see a complete *shuffling* cycle of the movement of the feet of the **Gidiosaurus**:



The Gidiosaurus' walk cycle with sliding feet

19. Now go to frame 1, select the **torso** bone, and lower it on the z axis for almost **-0.200**, then assign a position key.
20. Select the just added **torso** bone key in the **Action Editor** window, press **Shift + D** to duplicate it, and move the duplicate to frame **21**, then repeat for frame **41**:



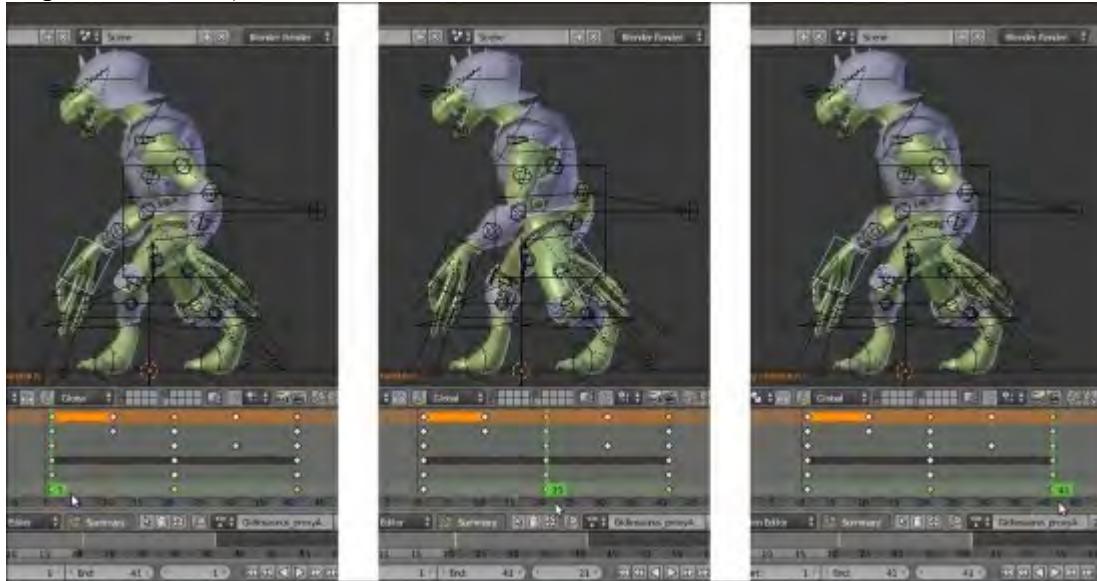
Animating the torso

21. Go to frame **11**, select the **foot_ik.R** bone, and move it on the z axis for **0.200**, then assign a position key.
22. As we already did at steps 16 and 17, copy the bone pose, go to frame **31**, and paste it reversed, then assign a position key to the **foot_ik.L** bone.



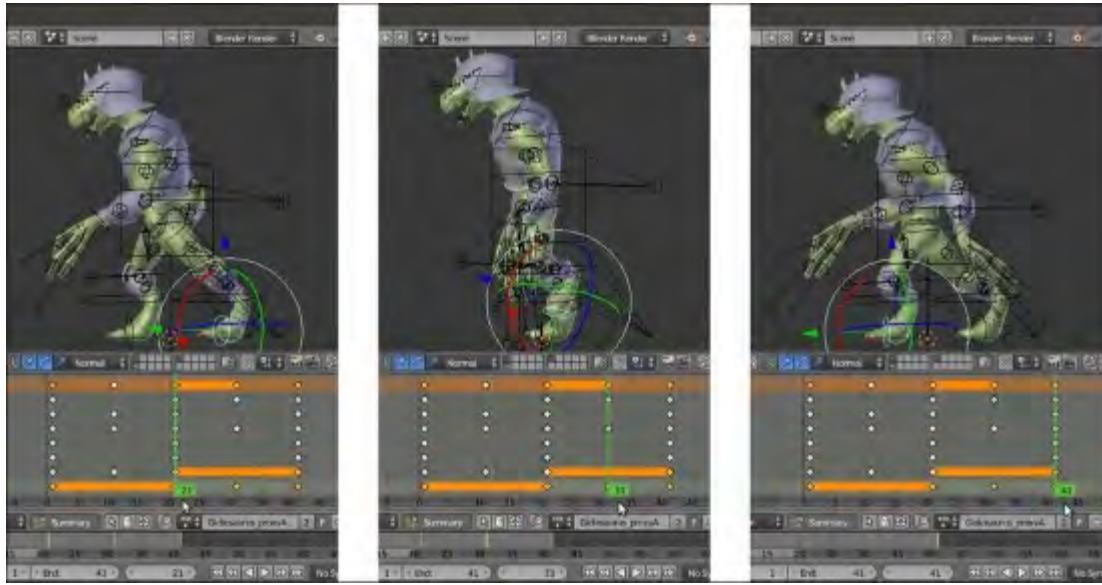
Assigning more translation keys

23. Working in the same manner, select the **hand_ik.R** and **.L** bones and animate them according to the **Gidiosaurus**' walk (note: as for any average walk cycle, in the opposite position with respect to the feet):



Animating the arms to complete the walk cycle

24. Reselect the **torso** bone, go to frame **1**, and move it forward for **0.240** on the **y** axis. Assign a new position key (to overwrite the old one), then delete the keys at frames **21** and **41** and substitute them with duplicates of the new frame **1** key.
25. Go to frame **11** and move the **torso** bone for almost **0.200** upward on the **z** axis. Duplicate the key for frame **31**.
26. Go to frame **1** and select the **toe.R** bone, then assign a rotation key. Go to frame **11** and rotate the bone on the normal **x** axis (the red circle in the widget tool with **Transform Orientation** set to **Normal**) for **75°**. Go to frame **21** and press **Alt + R** to clear the rotation pose and assign a rotation key. Use **Shift + D** to duplicate the last added key and move the duplicated one to frame **41**.
27. Select the **toe.L** bone and assign a rotation key at frame **1**, then go to frame **21** and repeat. Copy the **toe.R** pose at frame **11** and paste it reversed for the **toe.L** bone at frame **31**, then assign a cleared rotation pose key at frame **41**:



Adding the in-between poses for the feet

28. Following the previous procedures, set keys for the position and/or the rotation of all the affected bones, also adding movements such as the rotation of the **torso** and of the **hips**, the position of the **pole target** for legs and **arms**, the swinging of the **head** to compensate for the body's lateral movements, the closed **mouth** and the open **eyelids**, and so on:



The first phase of the walk cycle animation is almost done

The animation cycle, at this point, looks really stiff and *robotic*. This is simply because everything *happens at the same time*, that is, in the same frame, as you can easily see in the

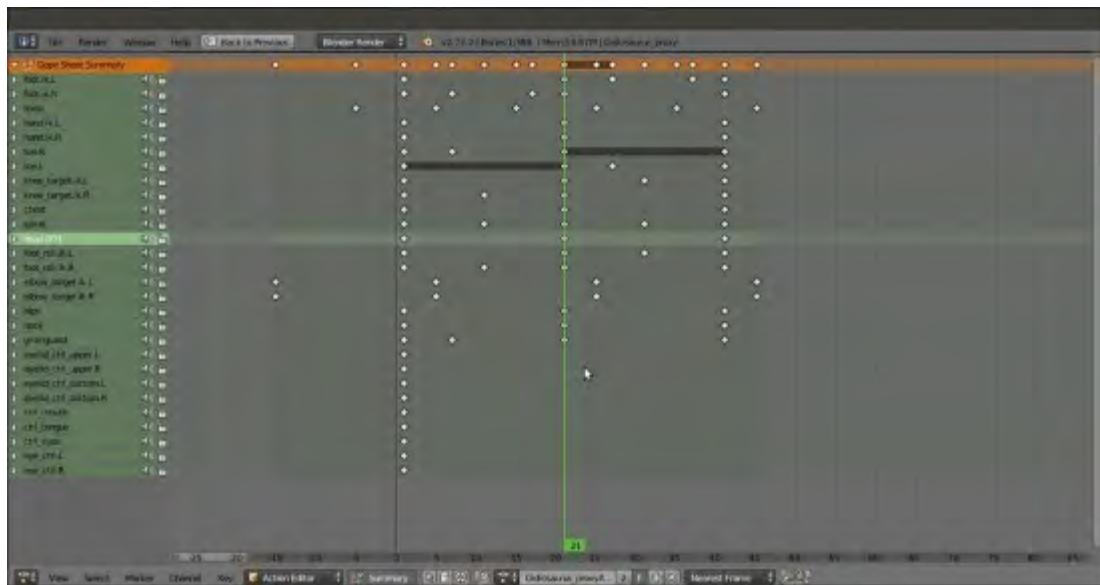
Action Editor window (to enlarge a window, put the mouse cursor inside it and press *Ctrl + Up Arrow*; to go back, press *Ctrl + Down Arrow*):



The maximized Action Editor window with the walk cycle action

To make the animation look more realistic and natural, we must offset some of the keys to make the different actions *happen at different times*; for example, the **torso** bone goes down a few frames later than the **foot** touching the ground, and goes up a few frames later as well, the same for the **head** swinging, and so on.

29. To offset the affected keys, simply select and/or *Shift-select* and move them for the required frames, forward or backward in the **Action Editor** window. Here, a bit of testing is needed to reach the right number of frames (usually in the range of **3-5** frames, by the way).
30. Where a *hole* happens at frame **1** in the action channel for a bone because of the dislocation of the keys, simply duplicate the last right side key of that bone and move it to the appropriate **negative frame** position. That is, to the left side of frame **0**, and be sure that the relative item, **Allow Negative Frames**, is enabled in the **Editing** tab of the **User Preferences** panel, as you can see in the following screenshot for the **torso** and for the **elbow_target_ik** bones:



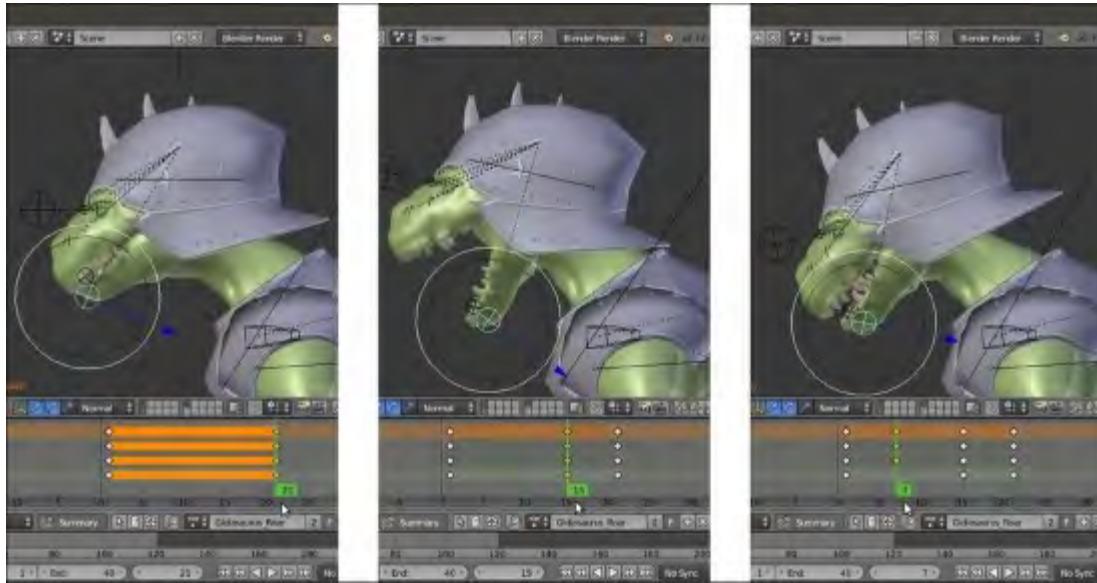
Duplicated keys moved to the Negative Frames space

31. Rename the action `Gidiosaurus_Walkcycle`. To better check the playing animation, go to the **Timeline** toolbar to set the end frame for the total length of the animation to **40** frames, because frame **1** and frame **41** are the same poses.
 32. Save the file.

At this point, we have made our first action with the **Gidiosaurus** character, and it's a 41 frame-long walk cycle meant to be repeated in loops for longer animations.

Because in the next recipe we are going to use the **Non Linear Action Editor (NLA Editor)** to re-use the **action datablocks** to build the final animation, we need now to create some more actions to be mixed with the walk cycle one.

33. Activate the **Fake User** for the **Gidiosaurus_Walkcycle** action by clicking on the **F** icon button to the side of the action datablock on the **Action Editor** toolbar, then click on the **X** icon button to unlink the action datablock.
 34. Put the mouse cursor in the 3D viewport and press the **A** key to select all the control bones, then press **Alt + G**, **Alt + R**, and **Alt + S** to clear any position, rotation, or scale and restore the rig default pose (actually, the only control bones using the **scale** operator for the animation are the **fingers**, which we haven't animated so far).
 35. Be sure to be at frame **1** and zoom to the character's **head**, select the **head.001** and **neck** bones, and assign a rotation key, then select the **ctrl_mouth** bone and assign a position key.
 36. Rename the action **Gidiosaurus_Roar**, then enable the **Fake User**; use **Shift + D** to duplicate the keys and move the duplicated ones to frame **21**.
 37. Go to frame **15** and rotate the **head.001** and **neck** bones clockwise to raise the **head**, then open the **mouth** wide by moving the **ctrl_mouth** bone down.
 38. Go to frame **7** and rotate the **head.001** and **neck** bones counterclockwise a bit to lower the **head**:



The Gidiosaurus_Roar action

We have now built a roar action for the **Gidiosaurus**, but it happens in only **21** frames, so it's really too fast. Although it is possible to scale any action strip in the **NLA Editor** window, in this case it's better to do it directly in the basic action itself.

39. In the **Action Editor** window, put the **Time Cursor** to frame **1**, then press the **A** key to select all the keys of the action. Press **S | X | 2 | Enter** to scale the action of the double to frame **41**:



Scaling the action on the position of the Time Cursor

40. Now that the action length has been doubled, we can move some keys of a few frames and also animate the movement of the character's **tongue** a bit during the *roar*:



Animating the tongue

41. Again, click on the X icon button to unlink the action datablock, select all the bones, then press *Alt + G* and *Alt + R* to clear the poses.
 42. *Shift-select* the **thumb.R** and **.L**, **f_index.L** and **.R**, and **f_middle.L**, and **.R** control bones and add a **Scaling** key. Rename the newly created action **Gidiosaurus_Fingers** and enable the **Fake User**:



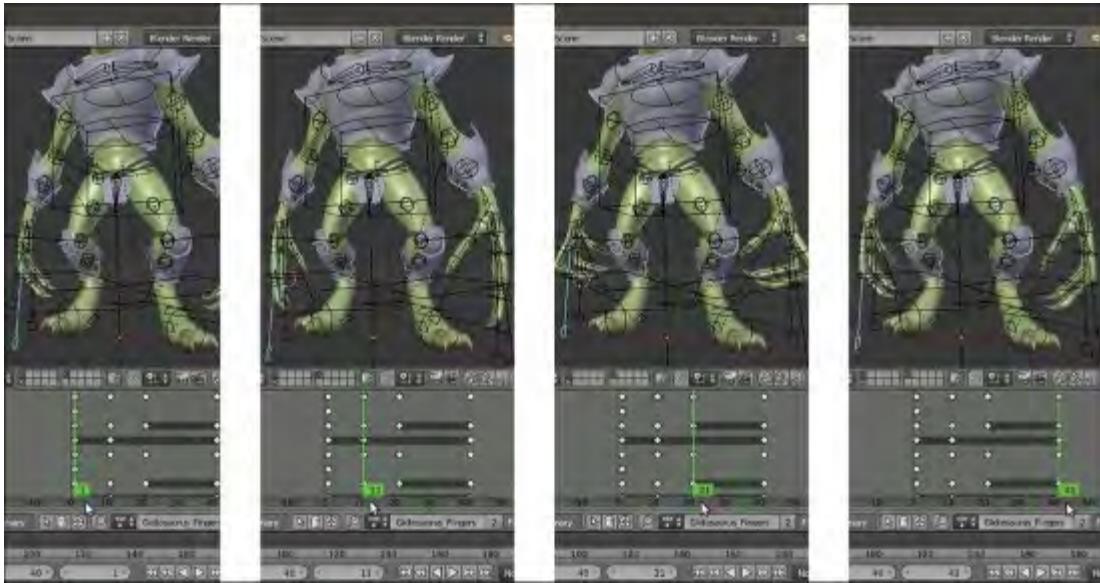
Renaming the fingers action and enabling the Fake User

43. Now *Shift-select* for both the **.L** and **.R** bones, **thumb.01**, **thumb.02**, and **thumb.03**, **f_index.01** and **f_index.02**, **f_middle.01** and **f_middle.02**, and the **palm** control bones, then add a **Rotation** key:



Adding a first rotation key for all the fingers at the same time

44. Now that we have all the finger bones' names in the **Action Editor** list-tree to the left (the **Channel Region**), start to click on the bone names to highlight them, for example, click on the **thumb.L** item, then press *Shift + PageUp* keys to move it to the top of the list.
45. Then highlight the **thumb.01.L** bone and by pressing the *PageUp* arrow, move it right after the **thumb.L** bone (press *Shift + PageUp* to eventually go directly to the top). Repeat with the **thumb.02.L** and the **thumb.03.L** bones, then go to the **thumb.R** bone, and so on. To move an item downward in the list-tree, simply press the *PageDown* key instead (or *Shift + PageDown* to go directly to the bottom).
46. Repeat the ordering until you have *grouped* the bones' names *by finger* in the list-tree, to make it easier to individuate them in the **Action Editor** window, then use *Shift + D* to duplicate all the keys and move the duplicated ones to frame **41**.
47. Go to frame **21**, select **thumb.L**, **.R**, **f_index.L**, **.R**, **f_middle.L**, and **.R** bones and press **S** to scale them to **0.900**. Assign a **Scaling** key, then select and rotate the other control bones, and assign **Rotation** keys (be aware that the previous scaling bones can also be rotated). Also, by using the *Copy/Paste* technique already shown, build a kind of *creepy hands* animation:



The "creepy hands" animation made by rotating and scaling the bones controls

48. When you are done, thanks to the re-ordering we made in the **Channel Region**, go to the **Action Editor** window and move groups of keys based on their finger group; in short, to avoid the *everything-at-the-same-time* issue, dislocate the timing of one finger with respect to the others:



Offsetting the finger' keys

49. After this, click on the *Display number of users of this data* button to create a new copy of the action and change the name to **Gidiosaurus_Fingers.L**. In the **Channel Region**, *Shift-select* all the **.R** bones items and delete them (**X** key), then enable the **Fake User**.

50. Click on the double arrows icon to the left side of the datablock name (*Browse Action to be linked*) and reselect the **Gidiosaurus_Fingers** action.
51. Again, click on the *Display number of users of this data* button to create a new copy of the action and change the name to **Gidiosaurus_Fingers.R**. *Shift-select* all the .L bones items and delete them. Enable the **Fake User** and click on the X icon button to unlink the action datablock.
52. Save the file.

To have a look at the completed walk cycle of the **Gidiosaurus** and the other actions, open the `Gidiosaurus_walkcycle_final.blend` file provided with this cookbook.

How it works...

An **Action** is a bones **F-Curves** datablock created at the same moment any animation key is added through the **Insert Keyframe Menu** (*I* key) or the **red button icon** (*Automatic keyframe insertion for Objects and Bones*) in the **Timeline** toolbar. The newly created **Action** automatically takes the name from the object itself (**Gidiosaurus_proxy** in this case) plus the **Action** suffix.

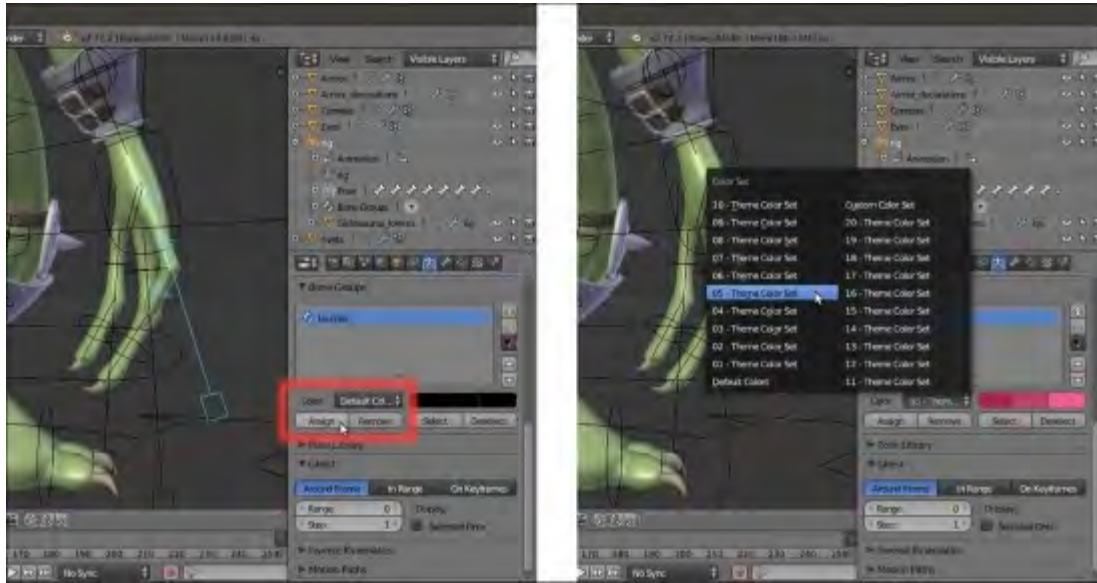
The **Actions** are stored inside the `.blend` file, but thanks to the **Fake User** they don't necessarily need to be linked to the rig to be preserved after saving and closing the file.

Note that the scaling operation for the selected keys of an **Action** in the **Action Editor** window (and the same for the **Graph Editor** and the **NLA Editor**) use the **Time Cursor** position as the pivot point. Also note that even though we did it in our recipe, it wasn't mandatory in this case to declare the *x* (horizontal) axis for the scaling.

There's more...

Organizing the bones' names in the list inside an action in the **Action Editor** window is a good way to quickly find the required item, but it can be improved even further by **Bone Groups**:

1. Open the `Gidiosaurus_library.blend` file and go to the **Outliner**; click on the eye icon to the side of the **rig** item to unhide it.
2. Select the **rig** and go to the **Object Data** window, then in the **Bone Groups** subpanel, click on the + icon to add a **bone group**.
3. Double click on it to rename it **thumbs**, then go into the 3D viewport and *Shift-select* all the **thumbs'** bones.
4. Click on the **Assign** button, then click on the **Color** slot to choose a **Theme Color Set** from the pop-up menu:



Choosing a Theme Color Set for the Bone Group

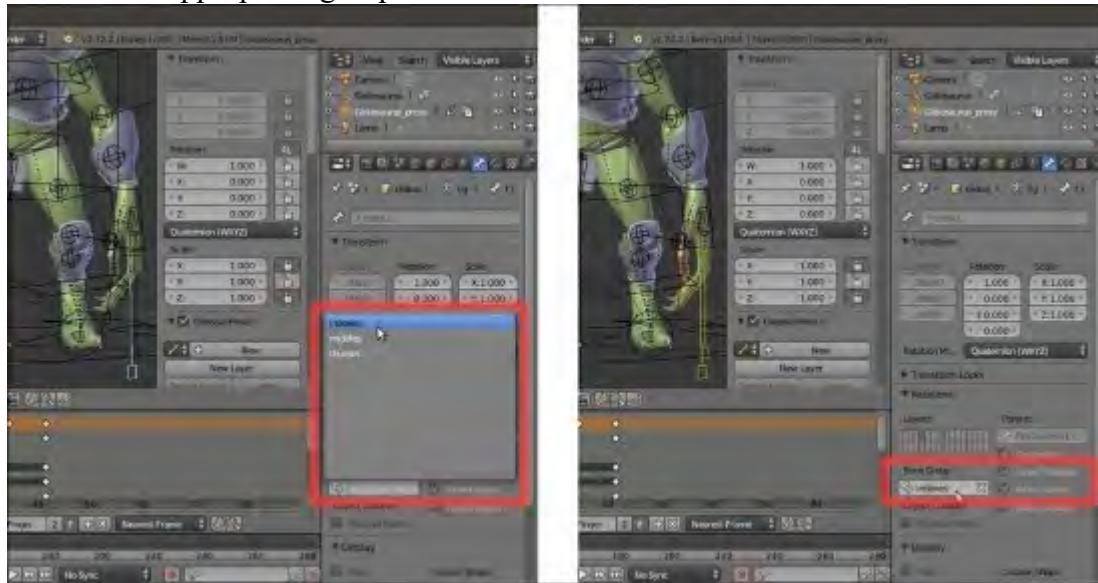
- Repeat the steps from 2 to 4 for the other two fingers, thus creating the **indexes** and **middles** bone groups and selecting a different **Theme Color Set** option for each group:



Three different Bone Groups

- In the **Outliner**, hide the **rig** item again and save the file.
- Re-open the **Gidiosaurus_walkcycle.blend** file; the colored bones don't show in the **proxified rig**, and this is because we had already proxified it and only later assigned the **bone groups** to the library file.

8. The solution to fix this is simply to select the affected bones one at a time and by going to the **Relations** subpanel under the **Bone** window, click on the **Bone Group** empty field to select the name of the appropriate group:



Reassigning the Theme Color Set to the proxified bones

By the way, it is always better to do the **Bone Groups** before the **proxy**, if possible.

The colors of the **Bone Groups** also show as *background color* for the bone channels inside the **Action Editor** window, making it a lot easier to select all the bones of a group; just be sure to have the **Show Group Colors** item enabled in the **View** menu on the **Action Editor** toolbar:



The Group Colors enabled for the bones

You can find the library with the colored fingers' control bones under the alternative file named `Gidiosaurus_library_colors.blend`.

See also

The walk cycle and the other actions we built in this recipe are, from an *animation point of view*, very simple and basic, not meant to teach you *how to animate* but only to show enough of Blender's tools for you to easily start animating a rigged character.

If you want to go deeper into the animation process, in Blender or not, here are some links to visit:

- http://www.fjasmin.net/walk_cycle_tutorial/index.html
- <http://cgcookie.com/blender/2010/01/24/learning-basic-animation-and-a-walk-cycle/>
- <http://wiki.blender.org/index.php/Doc:2.6/Manual/Animation>
- <http://wiki.blender.org/index.php/Doc:2.6/Manual/Animation/Techs>

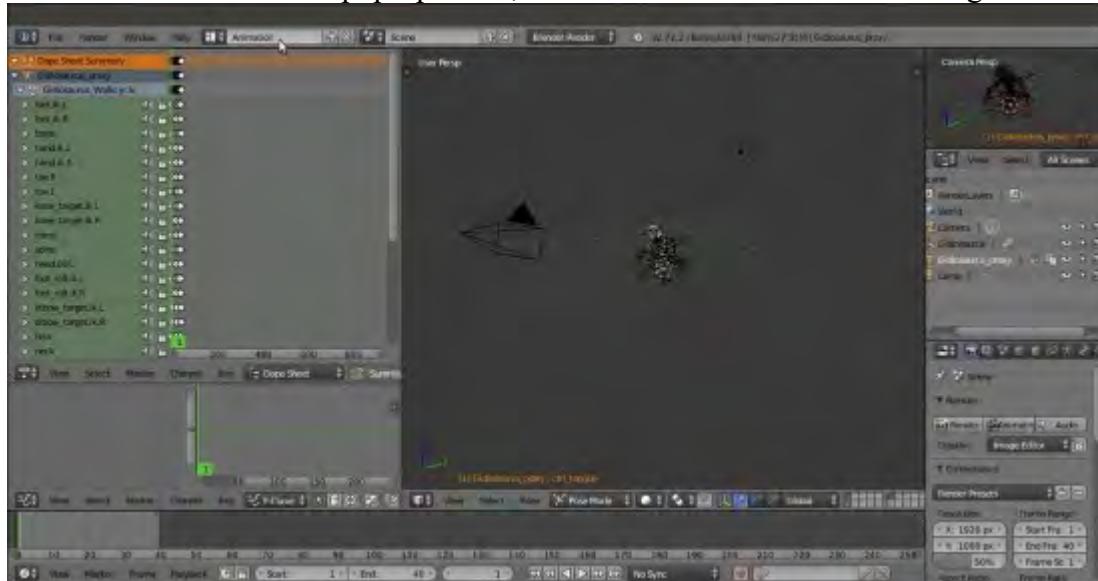
Tweaking the actions in Graph Editor

In the previous recipe, we built **Actions** by setting position, rotation, and/or scaling keys, which Blender **interpolates through F-Curves** to create the character's animation. In this recipe, we are going to see the **Graph Editor** window, a tool to modify these **F-Curves** to fix errors or fine-tune the movements of the animated character.

Getting ready

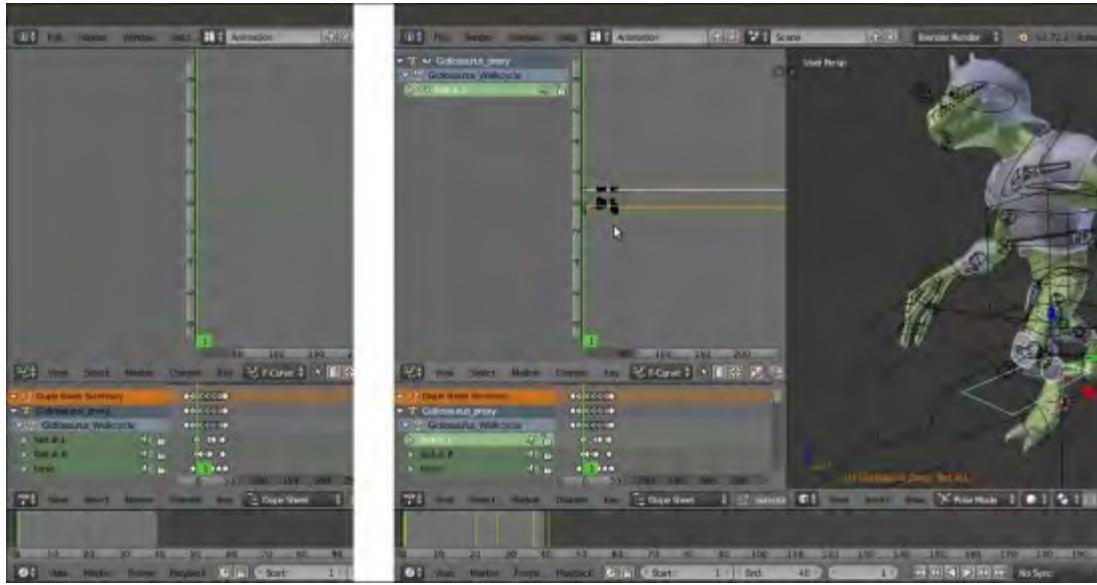
Open the `Gidiosaurus_walkcycle.blend` file.

1. If it's not already loaded, in the **Action Editor** window, load the **Gidiosaurus_walkcycle** action.
2. Go to the top main window header and click on the two little arrows to the left side of the button labeled as **Default**. In the pop-up menu, select the **Animation** item to change the screen layout:



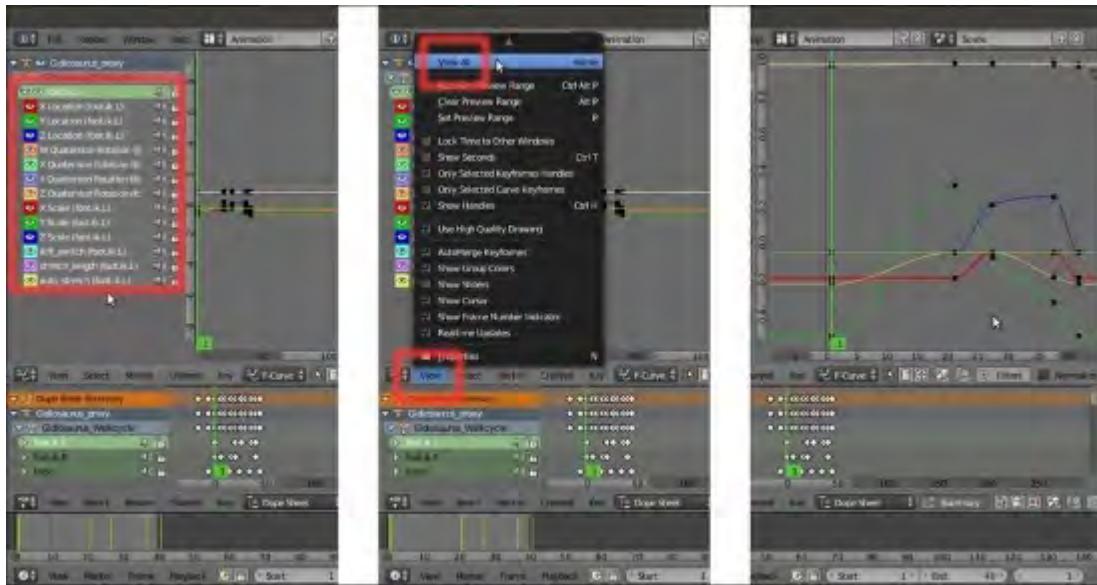
The premade Animation screen layout

3. Press the **Ctrl** key and left-click on the **top right corner** of the **Dope Sheet** window, then drag the mouse towards the **Graph Editor** window below to switch the two windows.
4. Go to the 3D viewport and zoom to the character to select the **foot_ik.L** bone; the **F-Curves** for the selected bone appear in the **Graph Editor** window:



The F-Curve of the animation keys of the selected bone in the Graph Editor window

5. Expand the **foot_ik.L** item in the **Graph Editor** list-tree by clicking on the little arrow to the side of the item itself, then click on the **View** item in the toolbar and select the **View All** item to better visualize the curves inside the **Curve Editor** area:



The list of the available F-Curves for the selected bone and the automatic zoom through the View All item

6. Hide (by clicking on the eye icon) and/or delete (select using left-click and the X key) the unnecessary curve items such as (at least in this case) **X Scale**, **Y Scale**, **Z Scale** or **ikfk_switch (foot_ik.L)**, and so on. Join the unnecessary windows together and adjust the size of the **Edit**

Area (the part with the keyframes) in the **Dope Sheet** to make them more easily readable. Optionally, enable the **Normalize** item in the **Graph Editor** toolbar to show all the **F-Curves** in a normalized **-1** to **1** range.

7. Save the file as `Gidiosaurus_F-Curves.blend`:

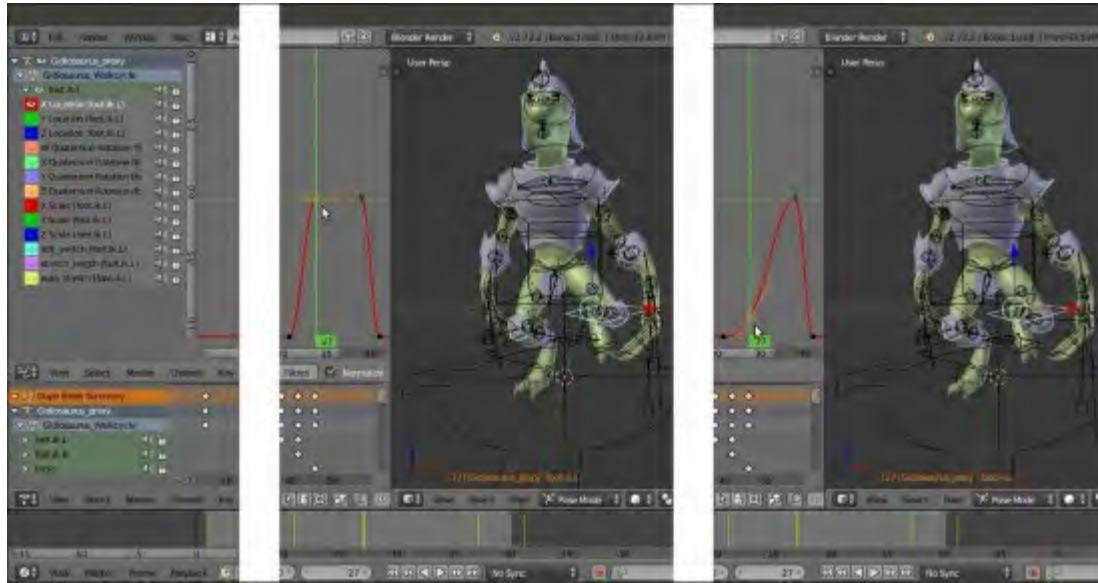


The Animation screen with a bit of customization

How to do it...

By selecting a curve name item and/or hiding the others in the **Graph Editor** list-tree, we can concentrate on one curve at a time. For example, what if we want to change the position of the **right foot** at frame **27** on the global **x** axis, when it's high off the ground?

1. Just left-click on the **X Location (foot_ik.L)** item in the list to highlight it and/or simply hide the others. Right-click on the curve keyframe/control point at frame **27** to reveal the **handles**, then press **G | Y** to move the keyframe handles on the vertical axis and see the foot move accordingly in the viewport on the global **x** axis:



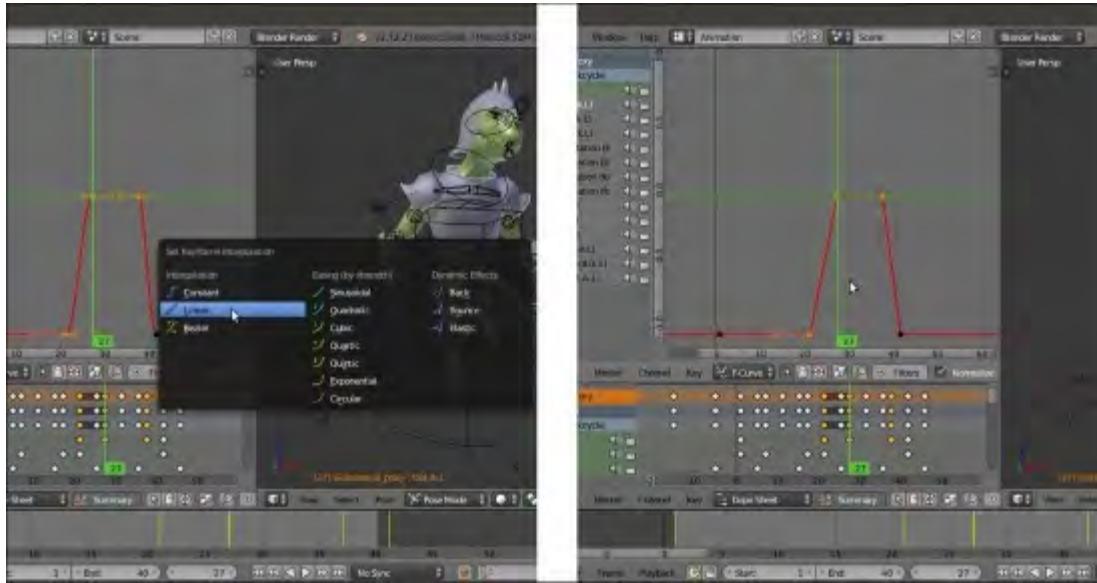
Editing the points of the F-Curve to tweak the bone's position

2. Or else, right-click only on one of the handles of the keyframe to move it and change the curve's envelope:



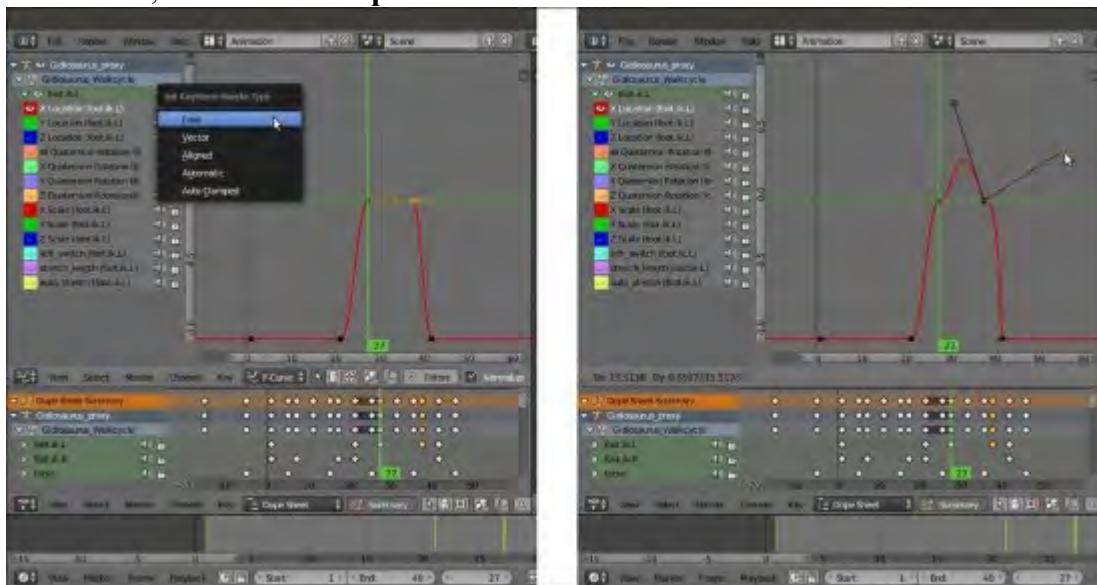
Changing the envelope of the F-Curve by modifying one of the point's handles

3. By *Shift*-selecting two or more keyframes of an **F-Curve** and pressing the **T** key, it is possible to set the **interpolation** type through the **Set Keyframe Interpolation** pop-up menu; by default the **F-Curves** are **Bezier**, but they can be switched to **Linear** or **Constant**. There are also **Easing** and pre-made **Dynamic** effects:



Changing the F-Curve Interpolation mode

- Finally, the handles' type can also be set through the **Set Keyframe Handle Type** menu by pressing the **V** key; by default the handles are **Aligned**, but they can be set as **Free**, **Vector**, **Automatic**, and **Auto Clamped** too:



Changing the handle's type to further tweak the curve's envelope

See also

- <http://wiki.blender.org/index.php/Doc:2.6/Manual/Animation/Editors/Graph>

Using the Non Linear Action Editor to mix different actions

It's finally time to use the **NLA Editor** to compose a longer animation using the actions we built in the previous recipes.

Getting ready

As usual, first let's prepare the screen:

1. Start Blender and press **Ctrl + Alt + U** to call the **User Preferences** panel; in the **Editing** tab, enable the **Allow Negative Frames** item.
2. Click on the **Save User Settings** button and close the panel.
3. Load the **Gidiosaurus_F-Curves.blend** file and switch the **Graph Editor** to the **NLA Editor** window, and the **Dope Sheet** below it with the **Action Editor** window.
4. If necessary, click on the **X** icon button in the **Action Editor** window toolbar to unlink any action from the rig and clear the pose:



The Animation screen with the (still empty) NLA editor window

How to do it...

We are going to add **Action strips** to the **Gidiosaurus_proxy** rig, so it's mandatory to have at least one bone selected (any one, but in this case, it's the **ctrl_mouth** bone):

1. Put the mouse cursor in the **Track Region (NLA-stack)** of the **NLA Editor** window, right under where it shows **Gidiosaurus_proxy | <No Action>** items, and press **Shift + A** to add a **NlaTrack** channel:



Adding a first track to the NLA Editor window

Now, take a moment and load the **Gidiosaurus_walkcycle** action in the bottom **Action Editor** window to see the action extension; it starts at frame **-15** and ends at frame **45**.

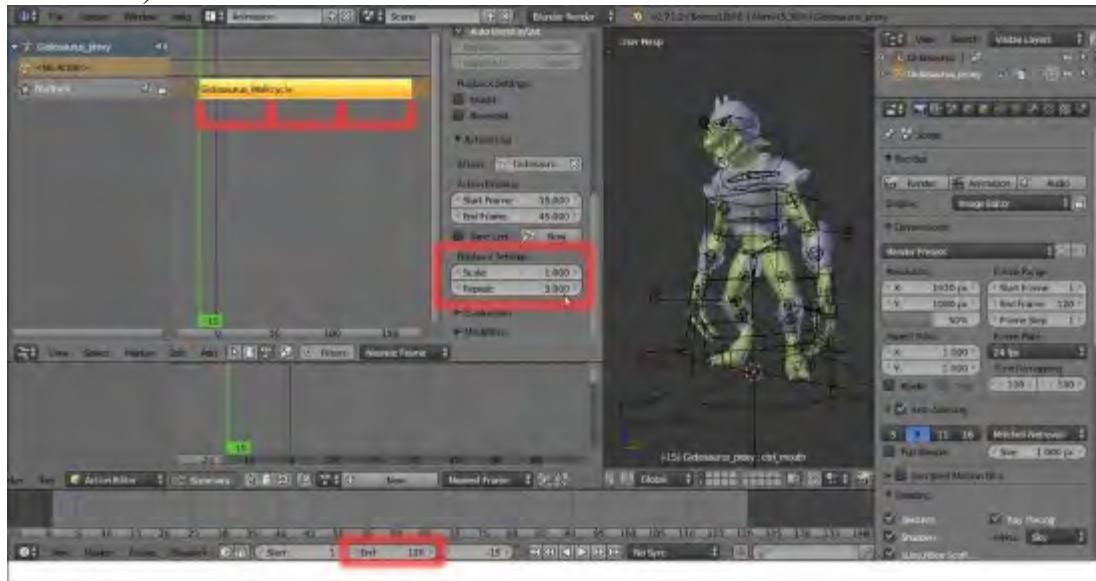
2. Unlink the action in the **Action Editor** and move the **Time Cursor** to negative frame **-15**; put the mouse cursor in the **Strip Edit** area to the right side of the **NlaTrack** item and again press **Shift + A**. From the pop-up menu, select the **Gidiosaurus_Walkcycle** item:



Loading the Gidiosaurus_walkcycle action into the track

A yellow action strip, with the **Gidiosaurus_Walkcycle** name superimposed, is added to the track at the **Time Cursor** location (the vertical green bar showing the frame number). If you now press the **Play** button in the **Player Control** on the **Timeline** toolbar, the animation starts at frame 1 and because the animation is only **40** frames long, it loops correctly, exactly as if the action was loaded in the **Action Editor** window.

3. If not already present, press the **N** key to call the **Properties** sidepanel of the **NLA Editor** window, right-click on the action strip to select it, and then go to the **Action Clip** subpanel. Under **Playback Settings**, set the **Repeat** value to **3.000**.
4. Click on the **End** button in the **Timeline** toolbar and change the frame value from **40** to **120** (**40** frames x **3**):

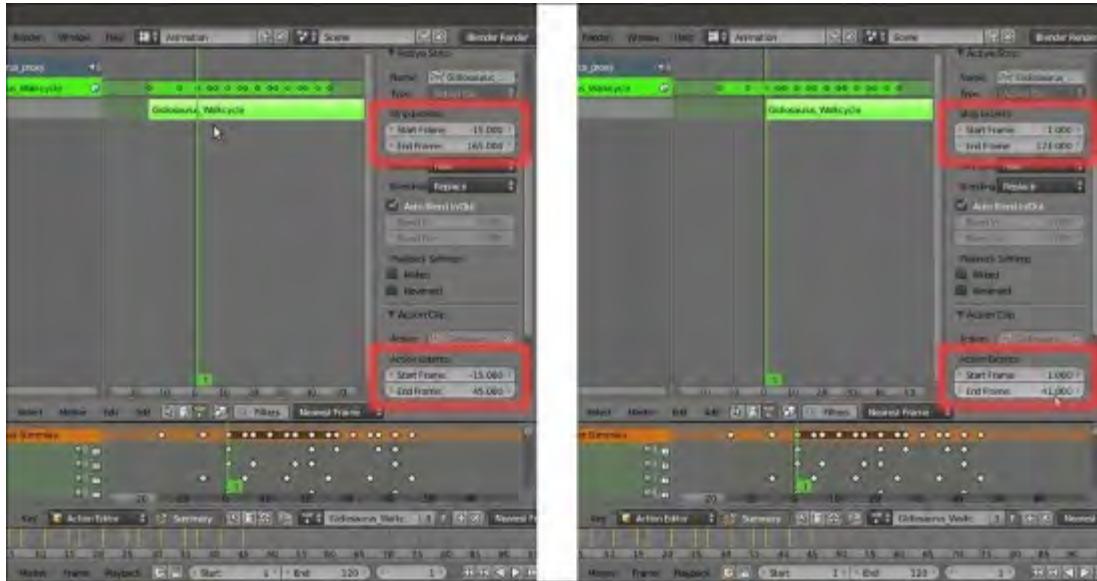


The Gidiosaurus_walkcycle action set to be repeated three times

If you press the **Play** button now, the animation is repeated **3** times *but* it doesn't loop correctly anymore because the **negative frames** keys are also included, in both the second and third repetitions. This is because we loaded the action at frame **-15**, so this is the **Start Frame** value for **Action Extents (Start Frame = -15, End Frame = 45)**.

Hence, some adjustment must be done to the action strip:

5. First, move the **Time Cursor** to frame **1**; with the strip selected, press the **Tab** key to go into **Edit Mode** and make the inner keys of the strip visible, both above the strip in the **NLA Editor** window, and as an **Action** in the **Action Editor** window. This way it's simpler to understand what keys are at what frame, and so on.
6. Second, go to the **Active Strip** subpanel and under the **Strip Extents** item, set the **Start Frame** value to **1.000**.
7. Go to the **Action Clip** subpanel and under the **Action Extents** item, set **Start Frame** to **1.000** as well and the **End Frame** value to **41.000**:



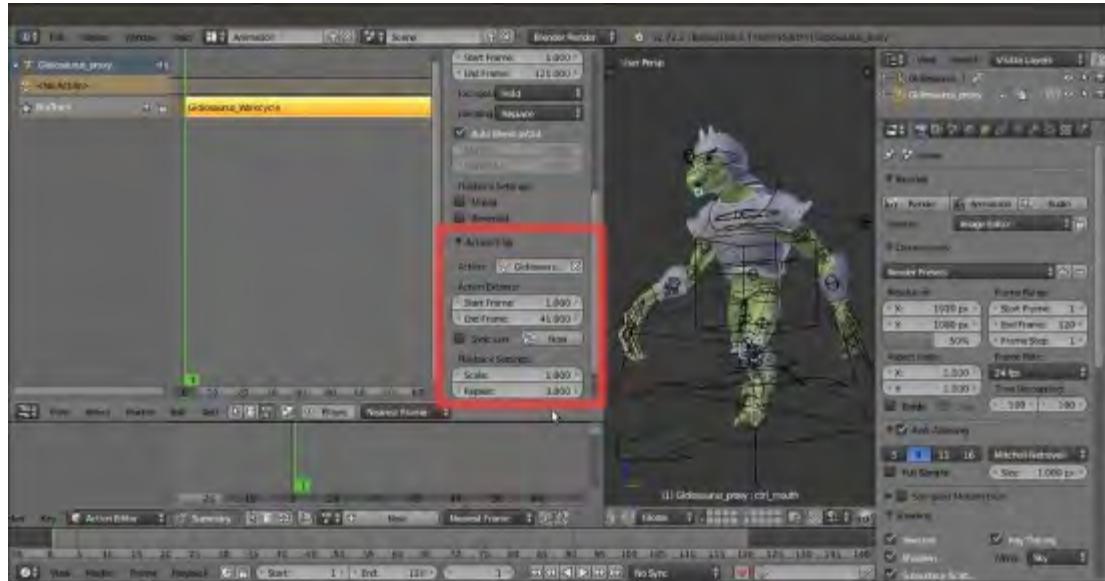
The action in Edit Mode and the Strip Extents and Action Extents values in the Properties subpanel of the NLA window

8. Press **Tab** to go out of **Edit Mode**.

Now, the walk cycle animation loops correctly for all the **120** frames, and obviously it is also possible to loop it even more by raising the **Repeat** value.

So, the correct and fastest procedure would have been, from the start:

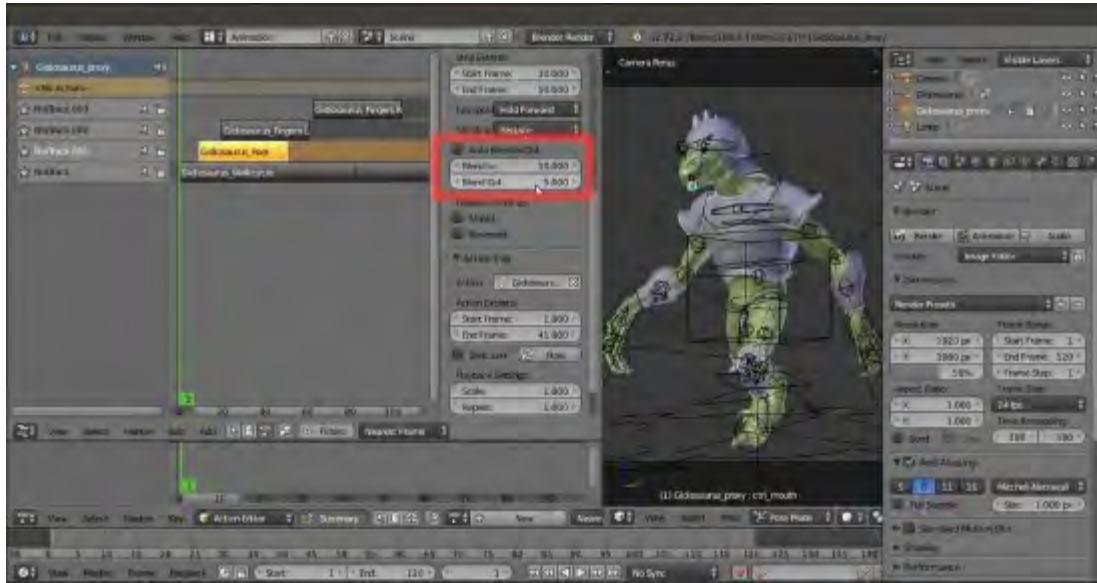
1. At frame **1**, load the action strip in the **NLA Editor** window.
2. In the **Properties** sidebar, under the **Action Extents** item in the **Action Clip** subpanel, set the **Start Frame** value to **1.000** and the **End Frame** value to **41.000**.
3. Under **Playback Settings**, set the **Repeat** value to **3.000** and the total length of the animation to **120** frames in the **End** button of the **Timeline** toolbar:



Recapitulating the action extents values to be set

Now, let's see how to add the other actions:

1. Put the mouse cursor under the **NlaTrack** item and press **Shift + A** to add a new track (**NlaTrack.001**); load the **Gidiosaurus_Roar** action strip and move it (**G** key) to start at frame **10**.
2. Select the **Gidiosaurus_Walkcycle** strip and in the **Active Strip** subpanel, disable the **Auto Blend In/Out** item but leave the values as **0.000**. Select the **Gidiosaurus_Roar** strip and disable the **Auto Blend In/Out** item as well, then set the **Blend In** value to **10.000** and the **Blend Out** value to **5.000**.
3. Add **two** more tracks, select the **NlaTrack.002** track and load the action strip **Gidiosaurus_fingers.L**, then select the **NlaTrack.003** track and load the action strip **Gidiosaurus_fingers.R**.
4. Select the **Gidiosaurus_fingers.L** strip and in the **Active Strip** subpanel, disable the **Auto Blend In/Out** item, and leave the values as **0.000**; repeat for the **Gidiosaurus_fingers.R** strip.
5. Move the two strips separately in different positions inside the **120** frames animation range.



Setting the Blend In and Blend Out values to mix the other actions

6. Press the **Play** button in the **Player Control** on the **Timeline** toolbar to watch the composited animation and save the file as **Gidiosaurus_NLA.blend**.

At this point it could be possible to start to render at least some **OpenGL** preview to see the result, but there are still several steps missing in our workflow before we reach the final goal, from the **texturing** to the **shaders**, **lighting**, and finally **beauty-rendering** and **compositing**; all stuff that we'll see in the next few chapters.

See also

- http://wiki.blender.org/index.php/Doc:2.6/Manual/Animation/Editors/NLA_Editor

Chapter 10. Creating the Textures

In this chapter, we will cover the following recipes:

- Making a tileable scales image in Blender Internal
- Preparing the model to use the UDIM UV tiles
- Baking the tileable scales texture into the UV tiles
- Painting to fix the seams and to modify the baked scales image maps
- Painting the color maps in Blender Internal
- Painting the color maps in Cycles

Introduction

In this chapter, we are finally going to create the textures for the **Gidiosaurus** character, meaning all the image textures that we'll need later, to build the **shaders** for the body and for the **armor**. Basically, the essential images we need are:

- A grayscale reptilian scales image to be used as a bump map and to color the skin
- Painted image textures for the skin diffuse coloration
- A tileable image for the worn armor metallic surface
- A bump image for the armor decoration patterns

In this chapter, we'll focus on the skin of the **Gidiosaurus**, and the last two textures for the **armor** will be treated in the next chapter.

The most difficult and tedious part is, no doubt, rendering the **scales** on the **Gidiosaurus** skin; I mean, if we had to paint the scales one by one. Instead, we'll try to obtain the complex scales pattern with the minimum effort possible, using a couple of techniques to speed up the work.

Trying to keep things clear, from now on we'll use *two different texture folders*: the usual `textures` one and a `textures_making` folder, where the latter is used to contain the images we need during the process to produce the final image textures.

Making a tileable scales image in Blender Internal

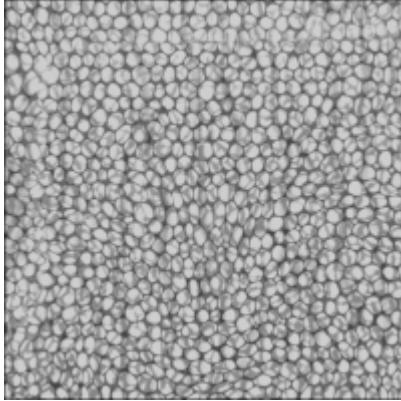
So, the first thing to do is to obtain a **tileable** grayscale **reptilian scales** image; we'll start from an already existing image obtained from an old and larger texture I had painted in **Gimp** for a dinosaur model some years ago... but that's a different story.

In any case, if you prefer, you can paint a new reptilian scales image from scratch by using painting software such as **Gimp** or **Photoshop** or open source applications such as **MyPaint** (<http://mypaint.intilinux.com/>) or **Krita** (<https://krita.org/>).

Getting ready

We are taking for granted that in your **Blender User Preferences** window, you still have the **Import Images as Planes** addon enabled; if not, start Blender and just enable it as already explained in [Chapter 1, Modeling the Character's Base Mesh](#). Then, follow these steps:

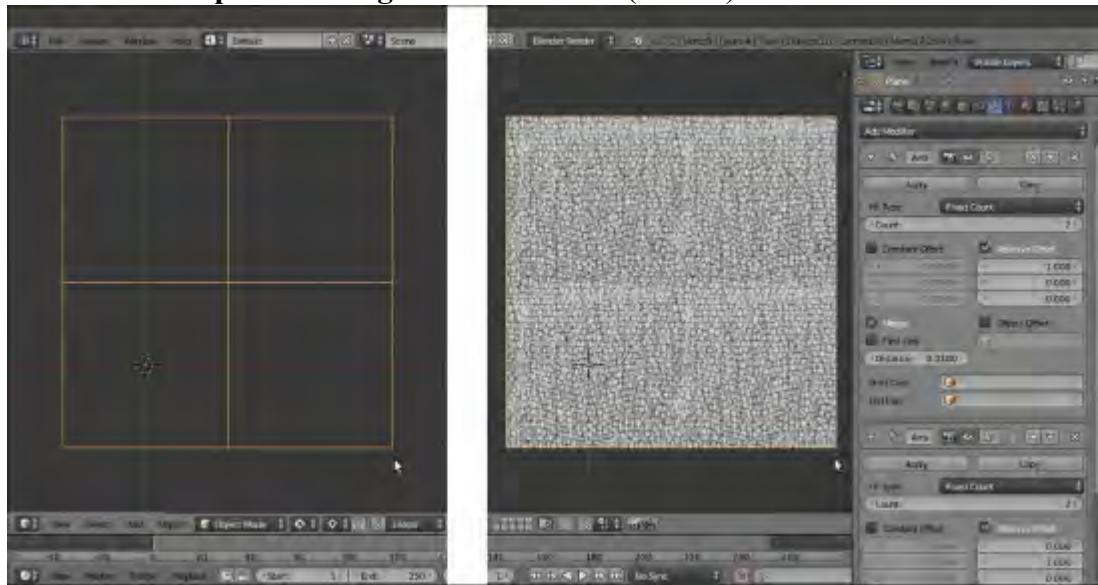
1. Select and delete the default **Cube** primitive in the scene. Select the **Camera** and the **Lamp** and move them to the **6th** scene layer.
2. Click on the main **File** menu and then on the **Import** item; select the **Images as Plane** item.
3. Browse to the **textures_making** folder and select the provided **scales.png** image texture, which is a gray painted scales image:



The "scales.png" image provided with this cookbook

4. Press the period (.) key on the numpad to center the view on the selected **Plane** and then the 7 key on the numpad to switch to the **Top Ortho** view.
5. Go to the **Object Modifiers** window and assign an **Array** modifier to the **Plane**; check the **Merge** item and leave all the other settings as they are.
6. Click on the **Copy** button to assign a new identical **Array** modifier, and in the new **Array** modifier, under the **Relative Offset** item, change **X** to **0.000** and **Y** to **1.000**.
7. Save the file as **4886OS_10_scales_tiles_01.blend**.

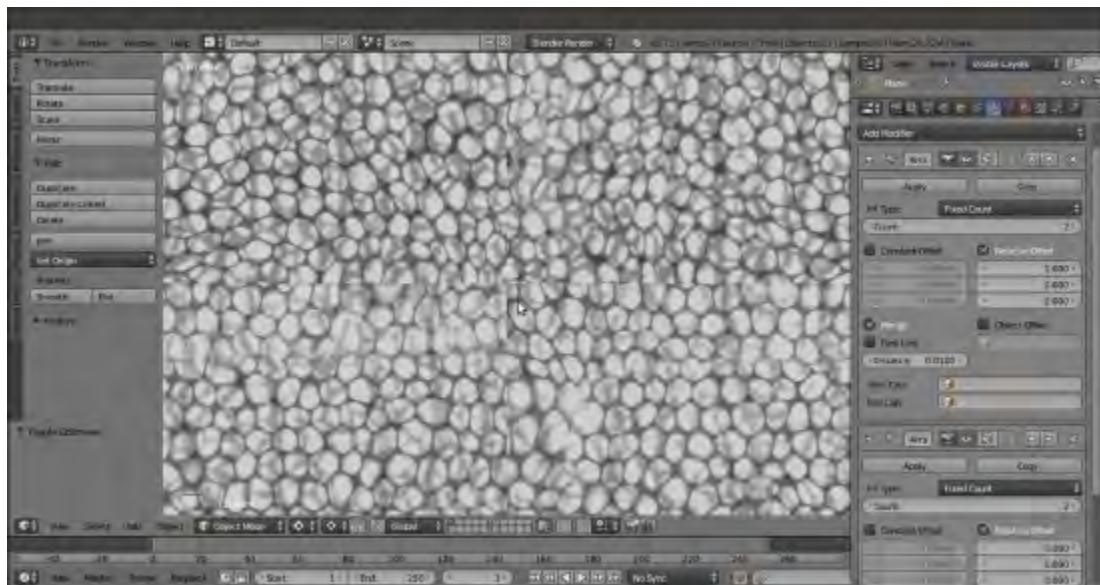
8. Press the period (.) key again on the numpad to center the view on the *enlarged Plane*, then switch the **Viewport Shading** mode to **Texture** (*Alt + Z*):



The UV mapped Plane with the Array modifiers assigned

As you can see in the preceding screenshot, the mapped **Plane** is now repeated 4 times.

By zooming towards the middle seams, it's clear that the mapped scales image is not tileable yet:



The visible seams at the borders of the Plane instances

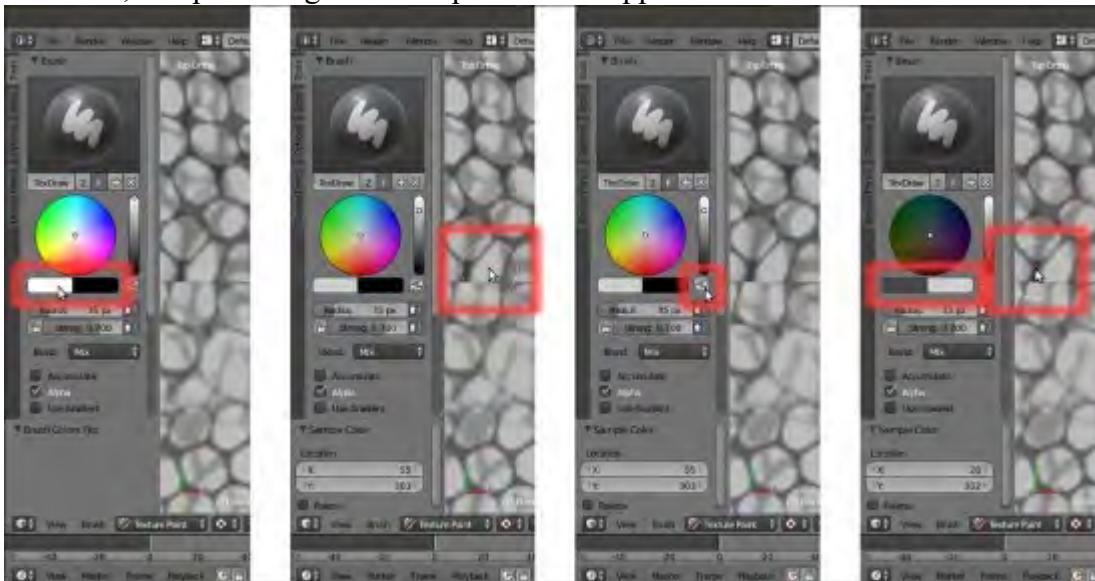
Tip

Just in case the previous screenshot is not readable enough, open the provided 4886OS_10_scales_tiles_01.blend file to have a better look.

How to do it...

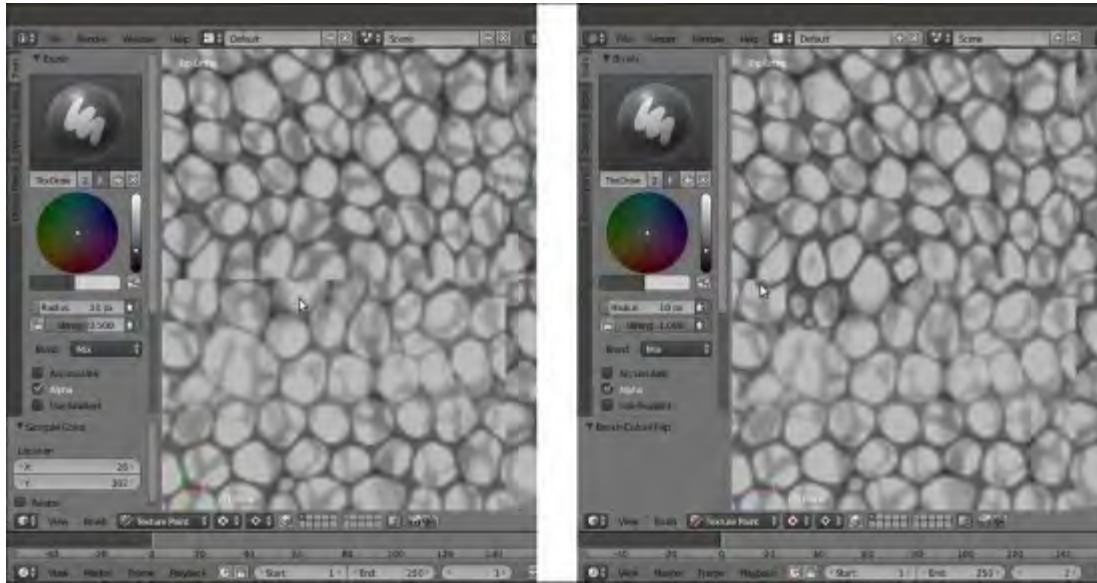
At this point, the file is ready and we can start to paint on the image to make it tileable:

1. Click on the mode button (*Sets the object interaction mode*) on the viewport toolbar to select the **Texture Paint** mode item.
2. Put the **mouse cursor** on a bright value in the scales image and press the *S* key to sample it, then go near the color selector in the **Brush** subpanel under the **Tool Shelf** to the left and click on the *Toggle foreground and background brush colors* button (the one with the two opposing arrows) to switch the active color. Otherwise, simply press the *X* key, put the mouse cursor on a dark area, and press *S* again to sample it as the opposite color.



Sampling the light and dark colors of the image

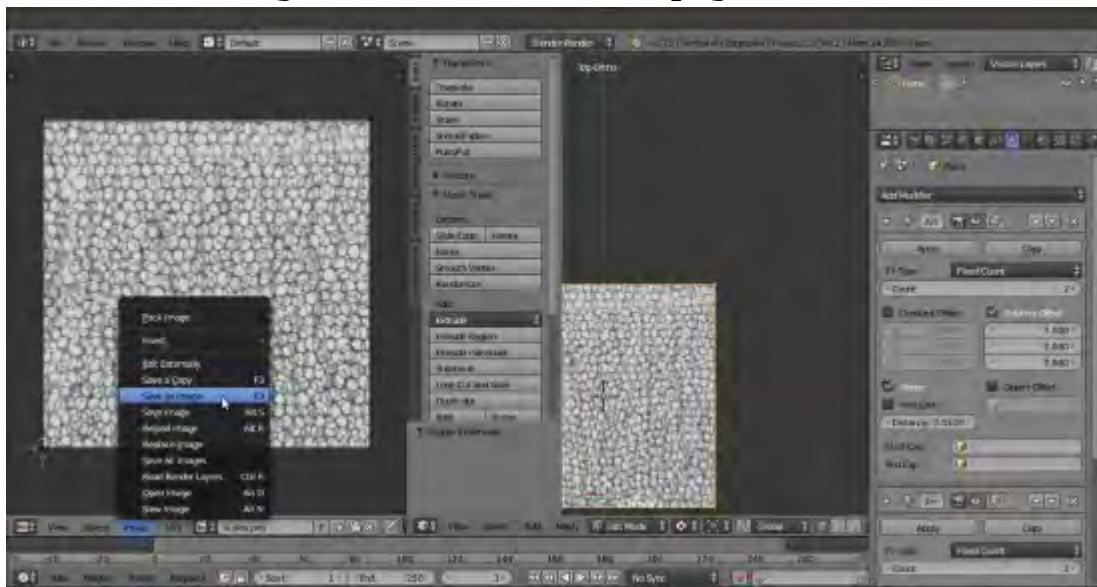
3. Set the brush's **Radius** and **Strength** values, and if you are using a **graphic tablet**, be sure to have the **two tablet pressure sensitivity** buttons at the sides of the previous items enabled.
4. Start to paint by fixing the scales on the image at the seams areas. Because we can also paint on the **Planes** duplicated by the **Array** modifier, and because we are always painting on the same instanced image, it's quite simple to visually join the scales at the four sides, actually making the image tileable:



Painting on the image at the borders to make the scales seamless

It's enough to fix the areas along the middle horizontal and vertical axes of the **Planes** to cover all the four edges (in fact, fixing **two** edges is automatically fixing **four**).

- When you are done, go back into **Object Mode** and open a new window with **UV/Image Editor**; press *Tab* to go into **Edit Mode**, make the revised image appear, and save it (by clicking on **Image | Save as Image** in the toolbar or simply press the *F3* key) in the **textures_making** folder as **scales_tiles.png**:



Saving the tileable scales image with a different name

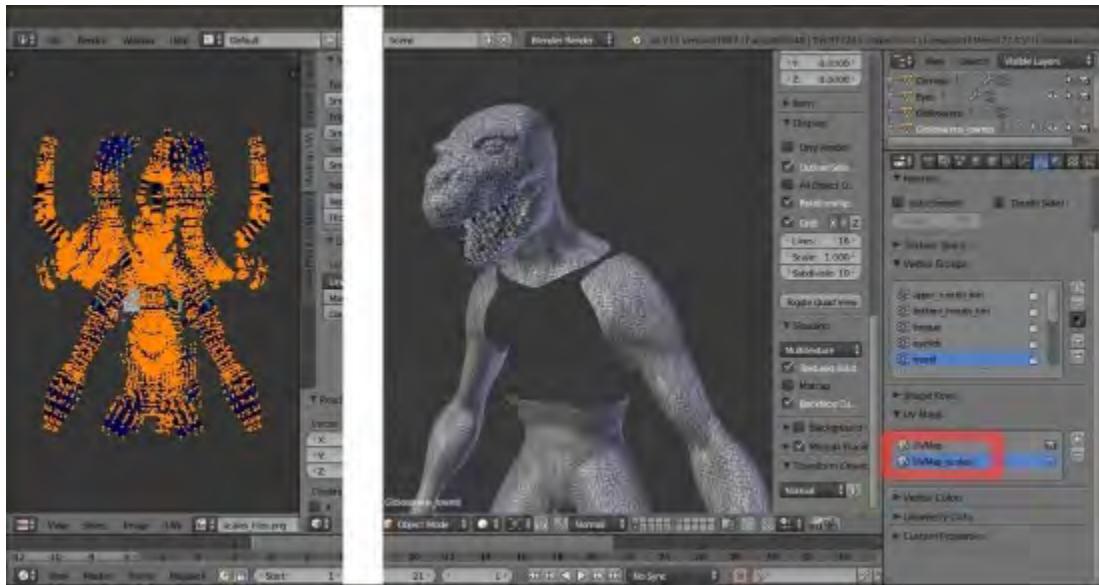
6. Save the file as 4886OS_10_scales_tiles_02.blend.

How it works...

You might wonder why we didn't use the **Make Seamless** filter of Gimp (**Filters** | **Map** | **Make Seamless**) to obtain a tileable image in one click; well, the answer is simple: the **Make Seamless** plugin actually offsets and blends together whole areas of the image, and this can work in several cases, but not for a complex pattern made by scales, where a simple fading is not good enough. In this case, I prefer to paint the joining line between them by hand.

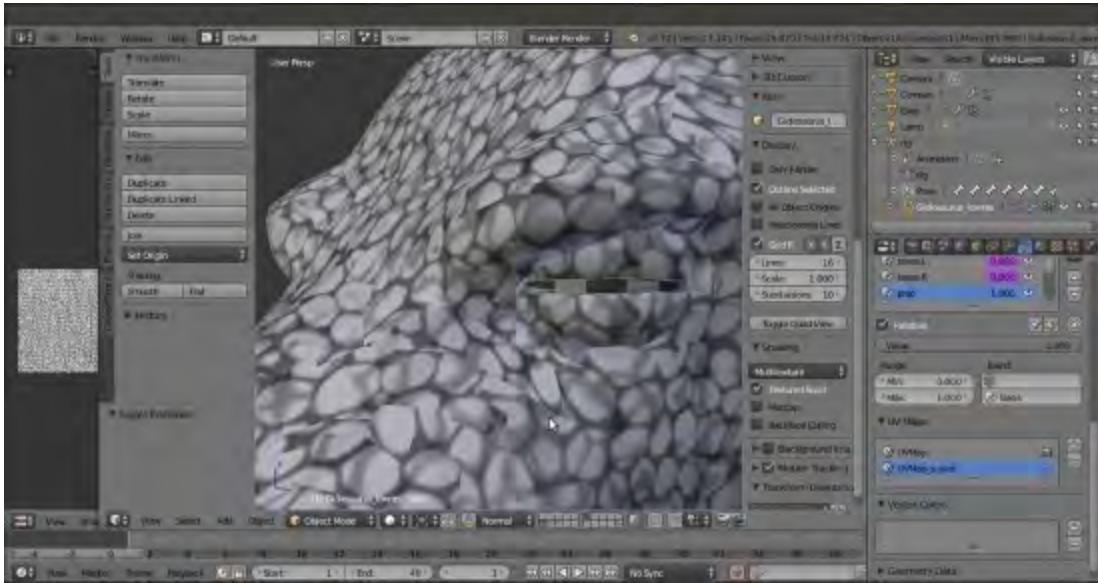
Preparing the model to use the UDIM UV tiles

In the previous recipe, we made the scales image texture seamless, ready to be seamlessly mapped on our model. If you go back to [Chapter 5, Unwrapping the Low Resolution Mesh](#), you'll remember that we assigned **two** different sets of UV coordinate layers to it: the **UVMap** layer, divided into **5 different tiles** (this is called **UDIM UV Mapping**; it's a popular standard in the industry and means **U-Dimension**), and the **UVMap_scales** layer, set up to repeat the `scales_tiles.png` image pattern at the right size on the model:



The two UV coordinates layers in the UV Maps subpanel

By zooming in and looking carefully at the result of the tiling (in **Textured Solid** mode, which is enabled under the **Shading** subpanel of the *N* viewport sidepanel), you can see that although we used a tileable scales image, we still have seams in some areas. This is obviously due to the fact that the **UVMap_scales** layer (as you can see in **UV/Image Editor** in the **Edit Mode** after selecting all the mesh's vertices) is made up of separated and overlapping islands to obtain a randomly distributed mapping of the scales on the **Gidiosaurus** skin:



The seams on the Gidiosaurus scales skin

A simple solution to fix these seams is to bake the random scales pattern on the **5** tiles of the **UVMap** layer and then use the **Paint Tool** to adjust the gaps. This a step that we have to do in any case to allow further texture modifications such as the painting of befitting facial scales around the **eyebrows**, the **eyes**, the **nostrils**, and so on, but let's go in order.

To be able to bake and then paint on the different tiles in real time through the 3D viewport, we must prepare the file a bit.

Getting ready

Start Blender and open the `Gidiosaurus_library.blend` file; save it as `Gidiosaurus_baking_scales_01.blend`:

1. *Shift-click* on the **13th** scene layer to disable it and hide the **Armor** object, then enable the **6th** scene layer to show **Camera** and **Lamp**. Split the 3D view into two windows and change the left one into a **UV/Image Editor** window (if it shows the **Render Result** image datablock, just click on the **X** icon button to unlink it).
2. Put the mouse in the 3D viewport and press the **T** key to hide the **Tool Shelf** panel, and then maximize the **UV/Image Editor** window as much as possible. Go to the **Outliner** and click on the eye icon to the side of the **Camera** item to disable its visibility in the viewport.
3. Go to the **UV Maps** subpanel under the **Object Data** window and be sure to have the **UVMap** layer selected (the one with the **5** different space tiles); if necessary, click on the camera icon to the right to enable it as the active UV coordinates layer.

How to do it...

Now, let's prepare the materials; remember that, at the moment, we are under the **Blender Render** engine and not under **Cycles**:

1. Go to the **Material** window and out of **Edit Mode**, click on the – icon button to the right of the materials datablock window (*Remove the selected material slot*) to unlink both the **Enamel** and the **Body** material datablocks.
2. Now click on the + icon button to add **5** material slots, and then add **5** materials by selecting each slot and clicking on the **New** button.
3. Starting from the top one, rename the **5** materials as **Material_U0V0**, **Material_U1V0**, **Material_U2V0**, **Material_U0V1**, and **Material_U1V1**.



Adding the 5 materials to the Gidiosaurus object

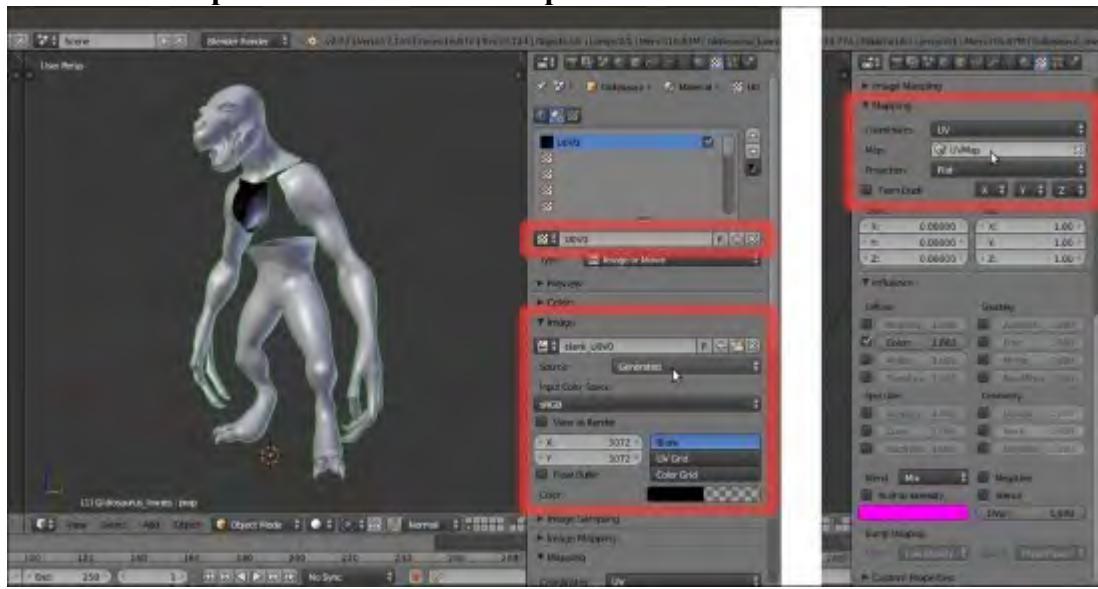
4. Select the **Material_U0V0** slot and go to the **Textures** window to click on the **New** button and add a texture.
5. Scroll down the vertical panel by rotating the middle mouse wheel and click on the **New** button under the **Image** subpanel; in the **New Image** pop-up panel, click on the **Color** slot to set the **Alpha (A)** value to **0.000**, then **Ctrl + click** on the **Width** slot and right after the default value of **1024**, type *** 3**, then press **Enter** (in Blender, you can do a math calculation for any parameter like this anywhere). Copy and paste (**Ctrl + C** and **Ctrl + V**) the result of the multiplication, **3072**, into the **Height** slot; click on the **Name** slot to write the texture name as **blank_U0V0**, then press the **OK** button at the bottom of the panel:



Adding a blank image texture to the first material

This adds a blank (*alpha background*) **3072 x 3072** pixels image as a texture on the material.

6. *Ctrl + left-click* on the **Unique datablock ID name** slot right above the **Type (Image or Movie)** slot, and rename the default **Texture** name as **U0V0**. Go down to the **Mapping** subpanel and click on the **Map** slot to select the **UVMap** item:



The "Unique datablock ID name" slot, the Image subpanel, and the Mapping subpanel

7. Go to the **UV/Image Editor** to the left side of the screen and click on the double arrows to the side of the **New** button in the toolbar; from the pop-up menu select the **blank_U0V0** item. Slide

the toolbar to the right and click on the **Image** item. In the pop-up menu, select the **Save as Image** item (or press the *F3* key) and save the image in the **texture_making** folder as **blank_U0V0.png**, then click on the pin icon button to the right to activate it (*Display current image regardless of object selection*):



The assigned blank image loaded in the UV/Image Editor window and pinned to be displayed regardless of the object selection

As the image is saved under the **Image** subpanel, the **Source** slot caption changes from **Generated** to **Single Image**.

8. Repeat the procedure for all the remaining four materials, assigning and saving a blank image texture for each material. So inside the **texture_making** folder, you have saved the images: **blank_U0V0**, **blank_U1V0**, **blank_U2V0**, **blank_U0V1**, and **blank_U1V1**.
9. Start to split the **UV/Image Editor** window until you have **5 UV/Image Editor** windows. Press the *Tab* key to go into **Edit Mode** with the mesh; put the mouse in the 3D viewport and press the *A* key to select all the mesh's vertices and therefore show the UV islands in all the **UV/Image Editor** windows.
10. Enlarge one **UV/Image Editor** window as much as possible and enable the *Keep UV and edit mode mesh selection in sync* button on the toolbar.
11. If it's the case, deselect everything, then box-select the islands (*B* key then left-click and drag the mouse) in the **U1V0** tile space. In the **Material** window, select the **Material_U1V0** slot and click on the **Assign** button. Go to the top right **UV/Image Editor** window and click on the **X** icon button on the toolbar to unlink the current image datablock (which is still **blank_U0V0**). Then click on the **Image** item on the toolbar and from the drop-down list, select the **blank_U1V0** image.
12. Press the *A* key to deselect everything and box-select the islands in the **U2V0** tile; select the **Material_U2V0** slot and again click on the **Assign** button. Go to the following image editor and unlink the current image datablock to load the **blank_U2V0** image.

13. Repeat for the other two missing tiles and material slots (note that this is not necessary for the **U0V0** ones, which are, by default, first assigned to the whole mesh and the first created material and so still remain associated to **Material_U0V0**). Then go out of **Edit Mode**.



The work-space prepared with the 5 UV/Image Editor windows with their respective blank images

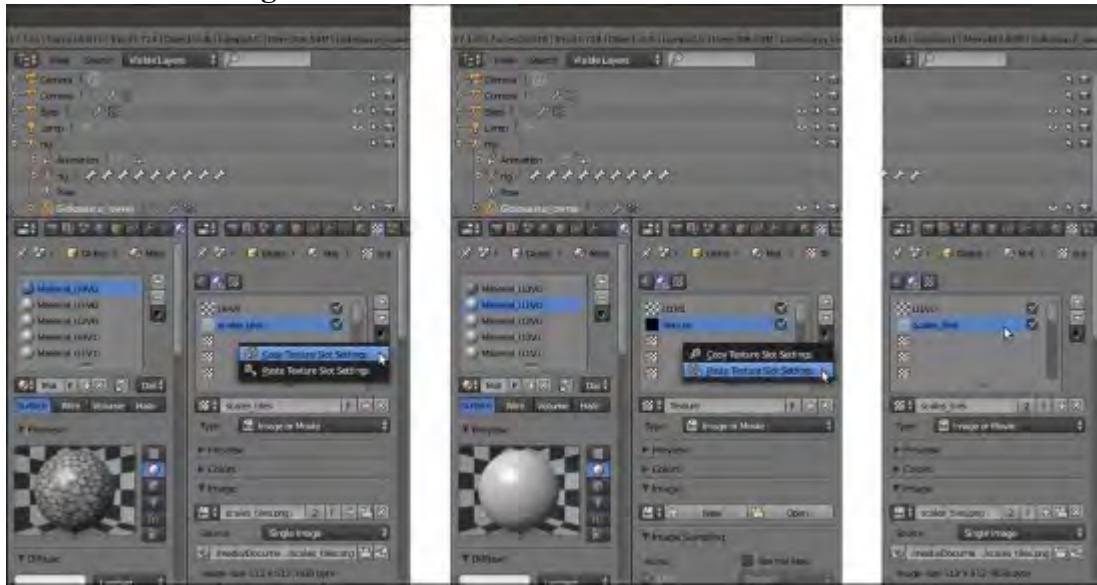
As you can see, by selecting the vertices of the UV islands in **UV/Image Editor**, the corresponding vertices on the mesh are also selected. Moreover, this makes all the UV islands visible in the image editor, even though, we haven't selected a single vertex on the mesh yet (normally, you see only the islands of the selected vertices in the image editor). This way, it's simple to associate a certain UV island with a certain material and a certain group of vertices on the mesh.

14. Go to the **Material** window and select the **Material_U0V0** slot. Go to the **Texture** window and click on the second texture slot right under the **U0V0** one. Click on the **New** button, scroll down to the **Image** subpanel, and click on the **Open** button to browse to the **texture_making** folder and load the **scales_tiles.png** image.
15. Go to the **Mapping** subpanel and in the **Map** slot, select the **UVMap_scales** UV coordinates layer. Rename the Unique datablock ID name slot as **scales_tiles**. Click on the checkbox to the side of the **U0V0** texture slot to disable it (this is just temporary but mandatory for the baking, otherwise it would create a dependency loop, that is, the *Circular reference in texture stack* message in the top main header and in the **Terminal** panel as well):



Disabling the blank texture image and loading the "scales_tiles.png" image in the first material

16. Click on the button with a black arrow pointing downward, right after the + and -icon buttons, and from the pop-up menu, select the **Copy Texture Slot Setting** item. Select the Material_U1V0 slot and then click on the second texture slot right under the U1V0 one and click on the New button. Click again on the black arrow button and this time, select **Paste Texture Slot Setting**:



Copying and pasting the "scales_tiles" texture slot to the other materials

17. Repeat this copy and paste for the other three materials, and also remember to disable the first texture slot for all the materials.

18. Press *Tab* to go out of the **Edit Mode** and save the file.

How it works...

Thanks to the pin icon button that is enabled for each loaded image, it's possible to keep the different images visible at the same time. At this moment, the **5** different PNG images are blank, so this isn't particularly evident; it will be a lot more clear when we start to actually paint on the model through the 3D viewport.

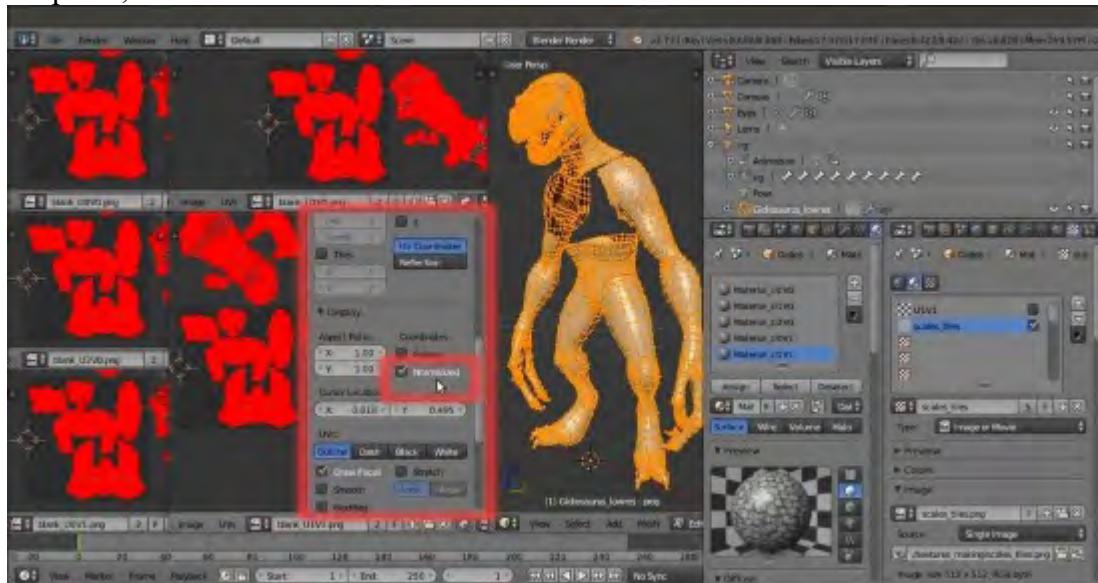
Baking the tileable scales texture into the UV tiles

What we have to do now is to bake the `scales_tiles.png` image map (used in all the materials and mapped on the **UVMap_scales** coordinates layer) on the **5** tiles of the **UVMap** coordinates layer.

Getting ready

At this moment, Blender is not able to bake automatically outside of the default **U0V0** tile space yet, so a bit of additional work is needed; nothing particularly difficult by the way. The steps are as follows:

1. Press **Tab** to go into **Edit Mode** again and then put the mouse in the **blank_U0V0 UV/Image Editor** window; press the **N** key to call the **Properties** sidepanel and under the **Display** subpanel, check the **Normalized** item:



The **Normalized** item in the **Display** subpanel under the **N Properties** sidepanel of the **UV/Image Editor** window

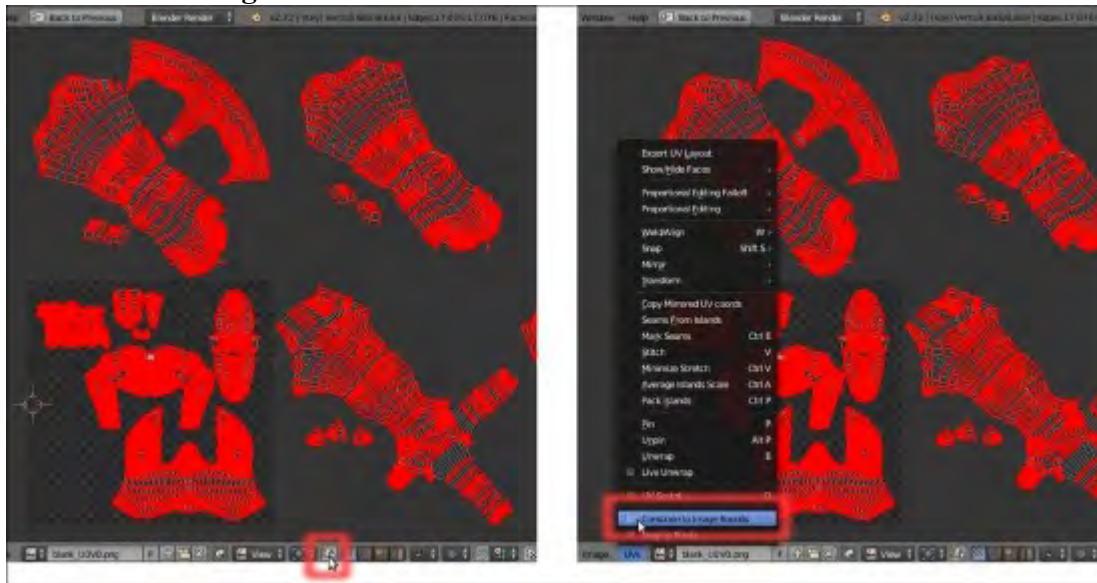
2. Press **N** again to hide the **Properties** sidepanel. Go to the **UV Maps** subpanel under the **Object Data** window and click on the **+** icon button to the right to add a new UV coordinates layer (**UVMap.001**), then rename it **UVMap_temp** (or whatever you prefer).

How to do it...

We are now going to create a new UV coordinates layer for the baking by moving all the islands in the outside tiles to the space of the default one; but before we go on, we must be sure about two things:

- In the toolbar of the **blank_U0V0** image editor window, the *Keep UV and edit mode mesh selection in sync* button must now be disabled

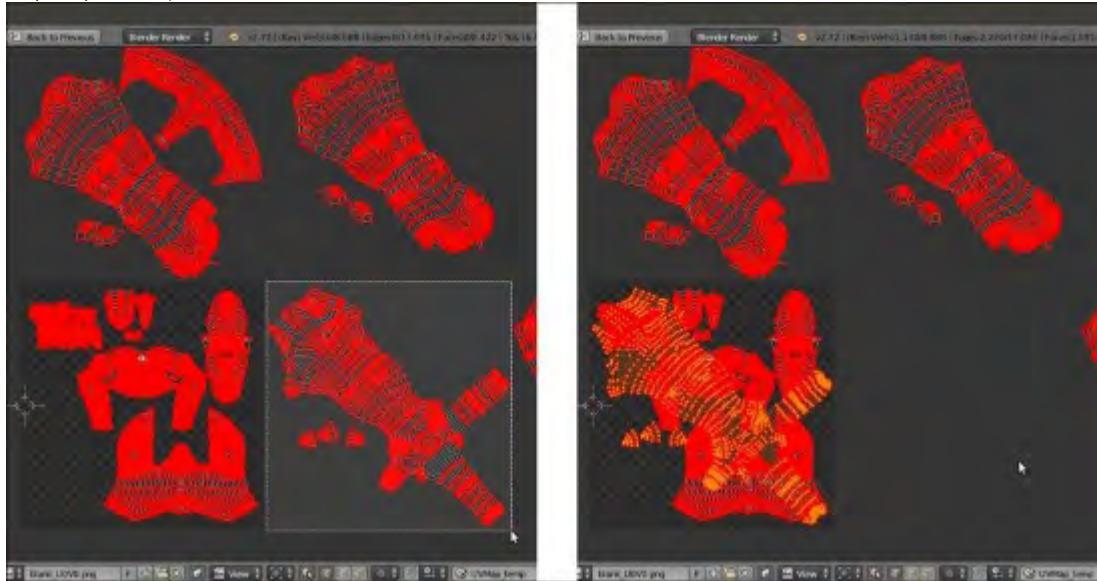
- In the pop-up menu, accessible by clicking on the UVs item in the image editor toolbar, the **Constrain to Image Bounds** item must be deselected:



The "Keep UV and edit mode mesh selection in sync" button and the Constrain to Image Bounds item

Go to the **blank_U0V0** image editor window; if you prefer, maximize it (mouse cursor into the window and press *Ctrl* + Up Arrow). If necessary, press *A* to deselect all the islands.

- Now, box-select the islands on the **U1V0** tile, and move them to the default **U0V0** tile space (*G* | *X* | **-1** | *Enter*):



The UV islands of the **U1V0** tile space, box-selected and moved to the default **U0V0** tile space

2. Deselect everything and box-select the islands at **U2V0**, then move them to the default space, which is the same as the previous one (**G | X | -2 | Enter**).
3. Repeat for the last **2** islands tiles (**G | Y | -1 | Enter**) and (**G | X | -1 | Enter** and then **G | Y | -1 | Enter**), then rearrange the image editor windows.
4. Go to the **Object Data** window and in the **UV Maps** subpanel, be sure to have the **UVMap_temp** layer, the **last one**, enabled as the active one, that is, the **camera icon** to the right side of the **UVMap_temp** item must be the one enabled and visible (*Set the map active for rendering*):



The new UVMap_temp coordinates layer

5. Out of **Edit Mode**, go to the **Render** window and then go to the **Bake** subpanel (usually at the bottom of the panel). If necessary, click on the **Bake Mode** slot to select **Textures**, then set the **Margin** value to **8** or higher; and check the **Clear** item flag. Be sure to have the **Gidiosaurus** object still selected and press the **Bake** button.

After a while, the baked scales textures appear on the **5** PNG images, baked according to the UV islands of the **5** tiles of the **UVMap** layer:



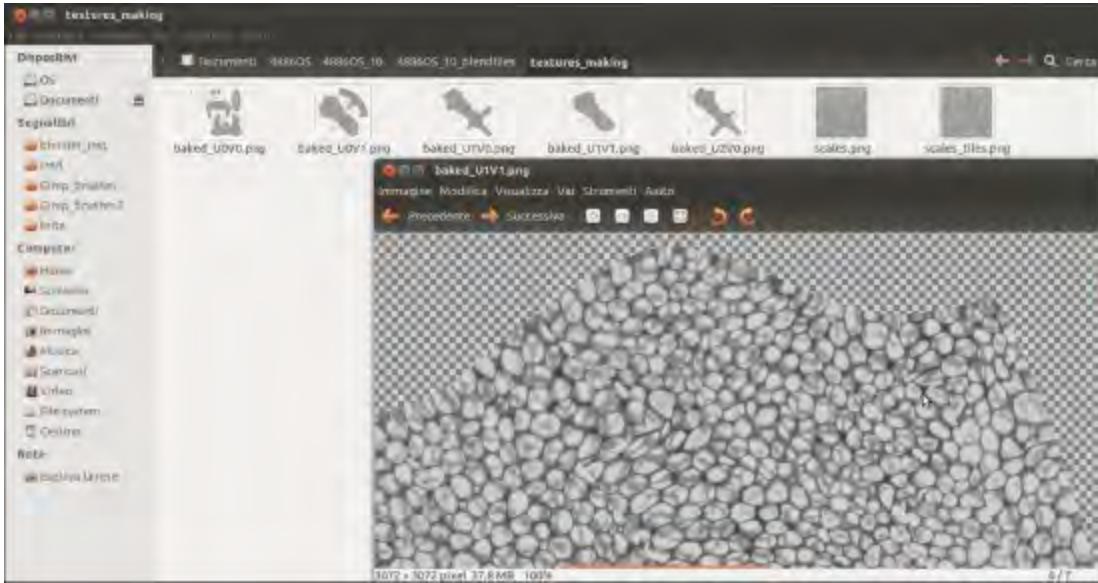
The 5 baked images and the Bake subpanel under the Render window

6. Click on the **Image** item on the **UV/Image Editor** toolbar and from the pop-up menu, select the **Save All Images** item or if you want to preserve your blank images (we are going to use them again later), just save each image at a time (**Save As Image** item or **F3** key) with the names `baked_U0V0.png`, `baked_U1V0.png`, and so on:



Saving the baked image maps

Opening the `texture_making` folder on your desktop, you will now find the baked textures:



The baked textures saved inside the "texture_making" folder

As you can see in the information bar at the bottom of the **GNOME** image viewer (I'm working in **Linux Ubuntu**), each image saved from Blender is **37.8** megabytes.

The large size of the images can of course be reduced (a lot) by opening them in **Gimp** (or any other 2D application) and re-saving.

How it works...

All the UV islands have been moved to the default **U0V0** tile space, which is the only one where the baking happens, but because each image is associated with a different part of the mesh, each image is correctly baked with the right islands and textures.

In fact, inside Blender, and in our case, the location of each tile in the UV space doesn't actually matter; we made a new UV coordinates layer and kept the old one just in case the model should be exported to a different 3D application.

To move the islands exactly by the correct number of pixels, we enabled the **Normalized** item in the **Display** subpanel of the image editor *N* sidepanel to display the UV coordinates from **0.0** to **1.0**, rather than in pixels. Anyway, without the **Normalized** item enabled, it would have been enough to move the islands by **3072** pixels, that is, the width (or/and height) in pixels of the assigned blank image.

There's more...

As with any other software, Blender is not free from bugs; particularly, the baking section seems to have an annoying bug, which is very difficult to fix because it happens very rarely and randomly, so that it

cannot easily be reproduced and consequently submitted to the Blender bug tracker (<https://developer.blender.org/maniphest/project/2/type/Bug/>).

It's difficult to understand the reason for this, but sometimes the software refuses to do the baking, claiming that *No objects or images (are) found to bake to* (the message appears on the top right main header and in the **Terminal** panel as well); in our case, this seems to happen when you switch the *active for rendering* UV coordinates layers.

If this happens, one thing you can do is check that all the images assigned in the **UV/Image Editor** windows to the different materials under one UV layer, also appear correctly assigned under the other UV layer (it shouldn't make a difference, but who knows), eventually re-assigning them one at a time.

If the baking still fails, there is a simple workaround; switch to the **UVMap** coordinates layer instead, rather than the **UVMap_temp** one, and just move the islands to the default **U0V0** space and bake them *one at a time*. To do this, first bake the islands of the **U0V0** tile space and save the image, then move the islands of the **U1V0** tile space to the **U0V0** tile space, bake and save as a different image, and so on with the islands of all the tiles.

Painting to fix the seams and to modify the baked scales image maps

In the previous recipe, we baked the *randomly* tiled scales image map on the **5** tiles of the **UVMap** coordinates layer. This was necessary for the next step to be able to fix seams and modify certain areas of the baked scales images through the **Paint Tool**.

In order to paint in real time on both the model and on all the images assigned to the **5** different UV tiles, and at the same time, once again we need to first prepare the file. To be more precise, we must assign **5** different materials to the mesh, one for each tile and each one with the appropriate image texture.

Getting ready

Start Blender and re-open the `Gidiosaurus_baking_scales_01.blend` file; save the file as `Gidiosaurus_baking_scales_02.blend`.

1. Minimize the image editor windows on the left as much as possible, then also minimize the **Outliner**, the **Material**, and the **Texture** windows on the right to make room for the 3D viewport.
2. Click on the **Viewport Shading** button in the 3D viewport toolbar and switch the shading mode from **Material** to **Solid**, then press the *T* key to call the **Tool Shelf**. Then switch from **Object Mode** to **Texture Paint** mode by clicking on the mode button in the toolbar:



Switching to Texture Paint mode

3. Click on the **Options** tab inside the **Tool Shelf** and under the **Project Paint** subpanel, enable the **Occlude**, **Null**, and **Normal** items:



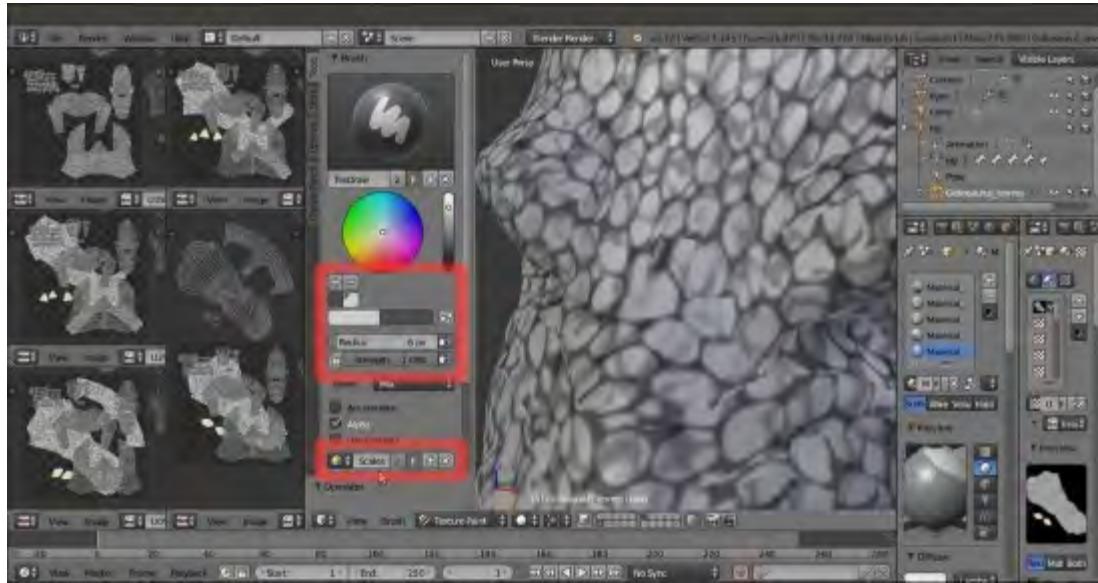
Items to be enabled under the Options tab

4. Click on the **Tools** tab inside the **Tool Shelf** to go back to the **Brush** subpanel options.

How to do it...

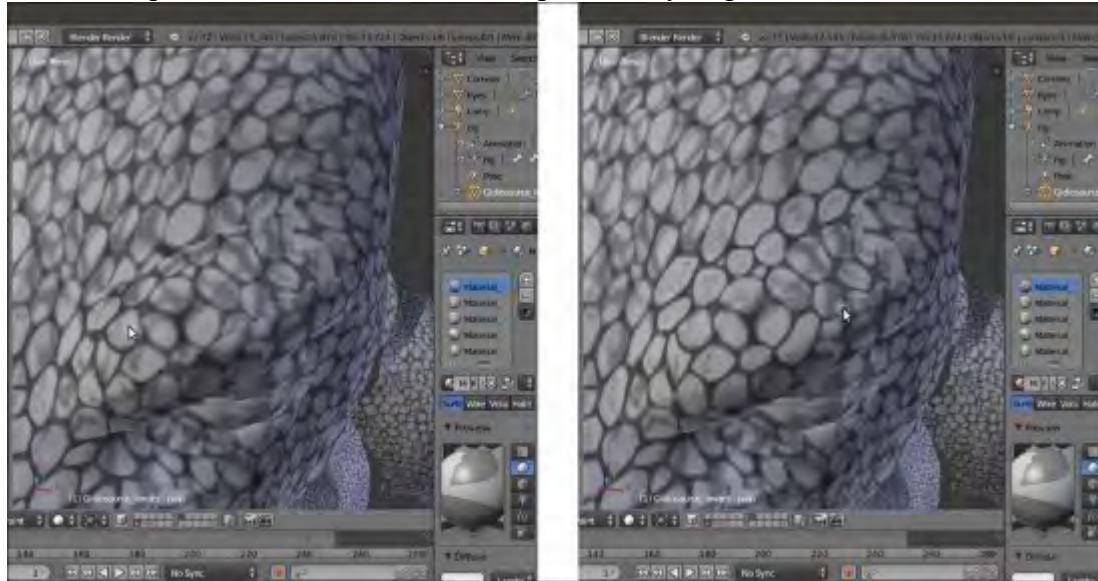
At this point, we are ready to start to paint both directly on the model in the 3D viewport or also in the **UV/Image Editor** windows (just for all eventualities, I suggest you make a copy of the baked scales images before starting to paint):

1. Zoom in on a part of the **Gidiosaurus** object in the 3D viewport, for example, the **head**.
2. Put the **mouse cursor** on a bright value of the scales image on the model and press the **S** key to sample it, then go near the color selector in the **Brush** subpanel under the **Tool Shelf** to the left and click on the *Toggle foreground and background brush colors* button (the one with the two opposing arrows) to switch the active color. Otherwise, simply press the **X** key, put the mouse cursor on a dark area, and press **S** again to sample it as the opposite color.
3. Scroll down and click on the **New** button (*Add new palette*) at the bottom of the **Brush** subpanel; + and – icon buttons will have appeared above the color switcher. Click on the + icon button to add the active color to the palette, then switch the colors and click on the + button again to add a new color to the palette.
4. Set the brush's **Radius** value to **6** and **Strength** value to **1.000**, and if you are using a **graphic tablet**, be sure to have the **2 tablet pressure sensitivity** buttons at the sides of the previous items enabled. Change the default **Palette** name in **Scales**.



Setting a palette and the brush strength and radius

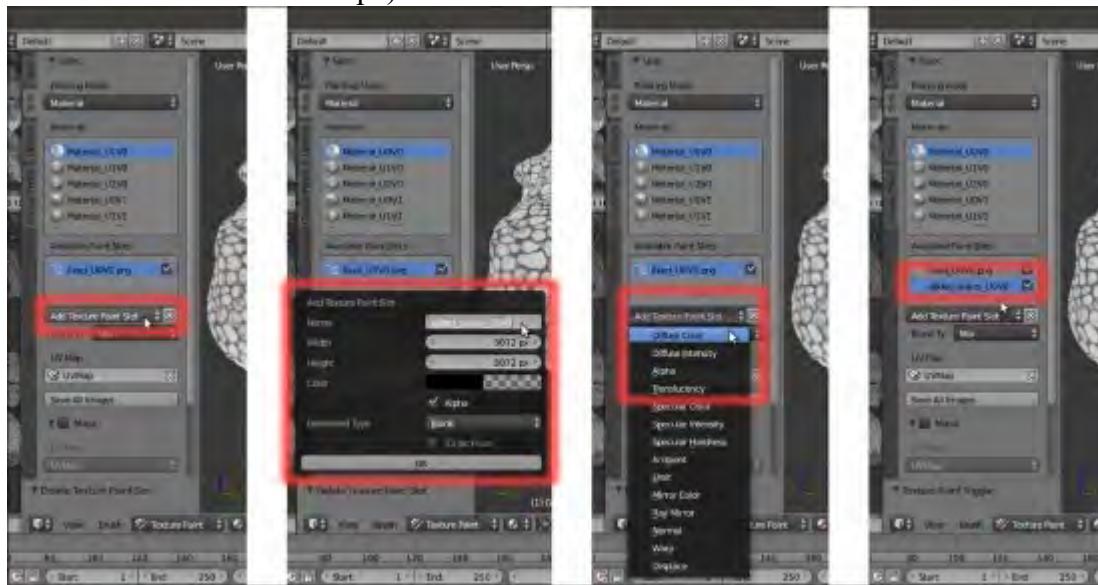
- Simply start to paint on the model, re-drawing the scales where there are seams by flipping the color as you need to, by pressing the X key and painting the dark folds and the light scales. The two colors we sampled, used with the pressure sensitivity enabled, should be enough, but feel free to sample new ones and add it to the palette as you go on:



Painting on the model to fix the image texture seams

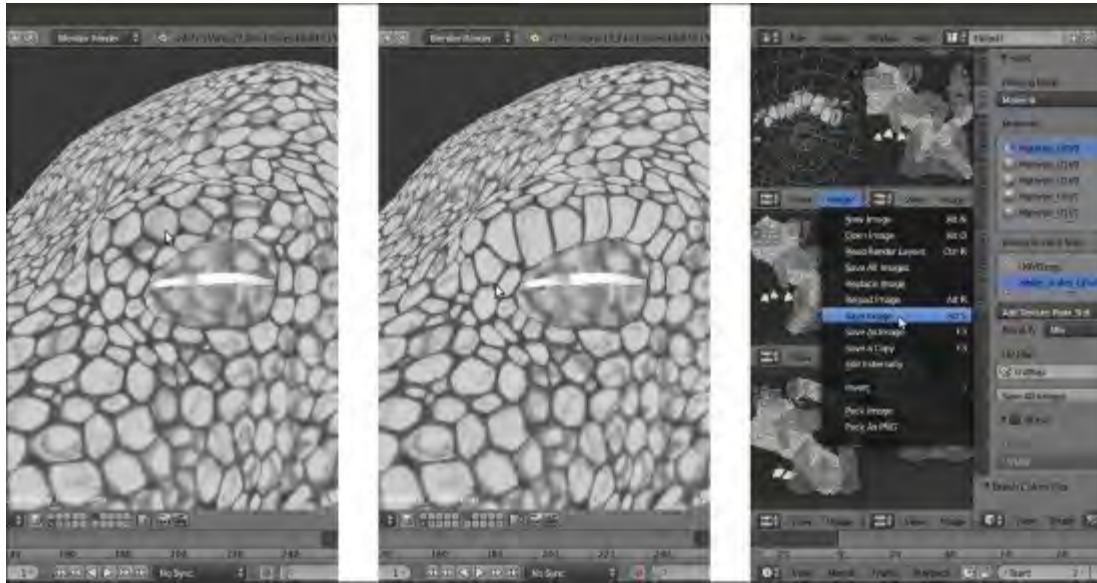
- From time to time, click on the **Slots** tab in the **Tool Shelf** and click on the **Save All Images** button.

7. Do most of the fixing you can, across the entire **Gidiosaurus** body, keeping in mind that it's quite useless to spend time fixing seams in areas that will later be covered by the **Armor** (for example, the top of the head).
8. When you are done, *be sure to have saved all the edited images* as explained in step 6 (but you can also do it one image at a time through the **Image | Save Image** item in each editor window toolbar or by pressing the **Alt + S** shortcut).
9. Now, be sure to have the **Material_U0V0** slot selected as active in the **Material** window and go to the **Texture** window; left-click on the empty slot right under the **U0V0** one and then click on the **New** button to add a new image texture.
10. Scroll down to the **Image** subpanel and click on the **New** button. In the **New Image** pop-up panel, write **added_scales_U0V0** in the **Name** slot, then set the **Width** and **Height** values to **3072** and the **Alpha (A)** value to **0.000** (basically add a new blank and background transparent image as shown in step 5 of the *How to do it...* section of the *Preparing the model to use the UDIM UV tiles* recipe):



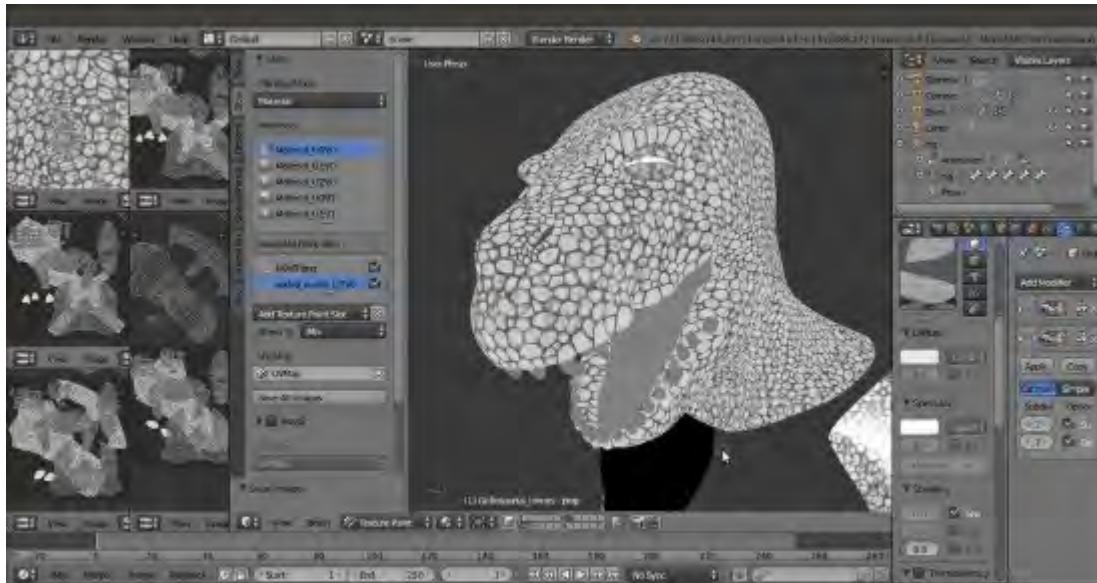
Adding a new texture paint slot layer

11. Go to the **Shading** subpanel of the **Material** window and enable the **Shadeless** item for the **Material_U0V0** slot. Then go to the 3D viewport toolbar and change **Viewport Shading** from **Solid** to **Material**.
12. If not selected already, click on the **Slots** tab in the **Tool Shelf** panel and select the **added_scales_U0V0** item that appears under the **U0V0 .png** once inside the **Available Paint Slots** window.
13. Directly in the 3D view, start to paint new scales on the **eyebrows** to replace the randomly distributed ones; use the light color of the palette to conceal the old scales on the first layer, and the dark color to draw the new ones. Try to build a consistent pattern, also using photos of real reptiles as references. When you are done, save the image in the **texture_making** folder:



Painting new scales on the eyebrow

14. Draw new scales around the **nostrils** and the **rim of the mouth**:



Painting new scales also around the nostrils and at the rim of the mouth

The Blender **Paint Tool** also has other handy brushes; a particularly useful one is the **Smear** brush, which smudges the borders or any blotch in the scales.

To access the brushes, just click on the big window in the **Brush** subpanel under the **Tool Shelf** and click on the chosen one to select it:



The brushes selection pop-up menu

Remember to always save the painted images before closing Blender, otherwise you'll lose them.

Also remember that if you have more than one texture layer to save, it's necessary to load each one of them into an **UV/Image Editor** window. This is actually very quick and easy, just select each layer in the **Available Paint Slots** window (in the **Slots** subpanel under the **Slots** tab) to make it appear in the image editor window and save it through the **Image | Save As Image** menu in the editor toolbar.

Once saved the first time, it's possible to re-save all of them in one single click, through the **Save All Images** items, both in the tab, as well as in the toolbar menu.

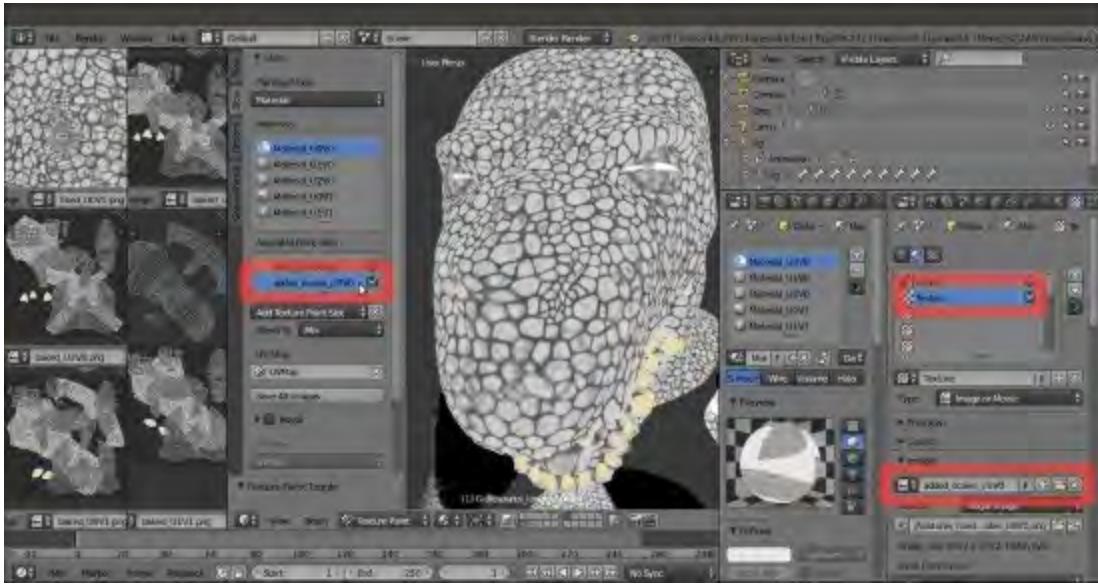
15. Save the file as `Gidiosaurus_painting_scales_fix.blend`.

Note

In the textures and blend files provided with this cookbook, you'll find textures fixed only in the **head** area; I leave the task of finishing the fixing and drawing of new scales on the rest of the body (for example, bigger scales can be added to the upper side of the hand **fingers**, **feet**, **shoulders**, and so on) to you.

How it works...

The new Blender 2.73 texture paint layering feature works simply by adding a new texture slot to the material, and automatically setting it as required, by the type of texture you selected in the **Add Texture Paint Slot**; in fact, by going to the **Texture** window, it is possible to see the added new texture slot and also, if necessary, to change the settings:



The added texture paint slot also appearing as a texture slot in the Texture window

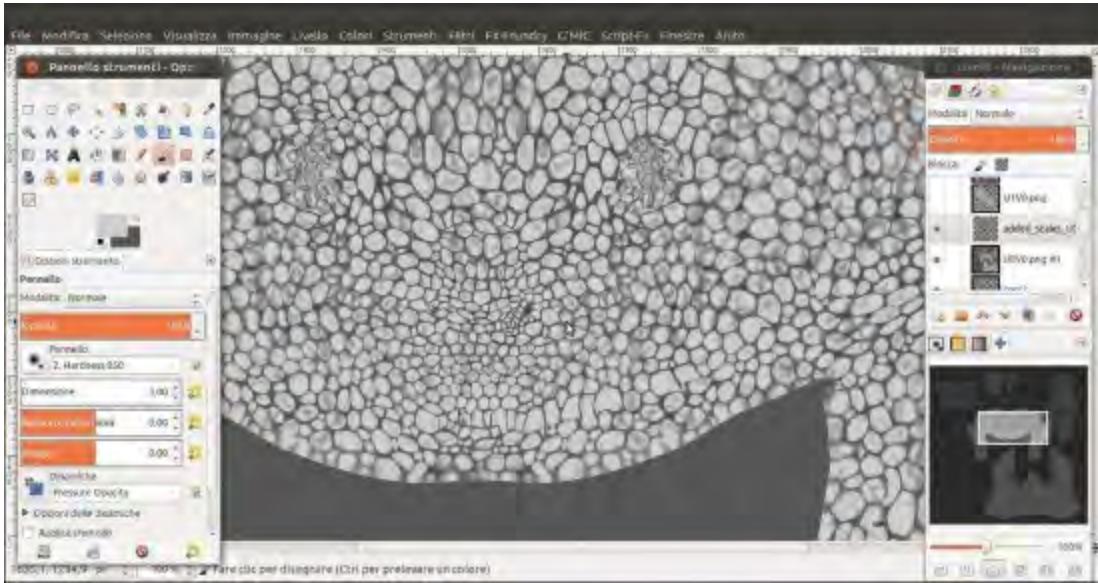
Nonetheless, it is a great addition to Blender that can simplify the texture painting workflow a lot.

There's more...

To bake the added scales as a single image with the background scales images, perform the following steps:

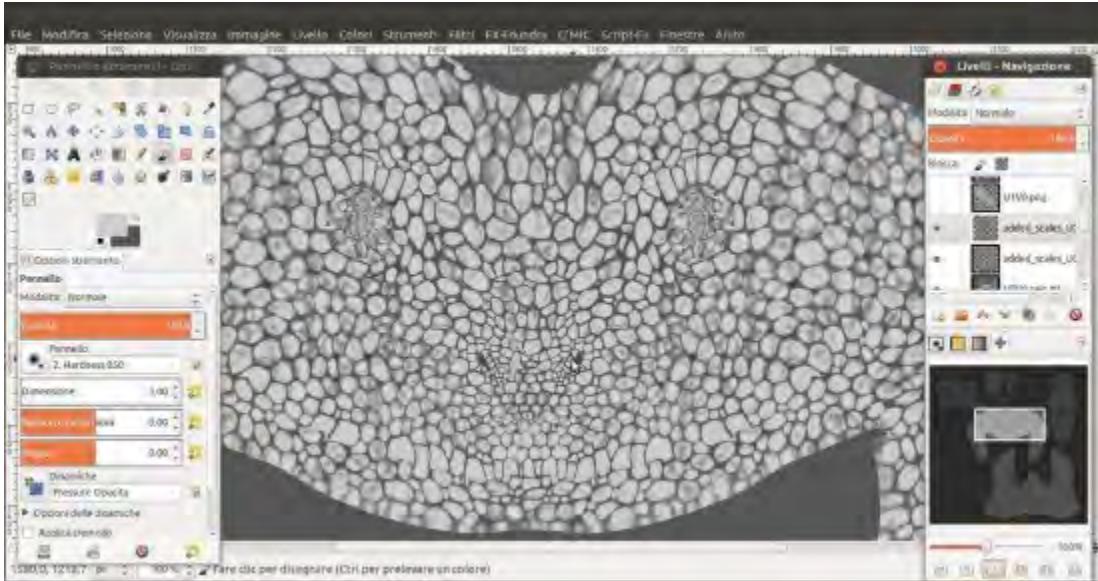
1. Enter **Edit Mode** and click on the **Select** button for **Material_U0V0** to select the vertices assigned to that tile.
2. Go to the top left **UV/Image Editor** window and press **Alt + N** to call the **New Image** pop-up menu; add a new blank image of **3072 x 3072** pixels named **baked_scales_U0V0**, and save it inside the **textures_making** folder.
3. Go out of **Edit Mode** and if it's the case, click on the double arrows icon to the left side of the image name datablock to re-assign the just-created **Untitled** image.
4. Repeat for the other four materials, naming the new images according to the tile and saving them inside the **texture_making** folder as well.
5. Go to the **UV Maps** subpanel under the **Object Data** window to make the **UVMap_temp** coordinates layer active.
6. Go to the **Render** window, and be sure that the **Bake Mode** under the **Bake** subpanel is set to **Textures**, then click on the **Bake** button.
7. After the baking is done, click on the **Image** item in the toolbar of one of the image editors and select the **Save All Images** item.

Not necessarily everything has to be fixed by painting in Blender; for example, it would be enough to fix the scales on only the half of the **head**, export the painted image texture, and open it in **Gimp** (or any other 2D image editing software):



The scales "U0V0.png" image map and the "added_scales_U0V0.png" layer in Gimp

Then, by duplicating the layer and mirroring it, plus a little bit of painting to adjust the seams, it's really simple to obtain the missing half of the new scales texture:



The duplicated and mirrored "added_scales_U0V0.png" layer in Gimp

Of course, if you want to fix every side and part by hand-painting on the model in Blender to obtain a more natural looking result, no one is going to stop you!

See also

- <http://wiki.blender.org/index.php/Doc:2.6/Manual/Textures/Painting>
- http://wiki.blender.org/index.php/Dev:Ref/Release_Notes/2.72/Painting
- <http://docs.gimp.org/en/gimp-tutorial-quickie-flip.html>

Painting the color maps in Blender Internal

After having obtained the scales textures, we must now paint the diffuse color of the **Gidiosaurus** character.

Getting ready

Start Blender and open the `Gidiosaurus_baking_scales.blend` file:

1. Go to the main **Properties** panel and be sure to have the **UVMap** coordinates layer selected and active, in the **UV Maps** subpanel under the **Object Data** window.
2. Go to the **Material** window and select the **Material_U0V0** slot, then go to the **Texture** window and be sure to have the **scales_tiles** texture slot selected; left-click on the **X** icon button to the right side of the name datablock to unlink it (*Shift + left-click* to remove it from the file):



Unlinking the scales_tiles texture slot datablock

3. Select and enable (by clicking on the checkbox to the right) the **U0V0** texture slot. Repeat the procedure at steps 2 and 3 for all 5 materials.

I'll take for granted that you have preserved your blank images and that they are the ones loaded into the current file; otherwise, substitute them with new blank images (you have to do this both in the **Texture** window as well as in the **UV/Image Editor** windows) by following steps 5 and 7 of the *Preparing the model to use the UDIM UV tiles* recipe in this chapter.

4. Minimize the image editor windows to the left and the **Material** and **Texture** panels to the right as much as possible, then click on the mode button (*Sets the object interaction mode*) on the toolbar to go into **Texture Paint** mode. Press **T** with the mouse pointer over the 3D view to call the **Tool Shelf** and click on the **Viewport Shading** button on the toolbar to switch to **Solid** mode:



Switching to Solid viewport shading mode

- Just to verify that everything works correctly, select a black color (or any other one) in the color wheel under the **Brush** subpanel and trace a continuous stroke in the 3D viewport that envelopes all the **Gidiosaurus** body parts:



Testing that everything works correctly with a single stroke on the mesh

By enlarging the **UV/Image Editor** windows, you will see that after the stroke, each image has been updated with the corresponding painting (pay no attention to the over-imposed and repeated for each window UV islands):



The test stroke correctly visible inside each one of the UV/Image Editor windows

6. Rearrange the image editor windows, then press **Ctrl + Z** to undo the stroke and save the file as **Gidiosaurus_painting_BI.blend**.

How to do it...

We are now ready to paint the basic color for the **Gidiosaurus** character. But first, one more little thing:

1. Select the **Gidiosaurus** object and enter **Edit Mode**; select the vertices of all the **teeth** and all the **talons**, then assign a new vertex group renamed **enamel**; press **Ctrl + I** to invert the selection and go out of **Edit Mode**.
2. Now, start by selecting a medium dark greenish color (**R 0.349, G 0.510, B 0.435**) in the color wheel under the **Brush** subpanel. Scroll down and go to the bottom of the subpanel and click on the **New** button to create a new palette, then click on the **+** icon button (**Add Swatch**) above the **Foreground Color** slot (the left one) to add the color to the palette. Rename the default **Palette** name as **Gidiosaurus_colors**.
3. In the 3D viewport toolbar, click on the *Face selection masking for painting* button to enable the masking tool; now it's possible to paint only on the part of the mesh that has selected vertices in **Edit Mode**, so in this case we want to paint only on the skin, leaving the teeth and the talons blank.
4. Click on the **Brush** window and select the **Fill** brush; if necessary, click on the greenish color box added to the palette (called **Swatch**) to load it as the **foreground color** (note that with the **Fill** brush, the background color swatch disappears and the foreground color swatch becomes the only one available). Set the **Strength** value to **1.000** and click on the **Gidiosaurus** object in the 3D viewport.

After a while, all the *paintable* parts of the mesh are filled with the active color (and therefore also the textures in the **UV/Image Editor** windows; there are weird straight lines, probably a

bug, but not a problem in this case because they don't show on the mesh and we can fill in the texture's backgrounds later anyway). If any tiny part is left out, just click on one of the parts again to fill it:



The Fill brush and the Mask button in the 3D view toolbar

5. Now select the **TexDraw** brush and a darker and more saturated green color (**R 0.129, G 0.275, B 0.125**) as the foreground color, and add it to the palette.
6. Under the **Tool Shelf**, go to the **Options** tab and be sure to have the **Occlude**, **Cull** and **Normal** items disabled still; then go into the **Ortho Side** view.
7. Now it's time to use a tablet, if you have one; enable both the tablet pressure sensitivity buttons to the side of the **Radius** and **Strength** items and start to shade the **Gidiosaurus** body on the **head, shoulders, arms, and legs**:



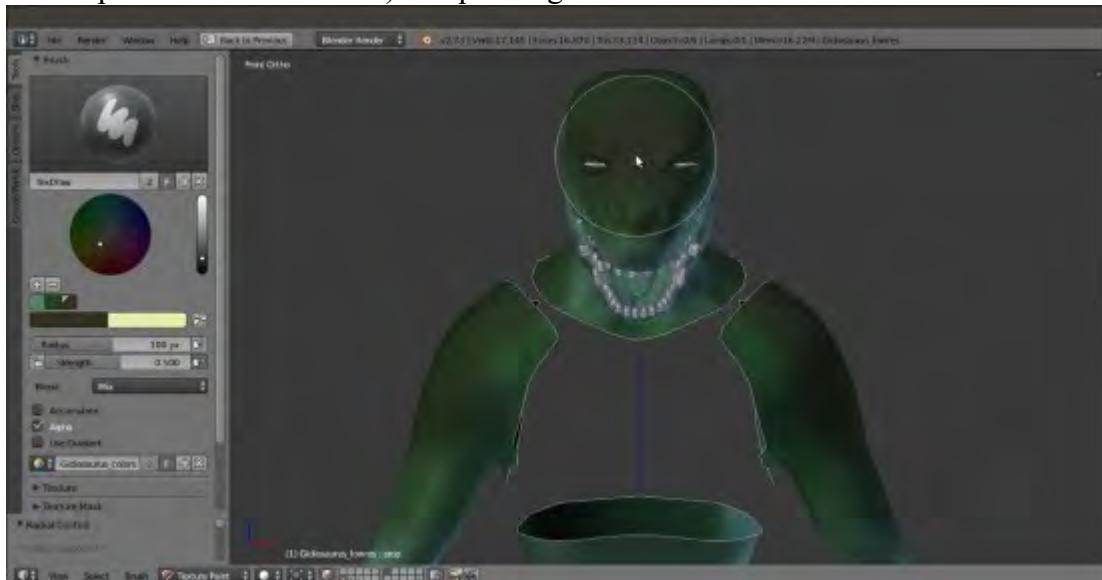
Painting colors on the model

8. Select a brownish color (**R 0.204, G 0.188, B 0.133**) and add it to the palette; disable the tablet pressure sensitivity for **Radius** and lower the **Strength** to **0.500**. Go into the **Front** view, maximize the 3D viewport (mouse pointer in the window and press *Ctrl + Up Arrow*), and keep on adding shades to the **hands, feet, and legs**:



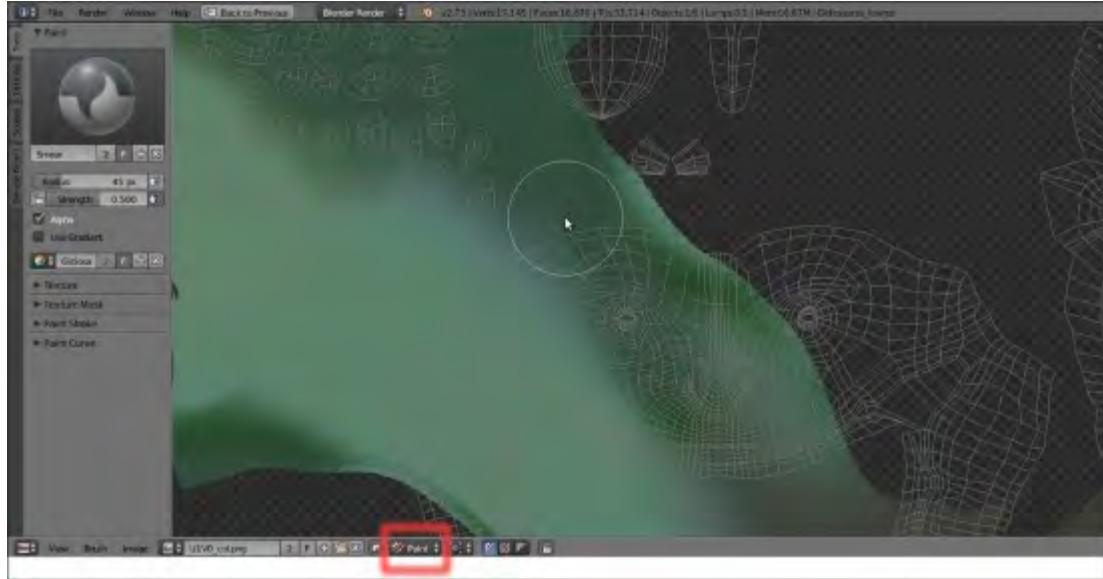
Shading the character's limbs with darker hues

9. Increase the **Radius** value to **100** (using the slider or by pressing the *F* key and moving the mouse pointer in the 3D view) and painting on the **head** and the **shoulders**:



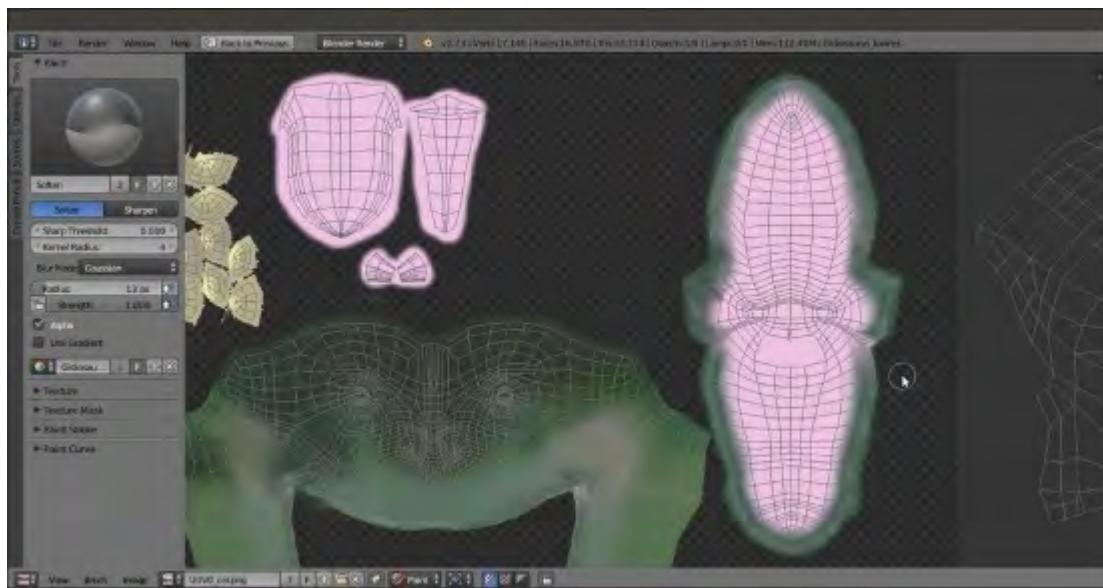
Shading the head and shoulders

10. If, for any reason, it becomes difficult to paint directly on the model through the 3D viewport, you can maximize the involved **UV/Image Editor** window (mouse pointer in the window and press **Ctrl + Up Arrow**), click on the **Mode** button (*Editing context being displayed*) in the toolbar (which by default shows **View**), and switch it to **Paint**. Press **T** to call the **Tool Shelf** and go on with the painting, smudging, or whatever, directly on the texture image:



Painting directly on the image map in the UV/Image Editor window

For example, this is the way I painted the inside of the **mouth** and the **tongue**, then went back to the 3D viewport to smudge and soften the joining line of the pink tissue with the green skin at the borders:



Working on the inside of the mouth in the UV/Image Editor window

I'm not going to show you every step in this process, but basically this is the procedure I used to paint the diffuse coloration for the character. I also added lighter and warmer colors for the face's areas close to the **mouth** and more bluish and colder hues to de-saturate the brownish **hands** and **feet**, and then inverting the **enamel** vertex group to paint in **Edit Mode**, through the use of the **Mask** tool, the **teeth** and **talons** as well:



The completed Gidiosaurus diffuse color texturing

To have a look at the final **Gidiosaurus_colors** palette, open the **Gidiosaurus_painting_BI_02.blend** file provided.

11. When you are done, go to the top left **UV/Image Editor** window, **blank_U0V0**, and click on the **Image** item in the toolbar. Save the image texture in the **textures_making** folder as **U0V0_col.png**, and do the same with the other **4** image textures.
12. To keep the palette, save the file.

How it works...

There is not that much to explain about this recipe, except I just want to highlight the fact that we disabled the **Occlude**, **Cull**, and **Normal** items in the **Options** tab under the **Tool Shelf**. This is so we were able to paint (from the **Side** view) on both sides of the model at the same time; in fact, with these settings disabled, the mesh is *not occluding itself*. It seems that all three items must be disabled for this to work.

Instead, to smear and/or soften the texture on some parts, for example, the inside of the **mouth**, we had to re-enable them, in order to prevent our mouth-painting from accidentally overwriting our skin-painting.

Remember, the **Occlude**, **Cull**, and **Normal** items should always be enabled if you want to paint only on the model's surface right under your brush. You can disable them to paint on the front/outer and the back/inside of the mesh at the same time.

See also

- <http://www.cgmasters.net/free-tutorials/layered-painting-in-blender-2-72/>
- <http://blender.stackexchange.com/tags/texture-painting>

Painting the color maps in Cycles

There are no differences in painting in **Blender Internal** or in **Cycles**, because the **Paint Tool** is exactly the same; the only difference is in the preparation of the materials.

In this recipe, we are not going to repeat the procedure already explained in the previous one; we'll just set up the file for the painting and test whether it's possible to paint in real time on all **5** image textures at the same time, as it is in **Blender Internal** (*spoiler*: it is).

Getting ready

Let's start with the `Gidiosaurus_painting_BI.blend` file; in that file, we already have the **UV/ Image Editor** windows set and the **5** materials assigned to the **5** different **UDIM** tiles and parts of the mesh.

In case you want to start with a brand new file, here you need to repeat the steps of the *Preparing the model to use the UDIM UV tiles* recipe in this chapter. Then, continue with the following:

1. Be sure you're in **Object Mode**.
2. Go to the main top header and click on the *Engine to use for rendering* button; switch from **Blender Render** to **Cycles Render**.
3. Split the 3D view into two horizontal rows and change the top one into a **Node Editor** window; press the **N** key to get rid of the **Properties** sidepanel.
4. In the **Material** window, select the **Material_U0V0** slot; click on the **Use Nodes** button or select the **Use Nodes** checkbox in the **Node Editor** toolbar:



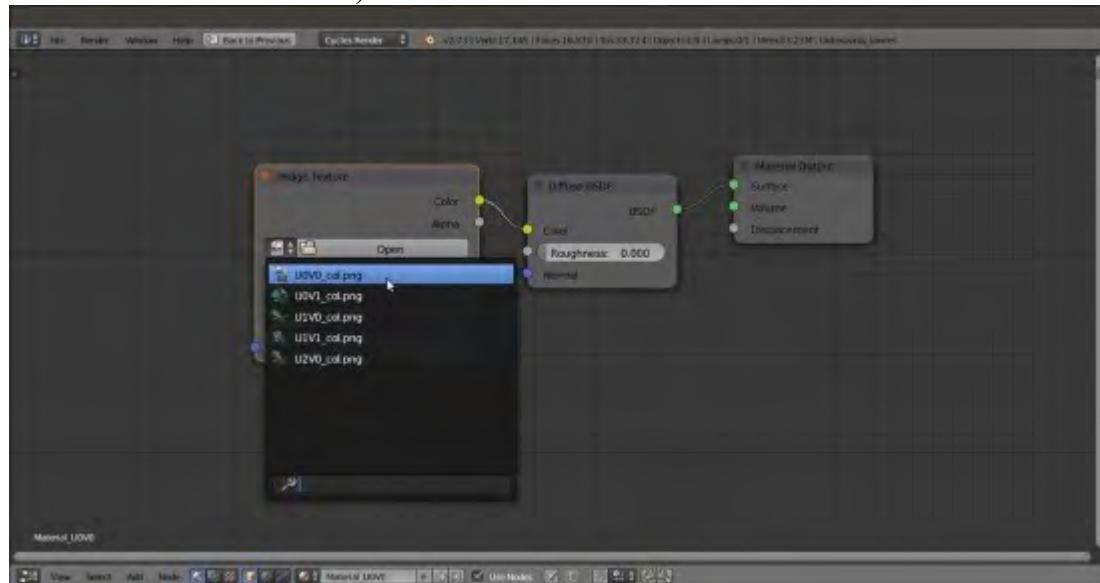
Enabling the nodes for the materials under Cycles

- Put the mouse pointer inside the **Node Editor** window and add an **Image Texture** node (press the *Shift + A* keys and in the pop-up menu, go to the **Texture** item to select **Image Texture**). Connect its **Color** output to the **Color** input socket of the **Diffuse BSDF** node.

At this point, if we haven't already painted the color textures in **Blender Internal**, we should load the `blank_U0V0.png` image in the **Image Texture** node and then do the same for the other **4** materials.

Instead, because we already have the color textures, let's load them in the **Cycles** materials. To see whether everything works as it should, we'll paint on them through the 3D viewport.

- Click on the double arrows to the side of the **Open** button in the **Image Texture** node and select the `U0V0_col.png` item from the pop-up menu (remember that the **5** color textures are already loaded inside the blend file):



Selecting one of the already loaded images in the Image Texture node for the materials under Cycles

- Repeat step 4 to step 6 for the other **4** materials.

How to do it...

Now, the steps are really simple:

- Go to the **Brush** subpanel and switch the foreground color with the background black color.
- Trace in the 3D viewport, a continuous stroke enveloping all the **Gidiosaurus** body parts:



The single stroke test under Cycles

- This is the proof that it works exactly as in **Blender Internal**.
3. Press **Ctrl + Z** to undo the stroke and save the file as **Gidiosaurus_painting_Cycles.blend**.

Chapter 11. Refining the Textures

In this chapter, we will cover the following recipes:

- Sculpting more details on the high resolution mesh
- Baking the normals of the sculpted mesh on the low resolution one
- The Armor textures
- Adding a dirty Vertex Colors layer and baking it to an image texture
- The Quick Edit tool

Introduction

In [Chapter 10, *Creating the Textures*](#), we have prepared the **color** and **bump** texture images for the **Gidiosaurus** skin. In this chapter, we'll see the process for creating some additional (but equally important, nonetheless) textures, both for the character and the iron **Armor**.

Sculpting more details on the high resolution mesh

In [Chapter 2, Sculpting the Character's Base Mesh](#), we sculpted the **Gidiosaurus** character's features, obtaining a high resolution mesh that we re-topologized in the following [Chapter 4, Re-topology of the High Resolution Sculpted Character's Mesh](#), to have a low resolution mesh for easy rigging and texturing.

Because in the following recipe (*Baking the normals of the sculpted mesh on the low resolution one*) we are going to bake the normals of the sculpted mesh on the low resolution one, we should now add as much detailing and finishing to the sculpted model.

I'm not going to explain every step in detail, here, because the procedure is the same as already seen in the [Chapter 2, Sculpting the Character's Base Mesh](#), so just a quick tour to show what I've done should be fine.

Getting ready

Let's start by preparing the file:

1. Start Blender and load the `Gidiosaurus_painting_BI.blend` file; if necessary, go out of **Texture Paint** mode back to **Object Mode** and save the file as `Gidiosaurus_details_sculpt.blend`.
2. Collapse all the **UV/Image Editor** windows on the left of the screen and then join them with the 3D viewport (put the mouse pointer on the edge of one of the two windows; as it changes into a two opposite arrows pointer, right-click and in the **Area Options** pop-up menu, left-select the **Join Areas** item; then, move the mouse pointer towards the window to be eliminated and left-click to join them).
3. Join the **Material** and **Texture** windows in the main **Properties** panel, switch to the **Object Data** window, and enlarge the 3D viewport as much as possible.
4. Click on the **File** item in the main top header and then select the **Append** item (or else, directly press the *Shift + F1* keys); navigate to the `Gidiosaurus_retopology_02.blend` file, click on it, and then click on the **Object** item (folder) to select the **Gidiosaurus** item.
5. Click on the **Append from Library** button on the top-right of the screen and then go to the **Outliner** window to click on the **eye** and the **arrow** icon buttons (*Restrict view-port visibility* and *Restrict view-port selection*) and enable both the object visibility and selection in the 3D viewport.
6. Move the appended high resolution **Gidiosaurus** mesh to the **14th** scene layer (*M* key):



The appended, sculpted Gidiosaurus mesh

7. Press **N** to call the **Properties** sidepanel and in the **Display** subpanel, enable the **Only Render** item; go down to the **Shading** subpanel, enable the **Matcap** item, and then select your favorite matcap type (mine is always the brick red colored **Zbrush-like**).
8. Enable the **12th** scene layer to show the **Eyes**; however, in the **Outliner**, just to be sure, disable the selection arrow icon button.
9. Press **N** again to hide the **Properties** sidepanel and then switch to **Sculpt Mode** and save the file.



The Gidiosaurus object ready for the new sculpting session

How to do it...

We are now ready to sculpt again on the **Gidiosaurus** mesh; first, let's do some more settings pertinent to the sculpt tools:

1. Go to the **Dyntopo** subpanel under the **Tool Shelf** and click on the **Enable Dyntopo** button; set **Detail Size** to **1.60 px** and check the **Smooth Shading** box.
2. Go down to the **Symmetry / Lock** subpanel to be sure that **Mirror** is enabled for the **x** axis.
3. Click on the **Options** tab and go to the **Options** subpanel to enable the **Fast Navigate** item (the **Threaded Sculpt** item should be already enabled by default).
4. Go back to the **Tools** tab and click on the **Brush** windows; select the **Crease** brush (press the **Shift + C** or **5** keys), zoom to the **Gidiosaurus's head**, and start to add **expression folds**:



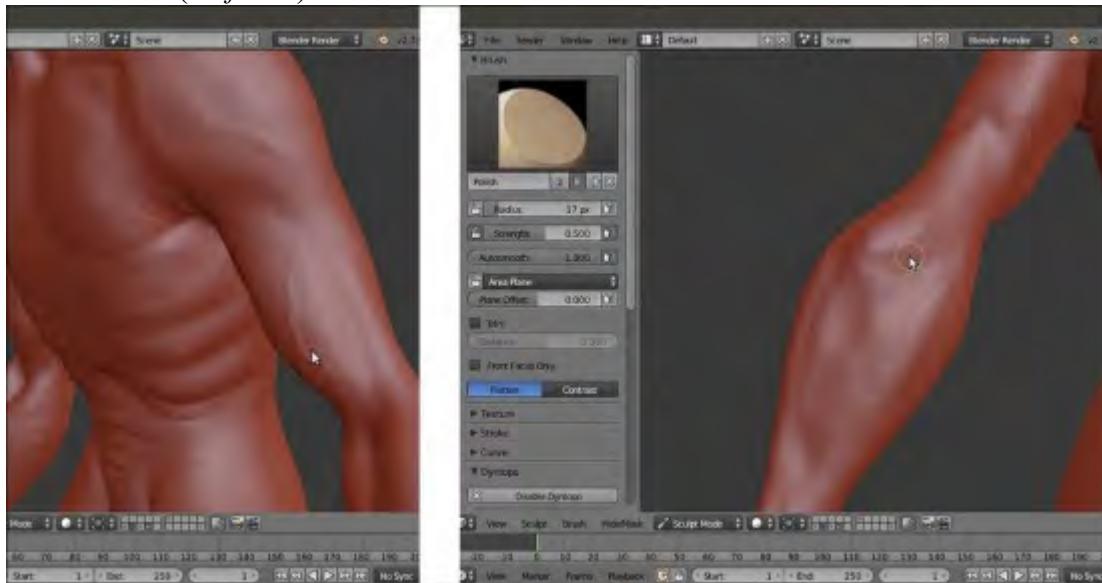
Adding expression folds with the Crease brush

5. Move to the **throat**, in the **Tool Shelf** panel, switch the effect of the brush from **Subtract** to **Add** through the buttons at the bottom of the **Brush** subpanel (or simply by pressing the **Ctrl** key while sculpting), and add veins to the area:



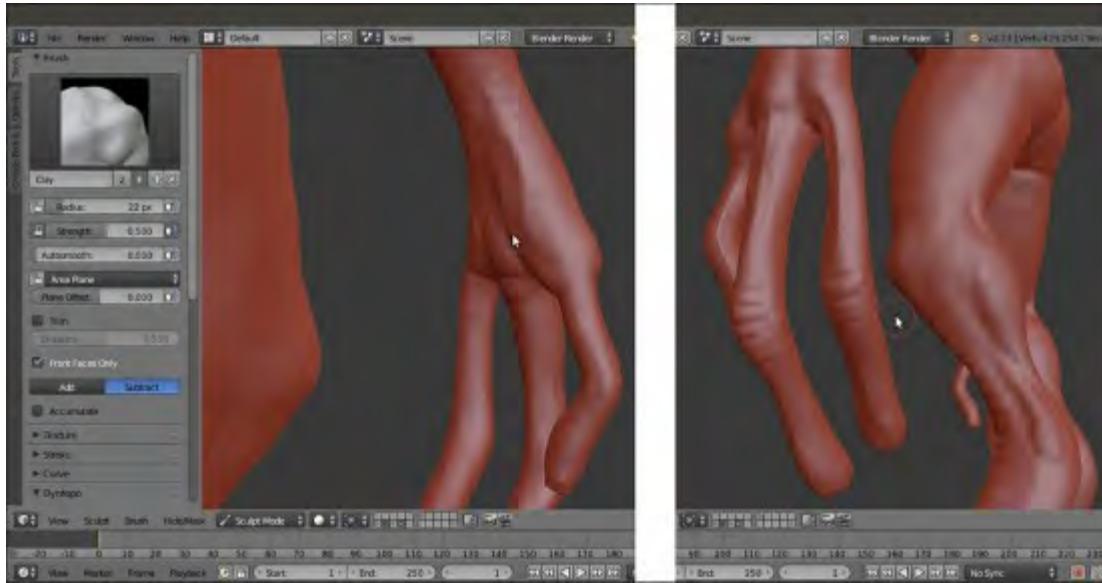
Adding veins under the jaw and on the neck by using the Crease brush again, but with inverted effect

6. By using the same technique, add **veins** also on the **shoulders** and the **biceps**; then, select the **Polish** brush (*Shift + 4*) and refine the **elbow** a bit:



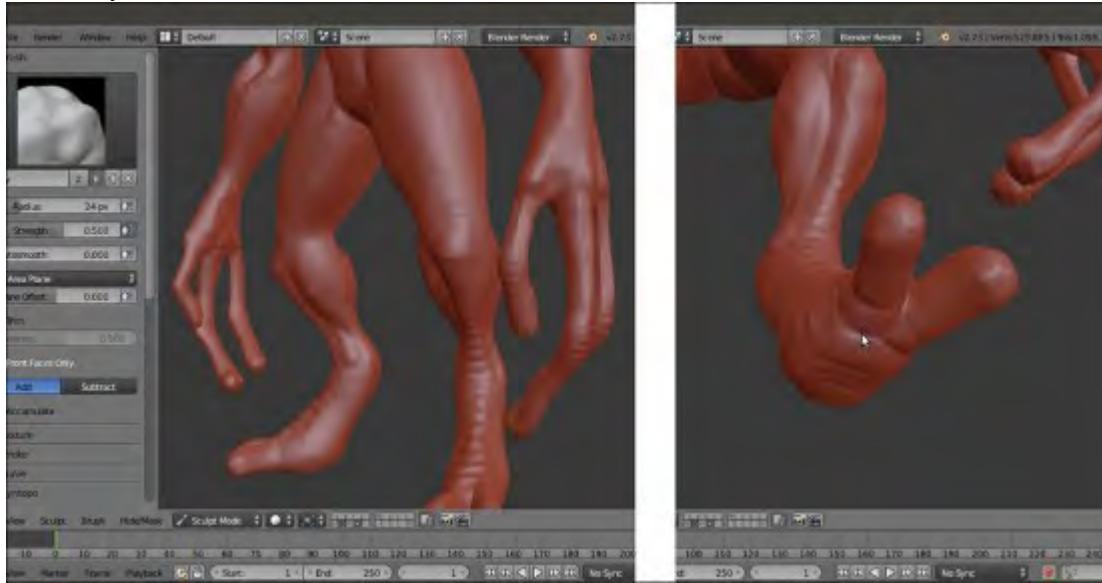
Adding the veins on the arm muscles and polishing the elbow's bulging muscle

7. By using the **Clay** brush (*C* or *3* keys) and also the **Crease** (*Shift + C* or *5* keys) and **Pinch** (*P* or *Shift + 3*) brushes, refine the shape and the folds of the **palm** and add details to the back of the **fingers**. The **Clay** brush can be used in **Subtract** mode too, to carve shapes:



Detailing the palm and the fingers of the hand

8. Similarly, add details and refine the back of the **foot** and the **sole**:



Detailing the feet

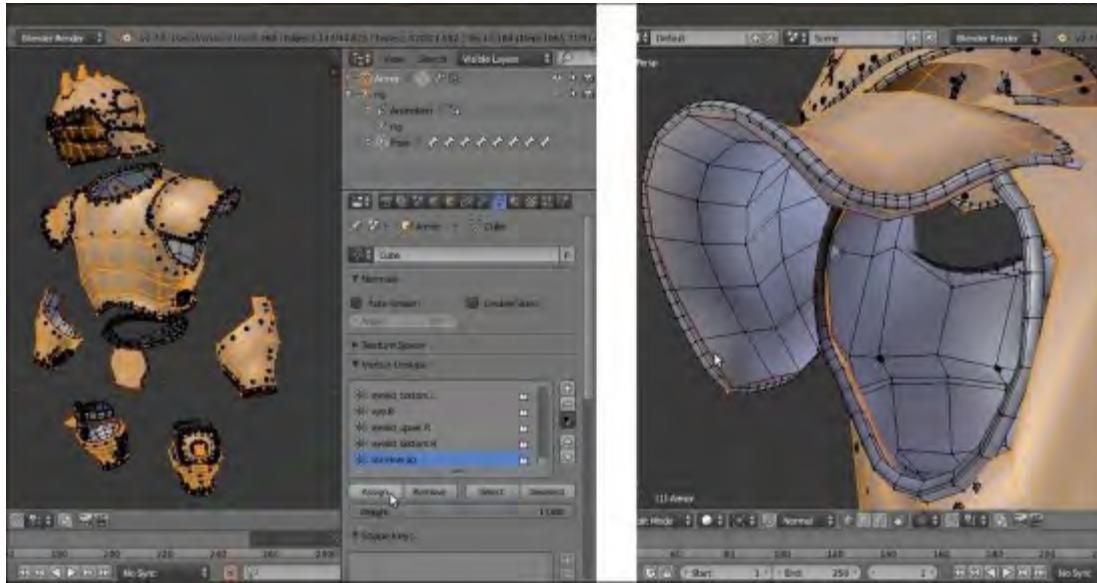
9. Use the **Smooth** brush (*S* or *Shift + 7* keys) to gently soften the character's features; when you are done, save the file.



Smoothing the added features

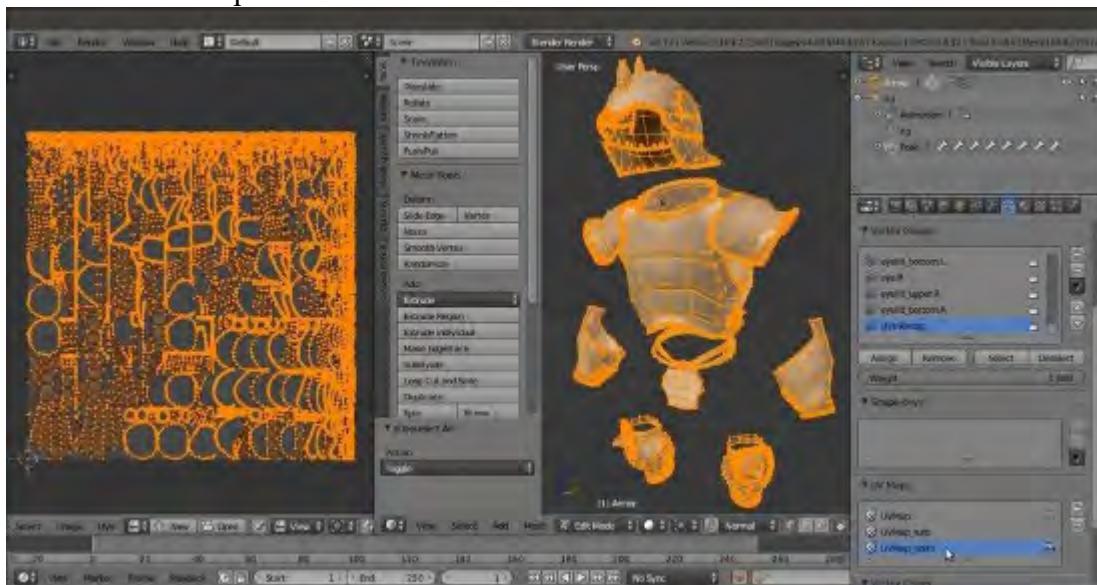
Now, as we have detailed the body of the **Gidiosaurus**, it would be a good idea to refine the **Armor** also.

10. Switch to the **13th** scene layer; select the **Armor** object and go to the **Shape Keys** subpanel under the **Object Data** window.
11. Select the **Basis** shape key and then click on the – icon button to delete it (this leaves the only remaining shape key, **Armor_fix**, as the base one, so permanently applying the morph to the mesh); then, also select the **Armor_fix** shape key and delete it.
12. Repeat the previous steps for the **rivets** and the **Armor_decorations** objects as well.
13. Through the **Outliner** window, *Shift-select* the **rivets**, **Armor_decorations**, and **Armor** objects; then, press **Ctrl + J** to join them as a single object.
14. Go to the **Vertex Groups** subpanel and add a new vertex group; rename it as **shrinkwrap**.
15. Enter **Edit Mode** and select the vertices on the outside of the **armor body plates**, leaving the **inside faces** of the plates, the **bottom** of the **spaulders**, the **decorations**, the **rivets**, and the **tiers**, unselected; if necessary, use the **seams** to help you to divide the outer from the inner parts of the mesh. Click on the **Assign** button at the bottom of the **Vertex Groups** subpanel:



Selecting the outer parts of the Armor

16. Split the 3D view into two windows and change the left one into a **UV/Image Editor** window.
17. Go to the **UV Maps** subpanel under the **Object Data** window, click on the + icon button to add a new UV coordinates layer, and rename it as **UVMap_norm**. Then, click on the camera icon on the right-hand side of the name to make it the active UV layer.
18. Put the mouse pointer in the 3D viewport and press **U**; in the **UV Mapping** pop-up menu, select the **Smart UV Project** item; in the pop-up panel, click on the **Island Margin** value (default = **0.00**) and set it to **0.001**. Leave the other values as they are and click on the big **OK** button at the bottom of the panel.



The UVMap_norm UV coordinates layer for the Armor object

19. Go out of **Edit Mode** and minimize the **UV/Image Editor** window as much as possible; press **Shift + D** to duplicate the **Armor** object and move the duplicated one to the **3rd** scene layer.
20. Enable the **3rd** scene layer; go to the **Object Modifiers** window and delete the **Armature** and the **Subdivision Surface** modifiers; in the **Outliner**, rename the new object (now **Armor.001**) as **Armor_detailing**.
21. Assign a **Multiresolution** modifier. Click on the **Subdivide** button until it reaches level **3**; then, check the **Optimal Display** item and go in **Sculpt Mode**. Using the same procedure as before, add scrapes, bumps, deformations, and so on, to the **armor** surface; add some kind of engraving also, for example, on the **groinguard**.



Sculpting the Armor_detailing object

22. Save the file.

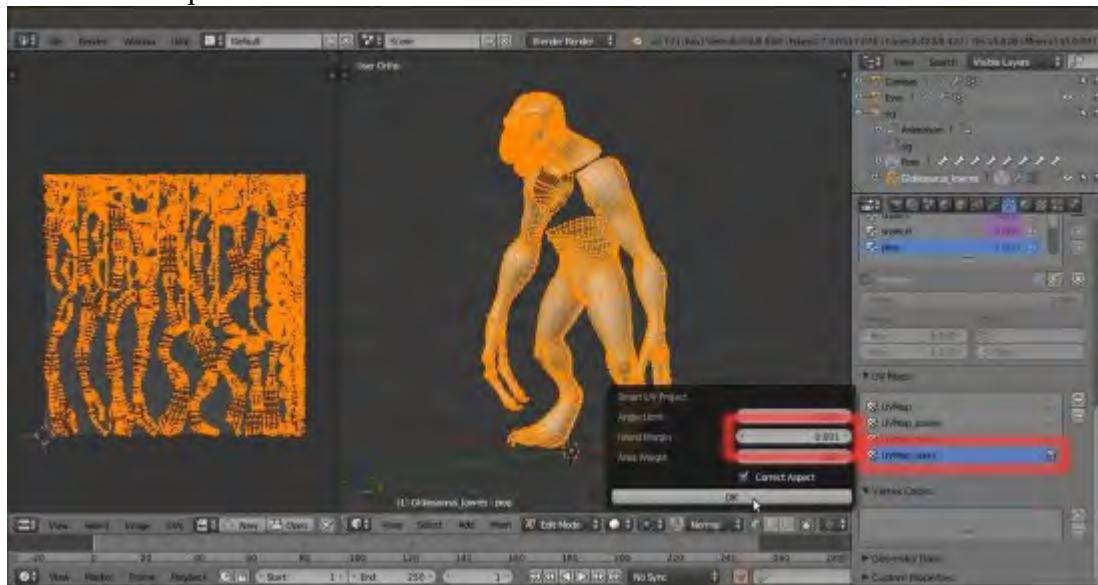
Baking the normals of the sculpted mesh on the low resolution one

At this point, we can transfer all the details sculpted on our high resolution meshes (the **Gidiosaurus** and the **Armor** objects) to the low resolution assets; to do this, we have to **bake** these details as **normal maps**.

Getting ready

Continue from the previous `Gidiosaurus_details_sculpt.blend` file:

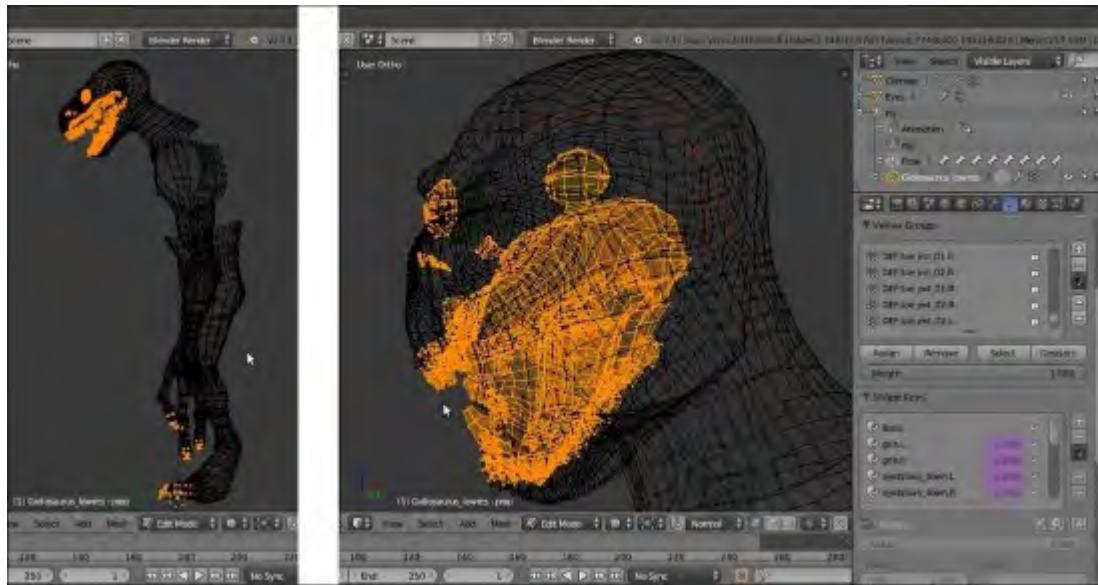
1. Split the 3D viewport into **two** windows and change the left one into a **UV/Image Editor** window.
2. Go to the **11th** scene layer and select the **Gidiosaurus_lowres** object; press the *Tab* key to enter **Edit Mode** and, if necessary, press *A* to select all the vertices.
3. Go to the **UV Maps** subpanel under the **Object Data** window, click on the **+** icon button to add a new UV coordinates layer, rename it as **UVMMap_norm**, and click on the camera icon to make it the active UV layer.
4. Put the mouse pointer in the 3D viewport and press *U*. In the **UV Mapping** pop-up menu, select the **Smart UV Project** item; in the pop-up panel, click on the **Island Margin** value (default = **0.00**) and set it to **0.001**. Leave the other values as they are and click the big **OK** button at the bottom of the panel.



The **UVMMap_norm** UV coordinates layer for the low resolution **Gidiosaurus** mesh

5. Deselect all the vertices (the *A* key again) and zoom to the **head**; **Shift-select** the vertices of all the parts that don't actually exist in the high resolution sculpted model such as the inside of the

mouth, the mouth inner rims, the tongue, the eyelids, the inside of the nostrils, the teeth, and the talons:



Selecting the low resolution mesh parts that don't have a counterpart in the high resolution sculpted mesh

6. Go to the **Vertex Groups** subpanel under the **Object Data** window and click on the + icon button to add a new vertex group; rename it as **shrinkwrap**.
7. Press **Ctrl + I** to invert the selection and then click on the **Assign** button below the vertex group list window in the **Vertex Groups** subpanel:



Assigning the inverted selection to the "shrinkwrap" vertex group

8. Go out of **Edit Mode** and press **Shift + D** to duplicate the **Gidiosaurus_lowres** object; move the duplicate to the **4th** scene layer and in the **Outliner** window, rename it as **Gidiosaurus_for_baking**.
9. In the **Outliner**, enable the 3D viewport visibility of the **rig**; select and move the **ctrl_mouth** bone upward to close the **Gidiosaurus's** mouth and then hide the **11th** scene layer.
10. Reselect the **Gidiosaurus_for_baking** object and go to the **Object Modifiers** window; click on the **Apply as Shape Key** button of the **Armature** modifier.
11. Go to the **Shape Keys** subpanel under the **Object Data** window to find a new shape key at the bottom of the list: **Armature**, with the value of **0.000**.
12. Rename the new shape key as **closed_mouth** and set the value to **1.000**:



The closed_mouth shape key

13. Go to the **Object Modifiers** window and assign a **Shrinkwrap** modifier to the **Gidiosaurus_for_baking** object; as **Target**, select the **Gidiosaurus_detailing** object and then click on the **Vertex Group** slot to select the **shrinkwrap** vertex group.

In the following screenshot, you can see the effect of the **Shrinkwrap** modifier on the low resolution mesh with the **Subdivision Surface** modifier enabled also for the 3D viewport:



The "shrinkwrapped" low resolution Gidiosaurus mesh

How to do it...

After this quite *intensive* file preparation, let's go with the baking itself:

1. Enter **Edit Mode** and select all the mesh vertices; in the **UV/Image Editor** window, add a new **3072 x 3072** blank image and rename it as **norm**.
2. Go out of **Edit Mode**, enable the **14th** scene layer, select the **Gidiosaurus_detailing** object, and then *Shift-select* the **Gidiosaurus_for_baking** object.
3. Go to the **Render** window, scroll the panel down and, in the **Bake** subpanel, check the **Selected to Active** item. Set **Margin** to **8** pixels, the **Bake Mode** to **Normals**, and the **Normal Space** to **Tangent**; click on the **Bake** button to start the baking:



The baked normals' image map, the two overlapping and selected objects, and the Bake subpanel

- Click on the **Image** item in the **UV/Image Editor** window toolbar to save the baked image as `norm.png` inside the `texture_making` folder.

How it works...

To close the **mouth** (to conform it to the sculpted mesh), we moved the control bone in the rig and then applied the **Armature** modifier as a **shape key**; be aware that a modifier cannot be applied to a mesh with shape keys (you get a warning message), so we had to use the **Apply as Shape Key** option or delete all the shape keys with drivers and redo them later. In this case, however, it wouldn't have been necessary to duplicate the **Gidiosaurus** low resolution mesh, but we did it anyway to keep things simpler and cleaner.

Right before the baking, a **Shrinkwrap** modifier has been assigned to the lowres **Gidiosaurus_for_baking** object, to conform its surface to the high resolution sculpted **Gidiosaurus_detailing** object and avoid any possible intersection between the two meshes (that would give ugly artifacts in the baked image); we used the **shrinkwrap** vertex group to keep the vertices that don't have a counterpart on the high resolution mesh (**teeth**, **eyelids**, **inner mouth**, and so on) out of the modifier influence.

As you can see in the following OpenGL screenshot, comparing the sculpted and the low res **Gidiosaurus** meshes, the result of the baked normals on the low resolution object is pretty good and effective:



Comparison between the high resolution sculpted mesh and the low resolution object with the baked normal map

There's more...

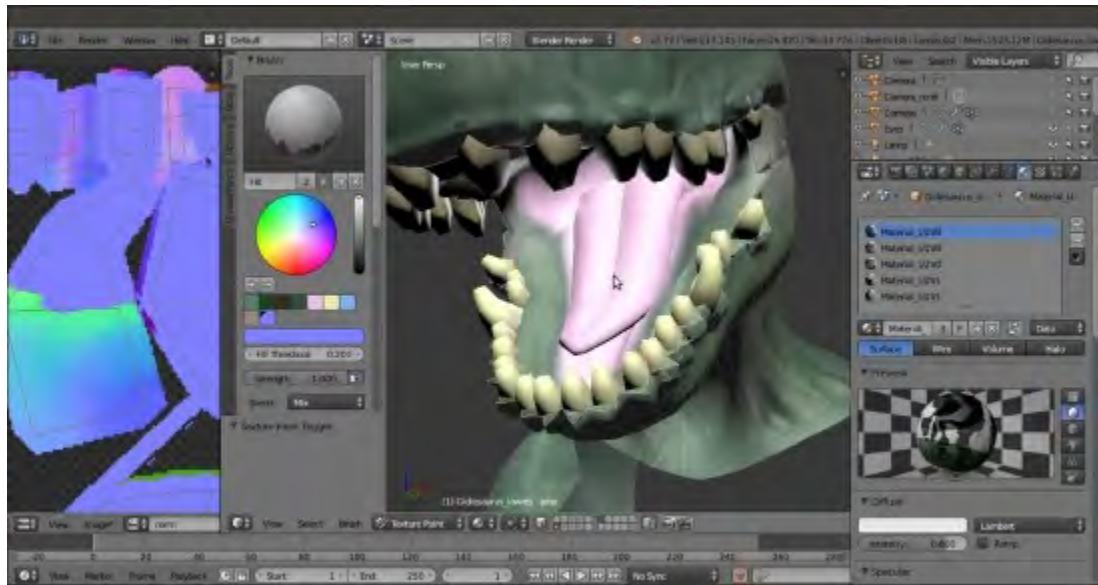
As we reopen the **mouth** by lowering the **close_mouth** shape key value to **0.000** or also by simply assigning the baked normal map to the **Gidiosaurus_lowres** object, we see that something is wrong inside the **mouth** (and, actually, also on the **teeth** and **talons**): the normals have been calculated for those parts too, but they show wrong and weird artifacts because there were no counterparts to take the normals from in the sculpted high resolution mesh.



Artifacts of the normal map in some mesh parts

The solution in this case is very simple: we must paint the areas on the baked normal map corresponding to the afflicted parts, such as the **teeth**, the **tongue**, and so on, with a *flat normal* color (**R 0.498, G 0.498, B 1.000**) to *flatten* and therefore erase the unwanted details.

We can do this directly in Blender, by selecting the vertices of the areas to be painted on and enabling the **mask tool** in the 3D viewport toolbar:



Flattening the unwanted artifacts by painting on the normal map

Alternatively, we can do it in an external painting software program such as Gimp; in this case, just delete the vertices of the parts that you don't want to change in the mesh and export the UV layer of all the remaining parts to be used as a guide to paint.

The Armor textures

The same procedure used in the previous recipe must be used for the **Armor** object, to bake the normals of the sculpted high resolution version on the low poly one.

Getting ready

So, in short, we will do the following:

1. Enable the **13th** scene layer; select the **Armor** object and go to the **Object Modifiers** window.
2. Temporarily, disable the **Armature** modifier both for rendering, and the viewport, and be sure that the **Subdivision Surface** modifier levels are both set to **2**.
3. Assign a **Shrinkwrap** modifier with a target to the **Armor_detailing** object; in the **Vertex Group** slot, select the **shrinkwrap** vertex group and, just to be sure, also check the **Keep Above Surface** item.

Also, in this case, thanks to the **shrinkwrap** vertex group, only the outside of the **armor** mesh is conformed to the sculpted mesh; the insides are not important and can even be deleted (only for the baking and, of course, on a duplicated **armor** object, as we did with the **Gidiosaurus_for_baking** object). In any case, they will be barely visible.



The Armor object prepared for the baking

How to do it...

Let's now bake the sculpted geometry in a few steps:

1. Enter **Edit Mode** and select all the mesh vertices; in the **UV/Image Editor** window, add a new **3072 x 3072** blank image and rename it as **norm2**.
2. Go out of **Edit Mode**, enable the **3rd** scene layer and select the **Armor_detailing** object, and then *Shift-select* the **Armor** object.
3. Go to the **Render** window, scroll the panel down; in the **Bake** subpanel, check the **Selected to Active** item, and set the **Margin** to **8** pixels, the **Bake Mode** to **Normals**, and the **Normal Space** to **Tangent**. Then, click on the **Bake** button.
4. Click on the **Image** item on the **UV/Image Editor** window toolbar to save the baked image as **norm2.png**, inside the **texture_making** folder.
5. Save the file as **Gidiosaurus_baking_normals.blend**.

In the following OpenGL screenshot, you can see the comparison between the sculpted and the low resolution **Armor** objects with the assigned normal map:



A comparison between the sculpted and the normal map versions of the Armor

There's more...

Inside the **texture_making** folder provided with this cookbook, there is also an already seamless **iron_tiles.png** image to be used for the **Armor**; it has been made seamless in **Gimp**, but after the mapping on the model, we'll need to fix some visible seams again by using the **Clone** brush of the **Blender Paint Tool**.

I won't go through all the required steps here, because this would be a repetition of recipes already explained in [Chapter 10, Creating the Textures](#).

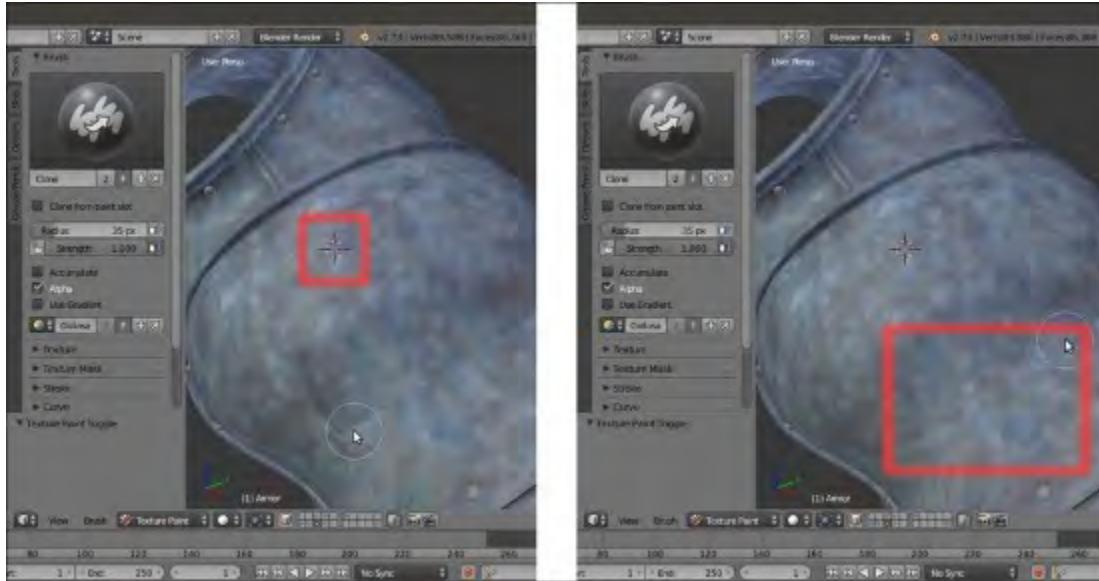
Note

Just remember that all we have to do is to bake the seamless `iron_tiles.png` image, which is mapped on the **UVMap_rust** coordinates layer, on the **UDIM UVMap** coordinates layer; in this case, shared into **two** tiles spaces and then fix the visible seams on the baked images.

So, we have to add two materials to the **Armor** object; each one with its own image texture and assigned to the vertices corresponding to each tile, and then also add the `iron_tiles.png` image to each affected material.

In short, we have to replicate the steps of the *Preparing the model to use the UDIM UV tiles, Baking the tileable scales texture into the UV tiles, and Painting to fix the seams and to modify the baked scales image maps* recipes from [Chapter 10, Creating the Textures](#).

To use the **Clone** brush (press the *I* key to call it after entering **Texture Paint** mode), press *Ctrl* + left-click on the area of the mesh you want to clone from; this will place the **3D Cursor** in that location. Then, left-click on the seams to clone the texture from the area under the **3D Cursor**:



The Clone brush is cloning the texture area at the 3D Cursor location

Besides the **Clone** brush, in this case, it is also possible to fix the seams with the **Smear** brush (*4* key).

When you are done, save the two iron images as `iron_U0V0.png` and `iron_U1V0.png` inside the **texture** folder.

See also

Be aware that the first following link is for Blender version **2.6** (seems there is very little official documentation for version **2.7** at the moment), and a few things in the **Paint Tool** have changed; in any case, I think it can still be an interesting reading:

- <http://wiki.blender.org/index.php/Doc:2.6/Manual/Textures/Painting>
- http://www.blender.org/manual/render/blender_render/textures/painting.html

Adding a dirty Vertex Colors layer and baking it to an image texture

Let's see now how to add a *dirty* map through the **Vertex Colors** tool and how to bake it to an image texture; such a texture map can be useful for the creation of the shaders (which we'll see in the next chapter).

Getting ready

To do this, we are going to use an already set .blend file:

1. Start Blender and open the `Gidiosaurus_baking_normals.blend` file; save it as `Gidiosaurus_baking_dirty.blend`.
2. Put the mouse pointer in the 3D view and press the Z key twice to switch into **Solid** viewport shading mode; click on the **14th** scene layer to enable the visibility of the **Gidiosaurus_detailing** object.

How to do it...

Let's first go with the creation of the **Vertex Colors** layer:

1. In the **Outliner**, select the **Gidiosaurus_detailing** object and then click on the mode button in the 3D viewport toolbar to select the **Vertex Paint** mode:



Selecting the Vertex Paint mode item

2. Now, click on the **Paint** item in the 3D viewport toolbar and from the menu, select the **Dirty Vertex Colors** item; the **Gidiosaurus** mesh, first filled with a plain white color, gets shaded in grayscale tones:



Using the Dirty Vertex Colors tool

3. Expand the last operation panel at the bottom of the **Tool Shelf** and press *Ctrl+click* on the **Dirt Angle** slot to enter the value **90°**; the grayscale shading on the mesh gets a lot more darker and contrasted:



Tweaking the settings for the Dirty Vertex Colors tool

4. Go back in **Object Mode** and then move to the **Material** window, where the **Body** material is already assigned to the high resolution mesh. Scroll down the panel to reach the **Shading** subpanel and enable the **Shadeless** item; then, reach down the **Options** subpanel and enable the **Vertex Color Paint** item:



The Shadeless and the Vertex Color Paint items

To understand the effect of the items we enabled in the **Material** window, just switch to the **Rendered** viewport shading mode; the mesh surface is self-illuminating and showing the dirty **Vertex Colors** layer:



The Dirty Vertex Colors layer visualized in the Rendered preview

Note that the **Shadeless** item is not actually mandatory for the baking, but is only required to see the object in the **Rendered** viewport shading mode as in the previous screenshot.

- Also, enable the **4th** scene layer (*Shift+left-click*) and in the **Outliner** window, select the **Gidiosaurus_for_baking** object. Go to the **Object Data** window to be sure that the **UVMap_norm** layer is the active one and then go to the **Render** window and scroll down to the bottom, to the **Bake** subpanel; click on the **Bake Mode** slot to select the **Textures** item from the pop-up menu:



Baking the Dirty Vertex Colors layer to Textures

- With the **Gidiosaurus_for_baking** object still selected, enter **Edit Mode** and select all the vertices; in the **UV/Image Editor** window, add a new **3072 x 3072** blank image renamed as **vcol**.
- Go out of **Edit Mode** and in the **Outliner**, select the **Gidiosaurus_detailing** object; then, *Shift-select* the **Gidiosaurus_for_baking** object and go to the **Bake** subpanel under the **Render** window to click on the **Bake** button:



The final baked "vcol.png" image map

8. Save the baked image as `vcol.png` into the `texture_making` folder.
9. Enable the **3rd** scene layer, select the **Armor_detailing** object, and repeat the procedure; save the baked image as `vcol2.png` in the `texture_making` folder and also save the file:



The baked vertex color layer for the Armor

How it works...

The **Vertex Colors** tool can add a color to each vertex of the mesh, so it's actually possible to paint an object without the need for an image texture; the denser the mesh, the better this works.

The **Dirty Vertex Colors** tool uses the proximity and the depth of folds and creases on the mesh surface to calculate grayscale values to be assigned to the vertices; thanks to the **Vertex Color Paint** item, enabled in the **Material** window, this grayscale shows up in the rendering and so it's also possible to bake it into an image.

See also

- http://wiki.blender.org/index.php/Doc:2.6/Manual/Materials/Special_Effects/Vertex_Paint
- http://www.blender.org/manual/render/blender_render/materials/special_effects/vertex_paint.html

The Quick Edit tool

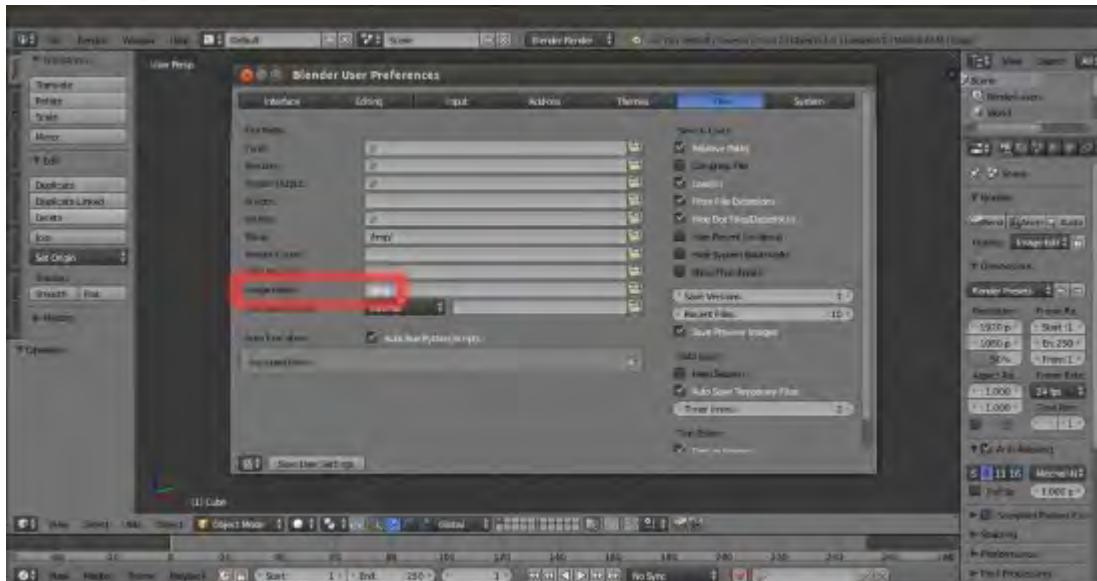
It's time to talk a bit about a very useful Blender tool: the **Quick Edit** tool.

Through this tool, it's possible to export a screenshot of the model in our favorite 2D painting software (**Gimp** or **Photoshop**, or whatever), paint on it using a new alpha background layer, and reassign the painted layer to the model in Blender, which is UV-mapped on the selected UV coordinates layer. All this, in just a few clicks.

Getting ready

In our case, we don't actually need to use this tool to refine the textures for the **Gidiosaurus**, so this recipe is going to be just an example. By the way, to fully understand how to use the tool, I suggest you to follow all the steps; just don't save the file at the end (or save it with a different name in a different directory if you want to keep it). So, carry on with the following:

1. Start Blender and call the **User Preferences** panel (*Ctrl + Alt + U*); go to the **File** tab and, in the **Image Editor** slot (*Path to an image editor*), write the path to your 2D image painting software installation (this is also done by clicking on the open/browse button at the right end of the slot). The path, of course, changes based on your OS; in my case, in **Linux Ubuntu**, it's enough to write `gimp`:



The Image Editor path in the File tab of the User Preferences panel under Linux Ubuntu

2. Click on the **Save User Settings** button at the bottom-left of the panel and close it.

How to do it...

Once you've set the path to the image editor, let's load our **Gidiosaurus** file:

1. Load the `Gidiosaurus_painting_BI.blend` file and maximize it as much as possible in the 3D viewport.

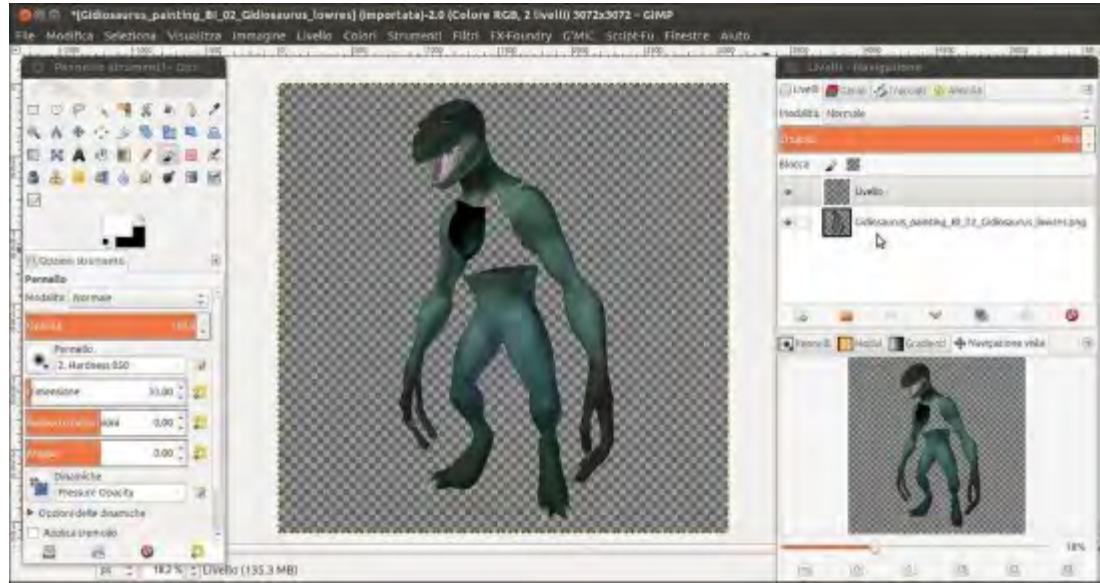
You can use both the **User** or the **Camera** view; it doesn't make any difference for the tool to work. By the way, it would be a good idea to use the **Camera** view so as to have a fixed point of view for any other case.

2. If necessary, press the `T` key to call the **Tool Shelf** panel; select the **Gidiosaurus** object and then go in **Texture Paint** mode.
3. Go to the **External** subpanel under the **Tools** tab; set a size for the screenshot to be exported (by default, it's **512 x 512** pixels; I set it to **3072 x 3072** pixels) and then click on the **Quick Edit** button:



The External Image Editor subpanel

After a while, the image editor automatically starts (in my case, it's **Gimp 2.8**) and opens the screenshot of the model:



The screenshot previously visible in the Blender Camera view opened in Gimp

4. Add a new transparent layer and start to paint on it, adding some kind of tribal make-up decoration to the Gidiosaurus:



Tribal painting on the Gidiosaurus warrior

5. When you are done, deselect the visibility for the export layer and export the transparent painted one *by saving it with the same name as the exported one*. That is, the **Quick Edit** tool exported the screenshot by saving a .png image inside the blend file directory with the name `Gidiosaurus_painting_BI_02_Gidiosaurus_lowres.png`; export the painted

layer by saving it as `Gidiosaurus_painting_BI_02_Gidiosaurus_lowres.png` as well:



The Gimp layer with the tribal painting "a solo"

This is necessary for Blender to find it in the next step.

6. Back in Blender, click on the **Apply** button under the **External** subpanel and watch the new layer added to the model in the 3D viewport:

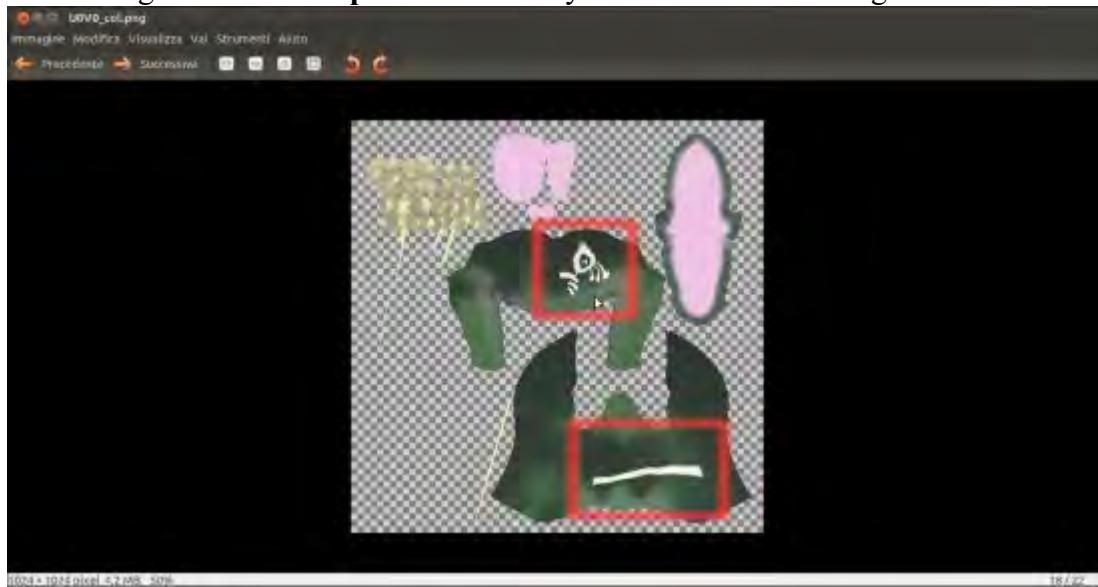


7. To make the textures editing permanent, click on the **Save All Images** button, both under the **Slots** tab and the **Image** item on the **UV/Image Editor** window toolbar:



The tribal painting transferred on the 3D model

The editing we did in **Gimp** is now correctly transferred on the image textures:



The tribal painting transferred on the image map

How it works...

As you have seen, the **Quick Edit** tool worked like a charm on all the **5** different materials assigned to the **Gidiosaurus** model for the painting. Be careful that, at least at the moment, this doesn't seem to work with **nodes materials** (which we'll see in the next chapter).

Chapter 12. Creating the Materials in Cycles

In this chapter, we will cover the following recipes:

- Building the reptile skin shaders in Cycles
- Making a node group of the skin shader to reuse it
- Building the eyes' shaders in Cycles
- Building the armor shaders in Cycles

Introduction

In [Chapter 10, Creating the Textures](#), and in [Chapter 11, Refining the Textures](#), we have prepared all the necessary texture images for the **Gidiosaurus** skin and for the iron **Armor** (the creation process for some textures, specifically the two textures for the character's **eyes**, hasn't been described, but basically it's a process similar to what we have already seen).

In this chapter, we'll see how to use these textures and how to set up the materials for the **Gidiosaurus** and the **Armor** in the **Cycles Render** engine.



A rendered example of the Cycles' shader final result

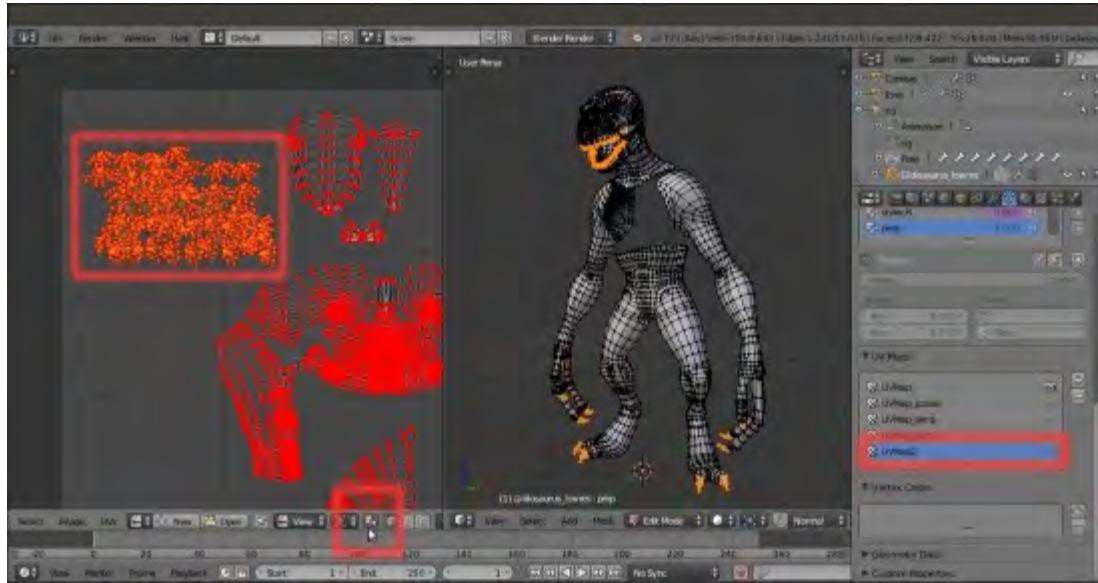
Building the reptile skin shaders in Cycles

So, let's start with the **Gidiosaurus** skin.

Getting ready

But first, as usual, we must prepare the file:

1. As the very first step, go to the `texture_making` folder and move the textures `vcol1.png`, `vcol2.png`, `norm.png`, and `norm2.png` to the `textures` folder.
2. Then start Blender and open the `Gidiosaurus_baking_normals.blend` file we saved in [Chapter 11, Refining the Textures](#).
3. Switch the left **UV/Image Editor** window with a **Node Editor** window and press the *N* key to get rid of the **Properties** sidebar. Put the mouse pointer in the 3D viewport to the right and press the *T* key to get rid of the **Tool Shelf** panel, then press the *Z* key twice to go in **Solid** viewport shading mode.
4. Enable the **3rd** scene layer, select and delete the **Armor_detailing** object (press the *X* key, then left-click to confirm).
5. Enable the **4th** scene layer and select and delete the **Gidiosaurus_for_baking** object as well. Enable the **14th** scene layer, and also select and delete the **Gidiosaurus_detailing** and the **enamels** objects.
6. Enable the **11th** scene layer and right-click on the **Gidiosaurus_lowres** object to select it.
7. It's not mandatory but, in case it is not already disabled, it is best to go to the **Object Modifiers** window and disable the **Armature** modifier visibility in the viewport by clicking on the eye icon button.
8. Go to the **UV Maps** subpanel under the **Object Data** window and select the **UVMap** coordinates layer (the first one); press *Tab* to enter **Edit Mode**, then click on the + icon button to the right side of the **UV Maps** subpanel to add a new coordinates layer; rename it **UVMap2**.
9. Go to the left window and change it into a **UV/Image Editor**; one by one, select the UV islands of the **talons** (at the moment these are placed inside the other **UDIM** tile spaces), and move them to the default **U0V0** tile, to overlap the location of the **teeth** islands. The reason for this will be clear later, when we will reuse the same color map both for the teeth and for all the talons.
10. If necessary, remember to disable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar:



Moving the talon islands to overlap the teeth islands inside the default UOV0 tile space

11. Go out of **Edit Mode** and switch the **UV/Image Editor** window back to a **Node Editor** window.
12. Click on the *Engine to use for rendering* slot on the main top header to switch to the **Cycles Render** engine.
13. Also, enable the **6th** scene layer to show the **Lamps**. Go to the **Outliner**, unhide and delete the **Lamp.001** object and select the **Lamp** object; in the **Object Data** window, change the type to **Spot** and then click on the **Use Nodes** button. Set the **Strength** to **10.000** and the **Color** to **R 1.000, G 1.000, B 0.650**, then set the **Size** to **0.500** and enable the **Multiple Importance** item (the **Multiple Importance Sampling** helps in reducing noise for big lamps and sharp glossy reflections, at the cost of the samples rendering a bit slower).
14. Put the mouse pointer in the 3D viewport and press **N** to call the **Properties** sidepanel; in the top **Transform** subpanel, type: **Location X = 6.059204, Y = -9.912249** and **Z = 7.546275**, and **Rotation X = 55.788944°, Y = 0°** and **Z = 30.561825°**.
15. Go to the **Render** window and, under the **Sampling** subpanel, set the **Samples** to **400** for **Render** and **300** for **Preview**.
16. Go down to the **Light Paths** subpanel and disable the **Reflective Caustics** item, then set **Filter Glossy** to **1.000**.
17. Re-select the **Gidiosaurus_lowres** object and go to the **Material** window:

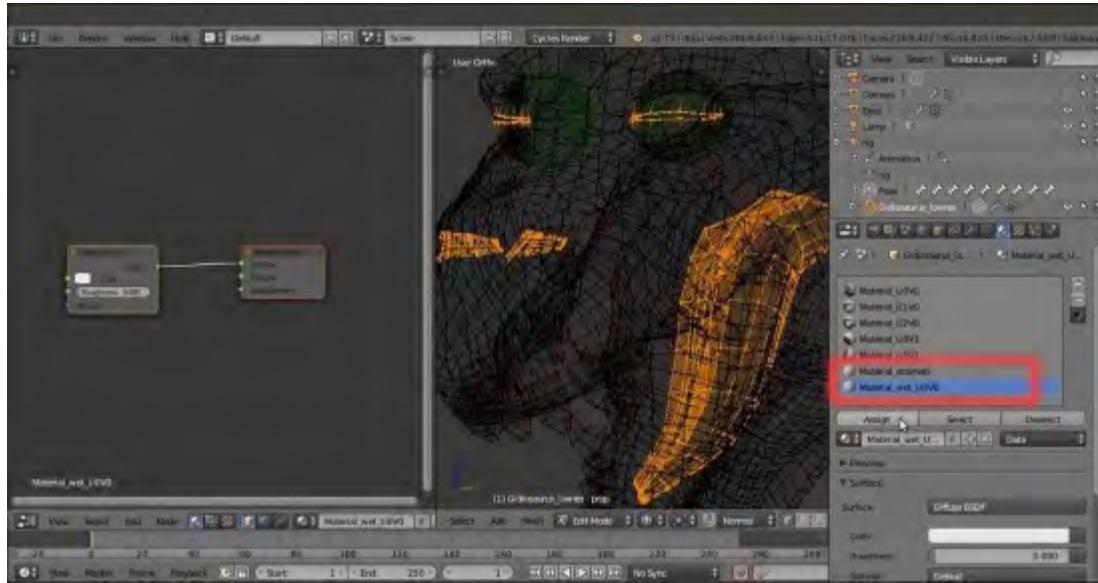


The 5 materials under the Cycles render engine

As you can see in the previous screenshot, the **Gidiosaurus_lowres** object has already assigned the **5** materials corresponding to the **5 UDIM** tile spaces (see [Chapter 5, Unwrapping the Low Resolution Mesh](#), and [Chapter 10, Creating the Textures](#)).

The materials have been created under **Blender Internal** so, switching to **Cycles**, they show but aren't *initialized* as **node materials** yet; besides this, just before starting the creation of the first Cycles material, we must add **two** more materials.

18. Click the + icon button **twice** (*Add a new material slot*) to the right side of the **Material** window to add **two** new material slots.
19. Select the penultimate slot and click on the **New** button; rename the new material as **Material_enamels**. Select the last slot, click on the **New** button and rename it as **Material_wet_U0V0**.
20. Press **Tab** to enter **Edit Mode** and select all the vertices of the **teeth** and the **talons**; assign them to the **Material_enamels** slot.
21. Zoom on the **Gidiosaurus** head; select the vertices of the inner **nostrils**, of the inner edges of the **eyelids** and of the **tongue**, as shown in the following screenshot, and assign them to the **Material_wet_U0V0** slot:



Selecting the vertices of the "wet" areas of the character's head to assign them to the "Material_wet_U0V0" slot

22. Save the file as `Gidiosaurus_shaders_start.blend`.

How to do it...

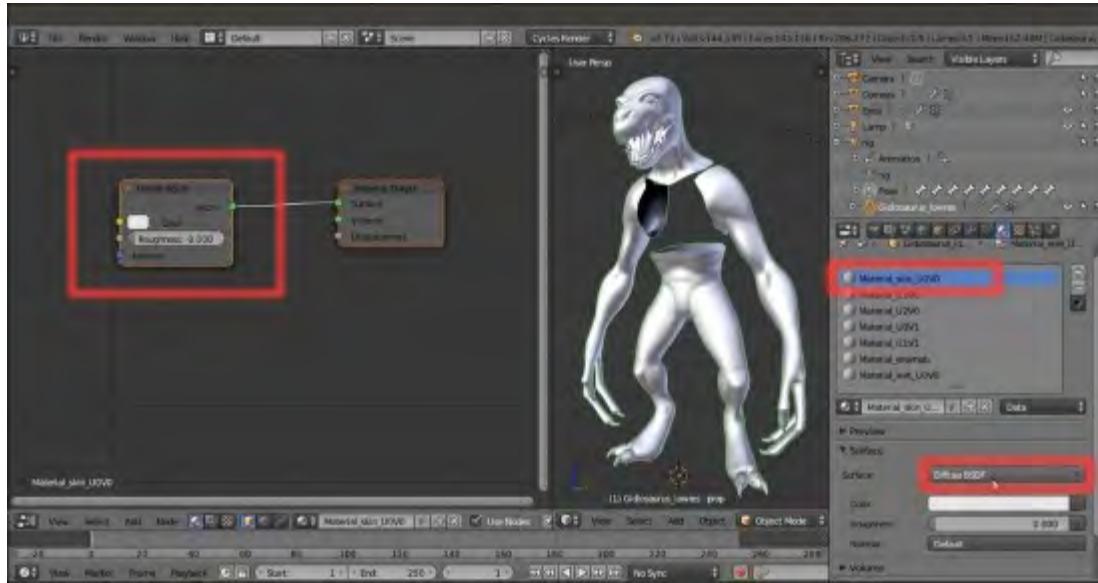
We know that the skin of our character is shared in **5** different materials; we are going to focus on the **head** (`Material_U0V0`), as the more representative one.

Once we are happy with the result, we will also copy (with all the due differences) the material to the other **body** parts.

Therefore, the steps are as follows:

1. In the materials list inside the **Material** window, select the `Material_U0V0` (the first top one) and press `Ctrl + left-click` on it to rename it as `Material_skin_U0V0`; then, move down and click on the **Use Nodes** button inside the **Surface** subpanel.

Immediately, a **Diffuse BSDF** shader node (already connected to a **Material Output** node) appears inside the **Node Editor** window to the left of the screen and listed in the **Surface** slot inside the **Surface** subpanel to the right:



The Diffuse BSDF shader node connected to the Material Output node

2. In the **Surface** subpanel, under the **Material** window, click on the **Surface** slot that now shows the **Diffuse BSDF** shader: in the pop-up menu that appears, select a **Mix Shader** node:



Switching the Diffuse BSDF shader node with a Mix Shader node through the Material window drop-down list

The **Surface** slot now shows the **Mix Shader** node item, and right below there are two new **Shader** slots that at the moment show the **None** item; in fact, looking at the nodes inside the

In the **Node Editor** window, we see that the **Diffuse BSDF** shader node has been replaced by a **Mix Shader** node, and that the two (green) **Shader** input sockets are still empty:



The **Mix Shader** node with its two **shader** input sockets in the **UV/Image Editor** window and in the **Material** window

3. Click on the first **Shader** slot under the **Surface** subpanel to select, again from the pop-up menu, a **Diffuse BSDF** shader node; click on the second **Shader** slot and select a **Mix Shader** node; both the two new nodes are added and connected to the proper input socket, as we can see in the **Node Editor** window:



Two new nodes connected to the two **shader** input sockets of the **Mix Shader** node

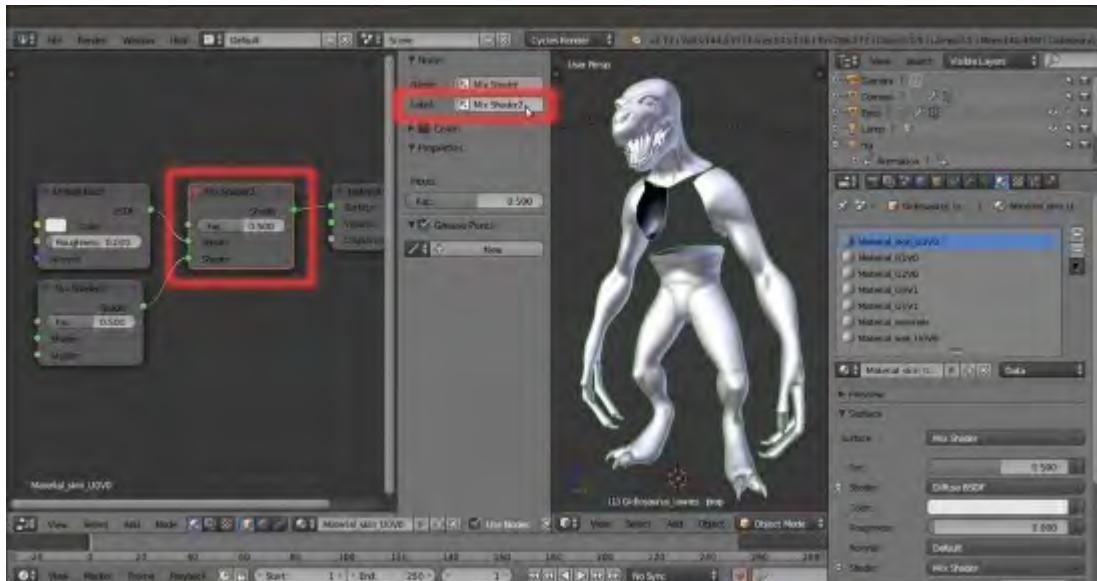
At this point, to avoid confusion, it's already better to start to label the various nodes with meaningful names.

4. Put the mouse pointer inside the **Node Editor** window and press the **N** key to call the **Properties** sidepanel.
5. Select the last **Mix Shader** node we added to the material and then go to click on the **Label** slot inside the top **Name** subpanel of the side **Properties** panel: type **Mix Shader1**:



Labeling the nodes

6. Select the other **Mix Shader** node (the *old* one) and repeat the procedure by labeling it as **Mix Shader2**:



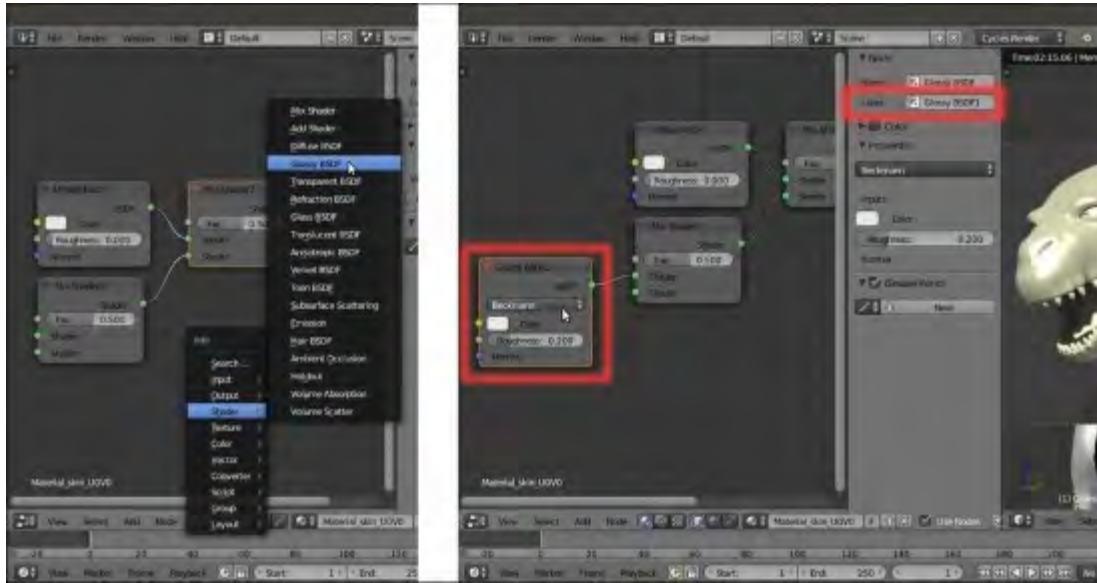
Labeling the nodes again

7. Put the mouse pointer on the 3D viewport and press the *0* key on the numpad to enter the **Camera** view.
8. Press *Shift + B* and by left-clicking draw a box around the **head** of the **Gidiosaurus** character to crop the area that can be rendered.
9. Zoom to the red square by scrolling the mouse wheel and then press *Shift + Z* to switch the **Viewport Shading** mode to **Rendered**:



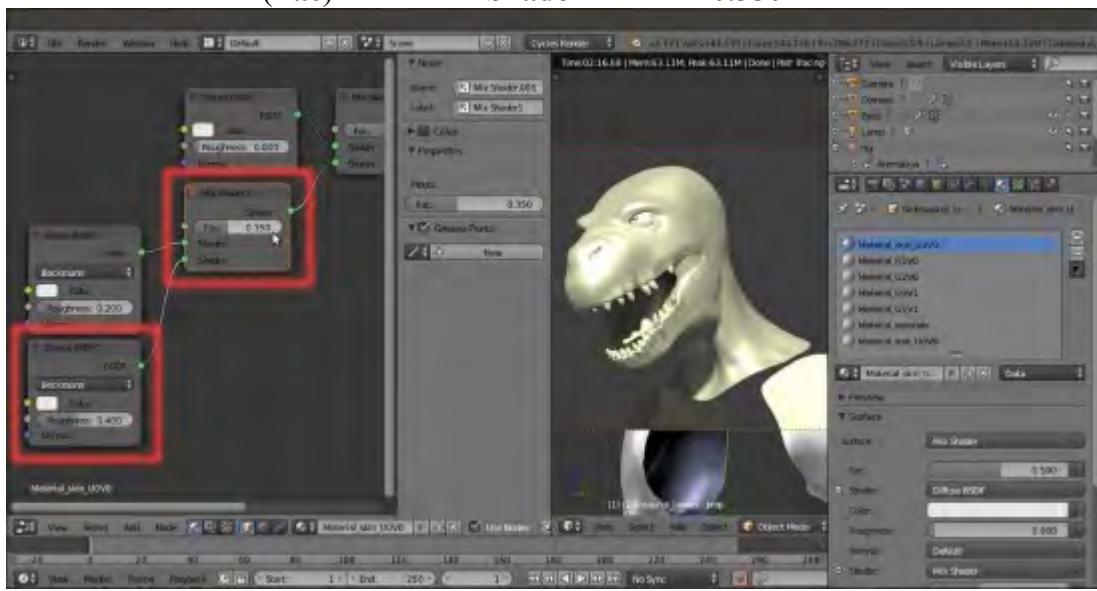
Cropping the renderable area and zooming to it

10. Put the mouse pointer inside the **Node Editor** window and press *Shift + A*. In the pop-up panel that appears, navigate to **Shader** and then click on the **Glossy BSDF** item to add the node; as it appears, move the mouse to place it to the left side of the **Mix Shader1** node.
11. Label it as **Glossy BSDF1**, connect its output to the first top **Shader** input socket of the **Mix Shader1** node, and set **Distribution** to **Beckmann**:



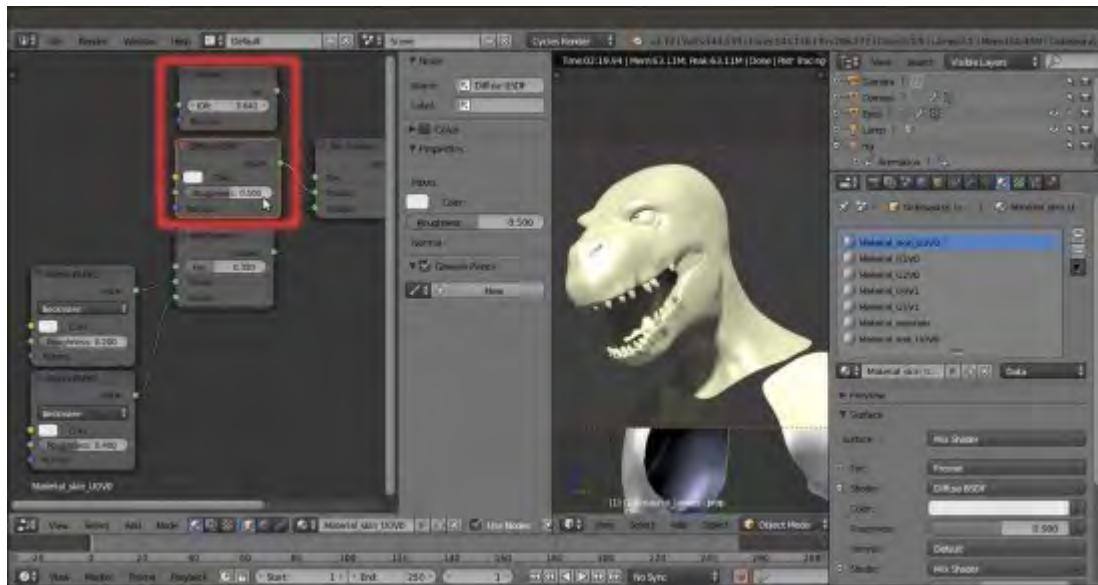
Adding a Glossy BSDF shader node and labeling it

12. Add a second **Glossy BSDF** shader node (**Shift + A | Shader | Glossy BSDF**) and place it right under the previous one; label it as **Glossy BSDF2**, connect its output to the second **Shader** input socket of the **Mix Shader1** node, and set **Distribution** to **Beckmann** as well and the **Roughness** to **0.400**.
13. Set the factor value (**Fac**) of the **Mix Shader1** node to **0.350**:



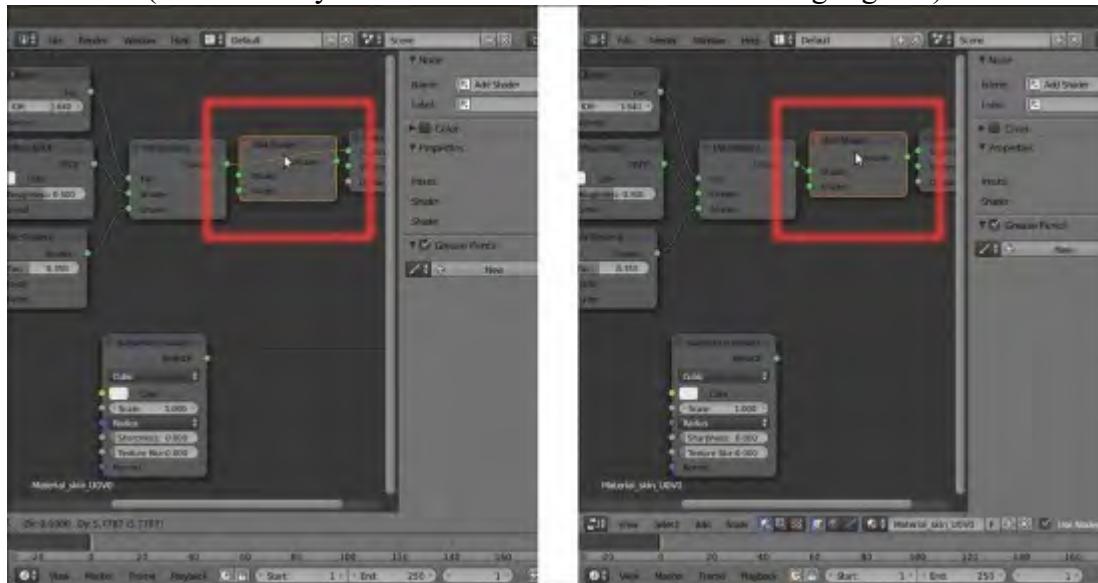
Adding a second Glossy shader node and blending it with the first one through the Fac value of the Mix Shader1 node

14. Add a **Fresnel** node (*Shift + A | Input | Fresnel*) and connect its **Fac** output to the **Fac** input socket of the **Mix Shader2** node; set the **IOR** value to **3.840**. Set the **Roughness** value of the **Diffuse BSDF** shader node to **0.500**:



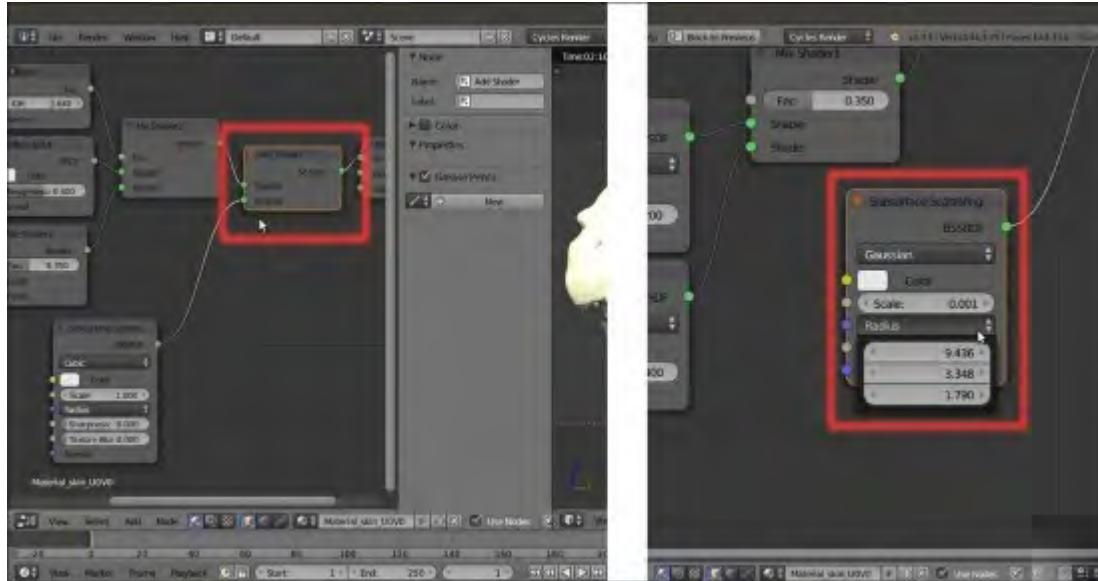
Adding a Fresnel node to set the Index of Refraction value to blend the diffuse with the glossy components

15. Add a **Subsurface Scattering** node (*Shift + A | Shader | Subsurface Scattering*) and an **Add Shader** node (*Shift + A | Shader | Add Shader*). Move this last one to the link that connects the **Mix Shader2** node to the **Material Output** node in order to paste it automatically between the two nodes (automatically when the connection line becomes highlighted):



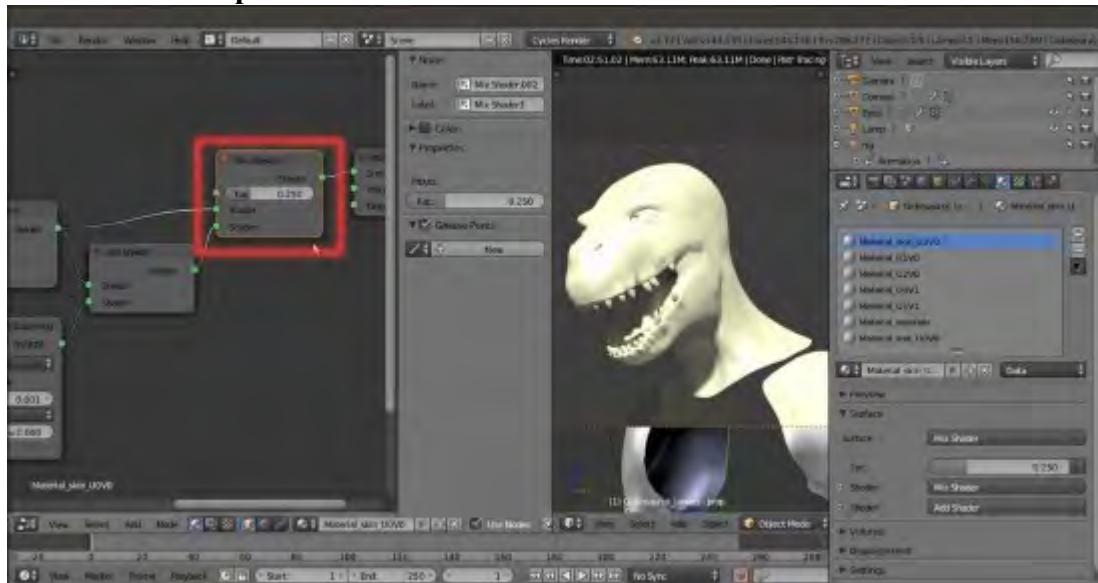
Automatically joining the Add Shader node

16. Connect the output of the **Subsurface Scattering** node to the second **Shader** input socket of the **Add Shader** node. In the **SSS** node, change **Falloff** from **Cubic** to **Gaussian**, set the **Scale** to **0.001** and click on the **Radius** button to set the **RGB** to **9.436, 3.348** and **1.790**:



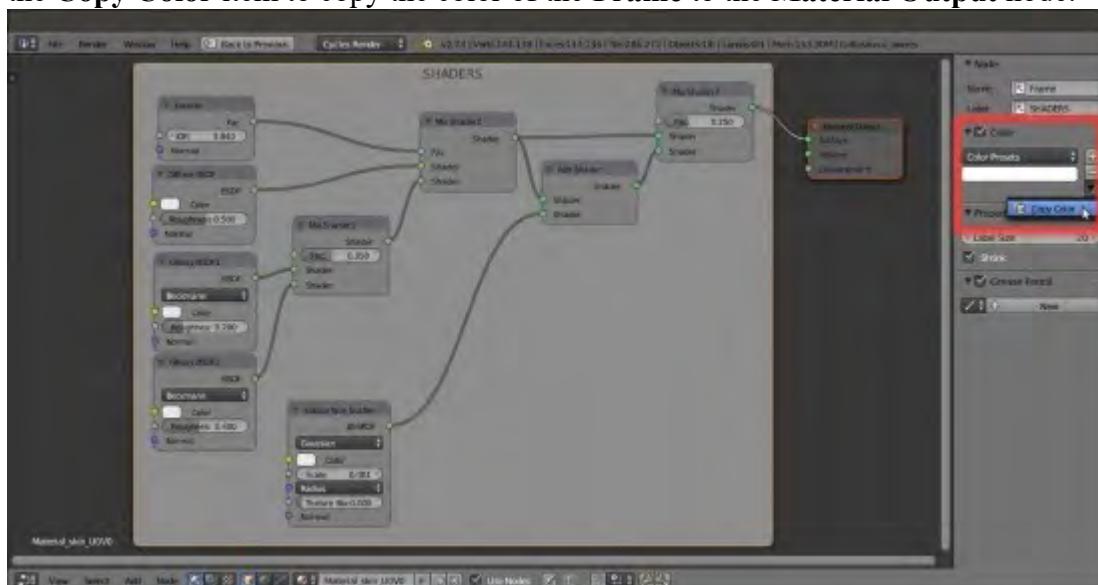
Connecting and setting the SSS node

17. Add a new **Mix Shader** node (**Shift + A | Shader | Mix Shader**) and label it as **Mix Shader3**. Connect the output of the **Mix Shader2** node to the first **Shader** input socket of the **Mix Shader3** node, and the output of the **Add Shader** node to its second **Shader** input socket. Set the **Fac** of the **Mix Shader3** node to **0.250** and connect its output to the **Surface** input socket of the **Material Output** node:



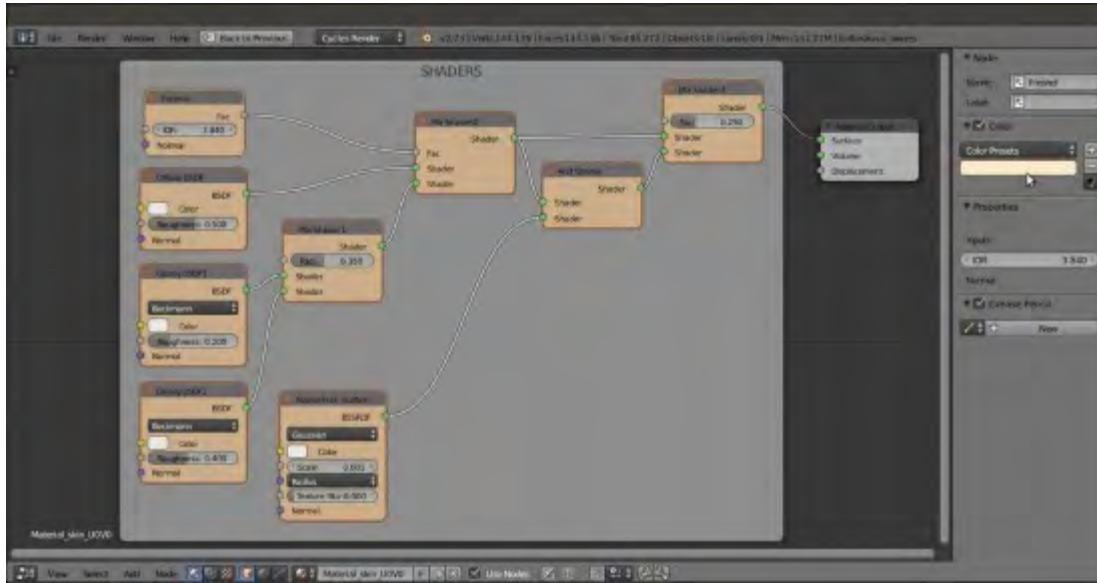
A little trick to tweak the influence of the Add shader node

18. Add a **Frame** (*Shift + A | Layout | Frame*), box-select all the nodes (except the **Material Output** node) and then press *Ctrl + P* to parent them to the frame; label the frame as **SHADERS**.
19. Select the **SHADERS** frame and go to the **Properties** sidepanel. Expand the **Color** subpanel (right under the **Node** subpanel) by clicking on the little horizontal black arrow, and enable the **Color** checkbox.
20. Click on the color slot and set a light color of your choice (I set it to **RGB 1.000**, which is totally white). Then click on the + icon button to the side and in the **Name** slot of the **Add Node Color Preset** pop-up panel, write **Frame**, then click the big **OK** button.
21. Select the **Material Output** node and then *Shift*-select the **Frame** again, then go to the **Color** subpanel and click on the big vertical arrow under the + and – icon buttons to the side. Click on the **Copy Color** item to copy the color of the **Frame** to the **Material Output** node:



The **SHADERS** frame with the nodes and the **Copy Color** tool under the **N** sidepanel

22. Select any one of the other nodes, for example the **Fresnel** node, enable the **Color** checkbox and set a new color of your choice (for these nodes, I set it to **R 1.000, G 0.819, B 0.617**, which is a light brown).
23. Click on the + icon button to the side and in the **Name** slot of the **Add Node Color Preset** pop-up panel, write **Shaders**, then click the big **OK** button.
24. Now box-select all the other nodes inside the frame and click on the **Copy Color** item to copy the color from the **Fresnel** node to all the other selected nodes at once:



Copying the label color from one node to all the other selected nodes

At this point we have completed the basic shader for the skin; what we have to do now is to add the textures we painted in both [Chapter 10, Creating the Textures](#), and [Chapter 11, Refining the Textures](#).

So:

25. Put the mouse pointer into the **Node Editor** window and add an **Image Texture** node (*Shift + A | Texture | Image Texture*); label it as **COL** and then use *Shift + D* to duplicate it; move the duplicated one down and change its label to **SCALES**.

As you label the newly added nodes, also assign colors to them to make them more easily readable inside the **Node Editor** window, and save these colors as presets as we did at step 20.

26. Click on the **Open** button of the **COL** node and browse to the **textures** folder. There, load the image **U0V0_col.png**.
27. Click on the **Open** button of the **SCALES** node and browse to the **textures** folder. There, load the image **U0V0_scales.png**; set the **Color Space** to **Non-Color Data**.
28. Add a **MixRGB** node (*Shift + A | Color | MixRGB*) and label it as **Scales_Col**; connect the **Color** output of the **COL** node to the **Color1** input socket of the **Scales_Col** node and the **Color** output of the **SCALES** node to its **Color2** input socket. Set the **Fac** to **1.000** and the **Blend Type** to **Divide**.
29. Connect the output of the **Scales_Col** node to the **Color** input socket of the **Diffuse BSDF** shader node inside the **SHADERS** frame.

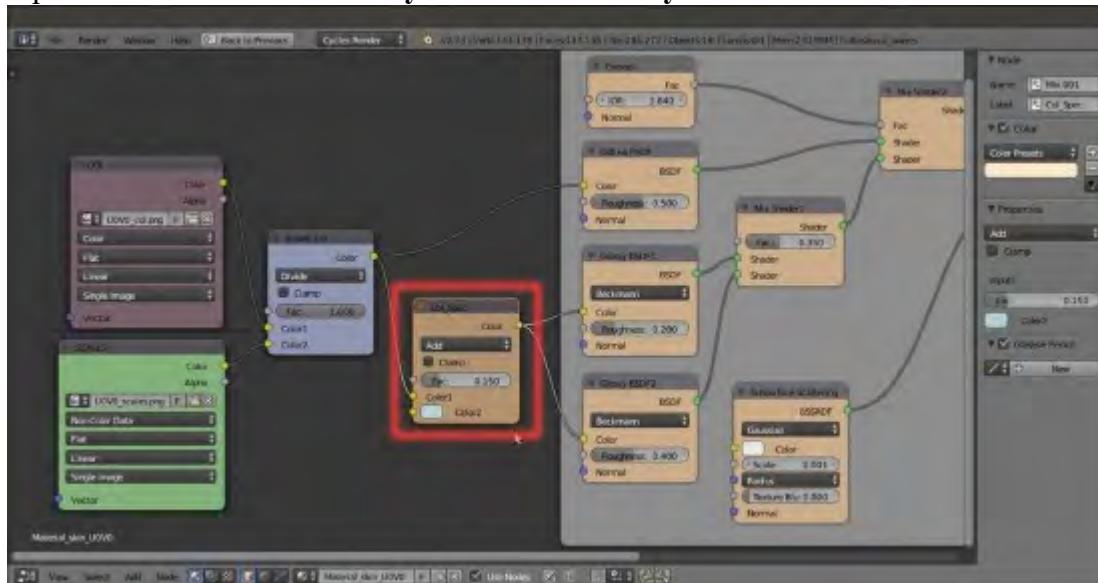
The result so far is visible in the real-time rendered preview to the right:



The rendered result of the two combined image texture nodes

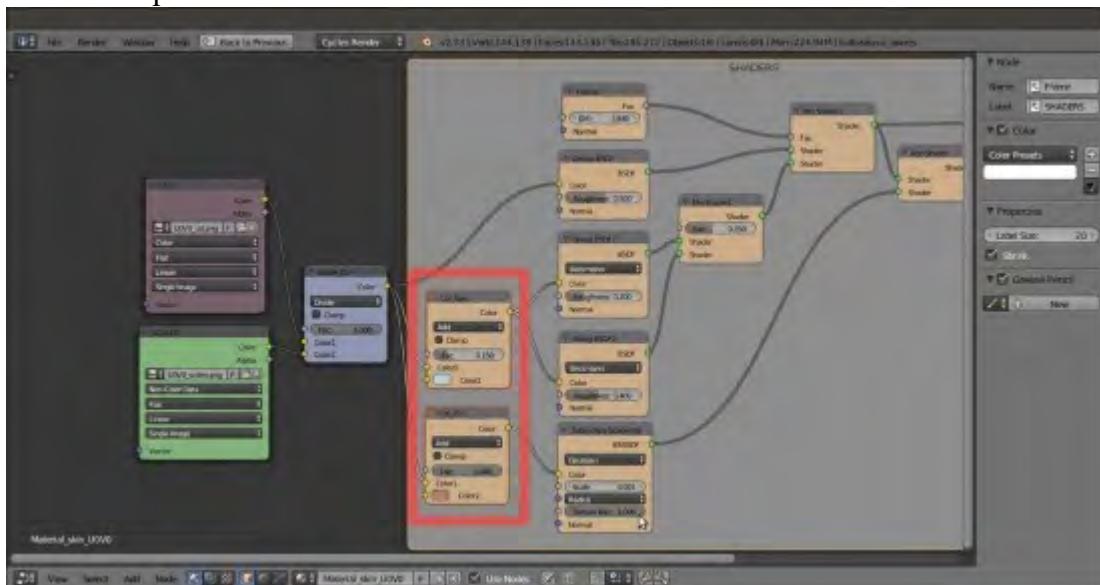
As you can see, the glossy component is strong in this one! We must lessen the effect, to obtain a more natural look.

30. Add a new **MixRGB** node (**Shift + A | Color | MixRGB**) and label it as **Col_Spec**; set the **Color2** to **R 0.474, G 0.642, B 0.683**, then also connect the output of the **Scales_Col** node to the **Color1** input socket of the **Col_Spec** node.
31. Set the **Fac** value to **0.150** and the **Blend Type** to **Add**, then connect its output to the **Color** input sockets of both the **Glossy BSDF1** and **Glossy BSDF2** nodes:



Varying the textures color output for the glossy component

32. Press **Shift + D** to duplicate the **Col_Spec** node and label the duplicate as **Col_SSS**; set the **Fac** value to **1.000** and the **Color2** to **R 0.439, G 0.216, B 0.141**. Connect the **Color** output of the **Scales_Col** node to the **Color1** input socket of the **Col_SSS** node and the output of this latter node to the **Color** input socket of the **Subsurface Scattering** node; increase its **Texture Blur** to the maximum value.
 33. *Shift-select* the **Col_Spec** and the **Col_SSS** nodes and then also the **SHADERS** frame, and press **Ctrl + P** to parent them:



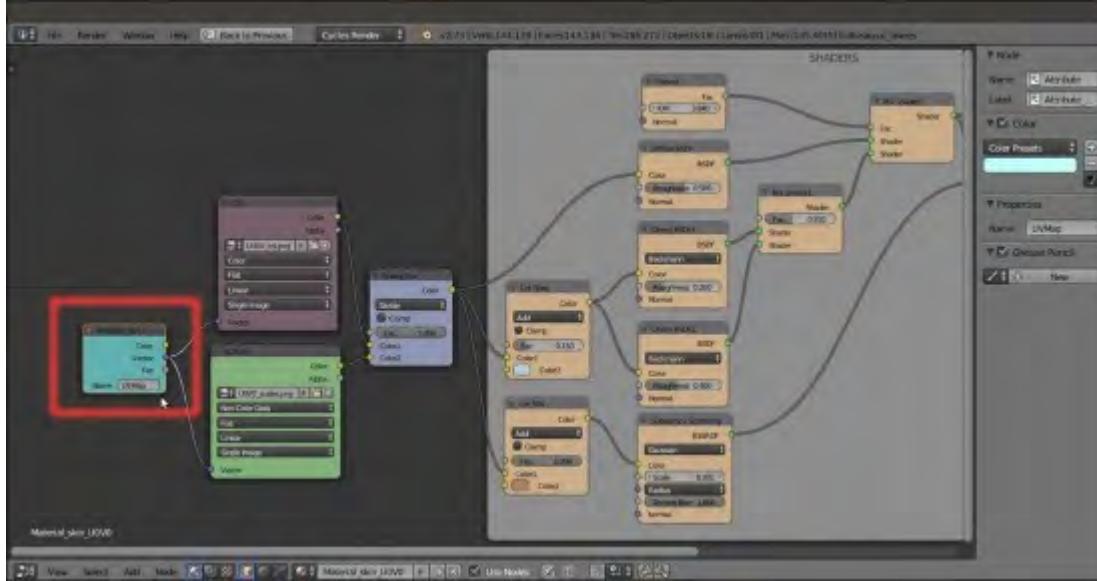
Varying the textures color output also for the SSS node

The new result looks a lot better:



A better result

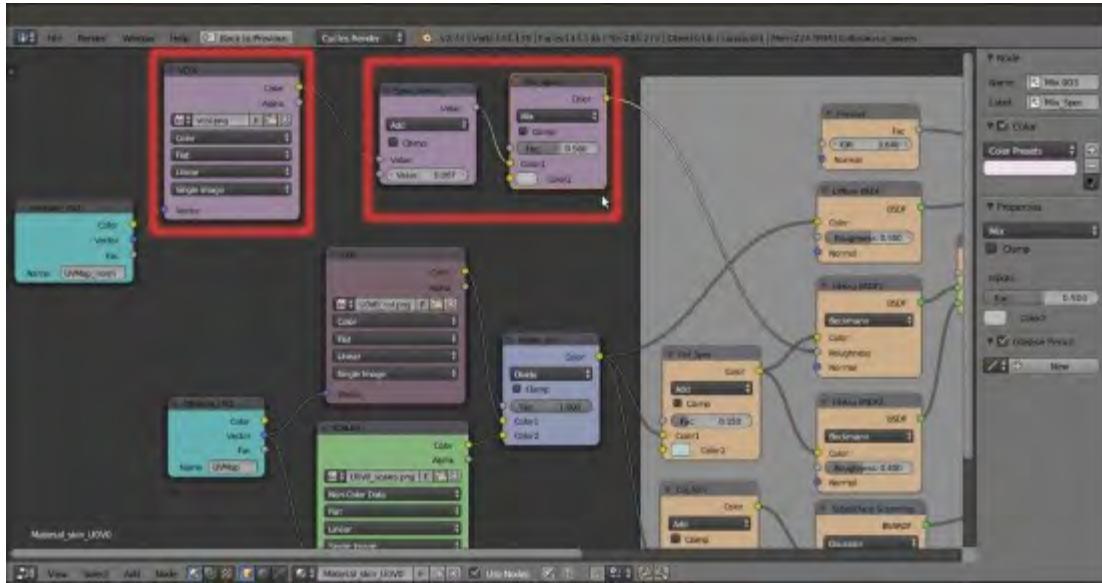
34. Add an **Attribute** node (*Shift + A | Input | Attribute*) and label it as **Attribute_UV1**. Connect its **Vector** output to the **Vector** input sockets of the **COL** and **SCALES** nodes and in the name field type **UVMap**:



Adding the Attribute node to establish the UV coordinates layer to be used

By the way, the glossy component is still a little unnatural.

35. Add a new **Image Texture** node (*Shift + A | Texture | Image Texture*) and label it as **VCOL**. Click on the **Open** button, browse to the texture folder and load the image **vc01.png**.
36. Press *Shift + D* to duplicate the **Attribute** node, change the label to **Attribute_UV2**, and change the **Name** field to **UVMap_norm**. Connect its **Vector** output to the **Vector** input of the **VCOL** node.
37. Add a **Math** node (*Shift + A | Converter | Math*) and a **MixRGB** node (*Shift + A | Color | MixRGB*); connect the **Color** output of the **VCOL** node to the first **Value** input socket of the **Math** node; label this one as **Spec_soften** and set the second **Value** to **0.007**. Connect its **Value** output to the **Color1** input socket of the **MixRGB** node, which is now labeled as **Mix_Spec**.
38. Connect the **Color** output of the **Mix_Spec** node to the **Roughness** input socket of the **Glossy BSDF1** node:



Using the baked Vertex Color image to "soften" the character's skin specularity

The specularity is now a bit more realistic:

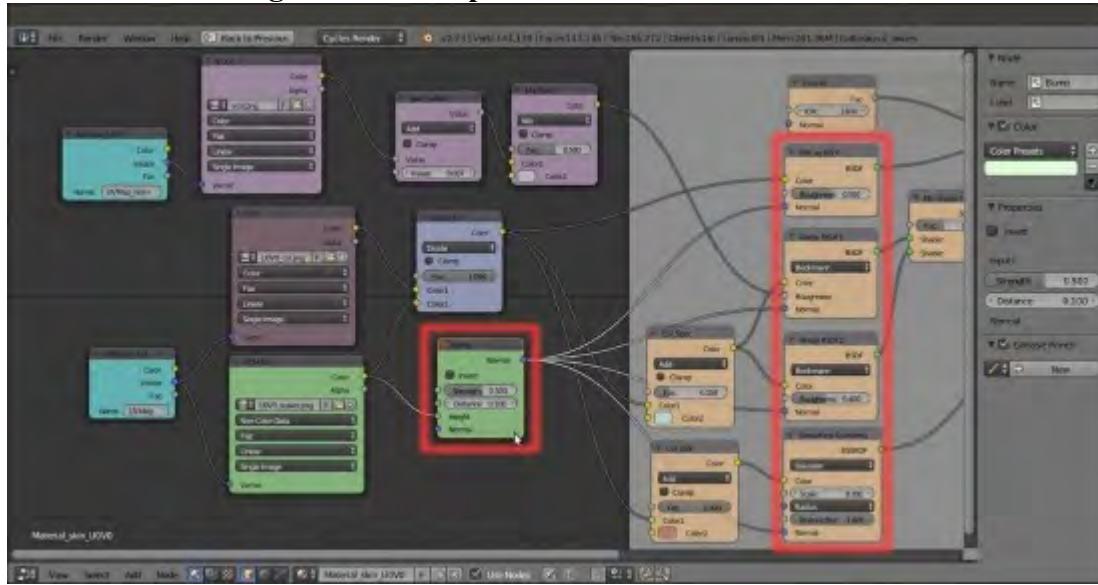


And the rendered result of this operation

Anyway, it's still missing the contribution of the bump effect.

39. Add a **Bump** node (*Shift + A | Vector | Bump*); connect the output of the **SCALES** node to the **Height** input socket of the **Bump** node and the **Normal** output of this latter node to the **Normal**

input socket of the **Diffuse BSDF**, **Glossy BSDF1**, **Glossy BSDF2**, and **Subsurface Scattering** nodes. Set the **Strength** of the **Bump** node to **0.500**:



Adding the bump pattern to the shaders

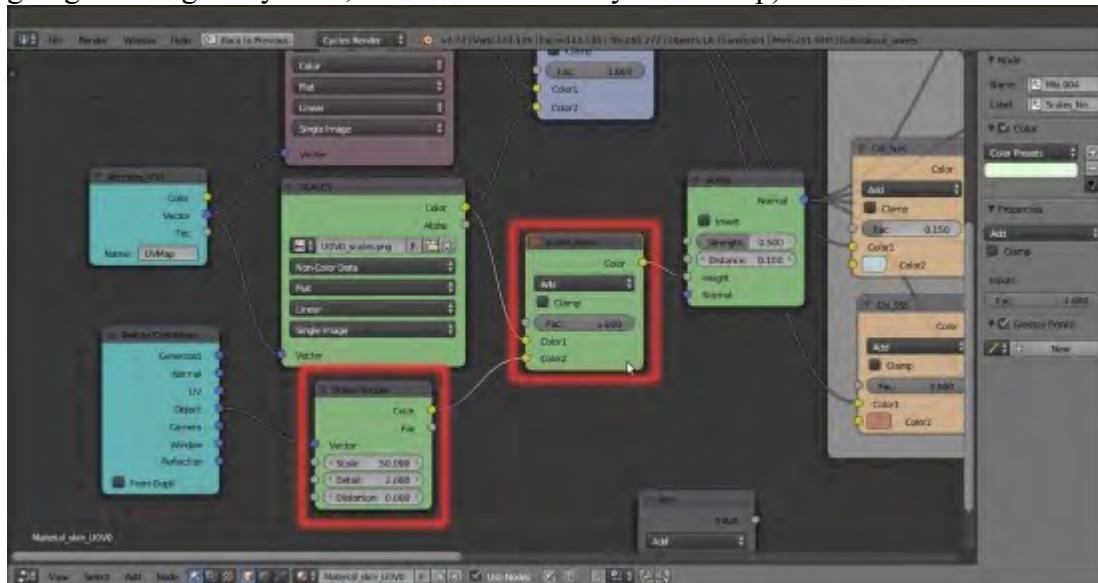
Now we start to see something!



The bump effect in the rendered preview

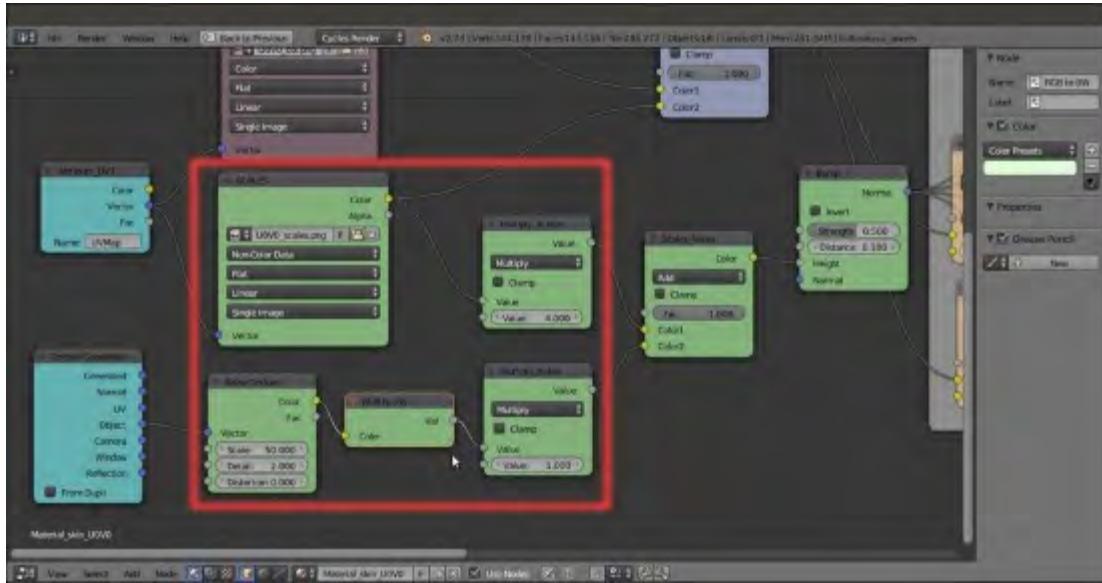
By the way, the bump pattern is too even and, therefore, unrealistic; we must therefore *break* it in some way.

40. Add a **Noise Texture** node (*Shift + A | Texture | Noise Texture*) and a **Texture Coordinate** node (*Shift + A | Input | Texture Coordinate*). Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Noise Texture** node, then set the **Scale** of the texture to **50.000**.
41. Add a **Math** node (*Shift + A | Converter | Math*) and a **MixRGB** node (*Shift + A | Color | MixRGB*). Connect the **Color** output of the **SCALES** node to the **Color1** input socket of the **MixRGB** node, and the **Color** output of the **Noise Texture** to the **Color2** input socket.
42. Set the **MixRGB** blend type to **Add**, the **Fac** value to **1.000** and label it as **Scales_Noise**. To see the effect, connect its **Color** output to the **Height** input socket of the **Bump** node (but this is going to change very soon, so it's not mandatory at this step):



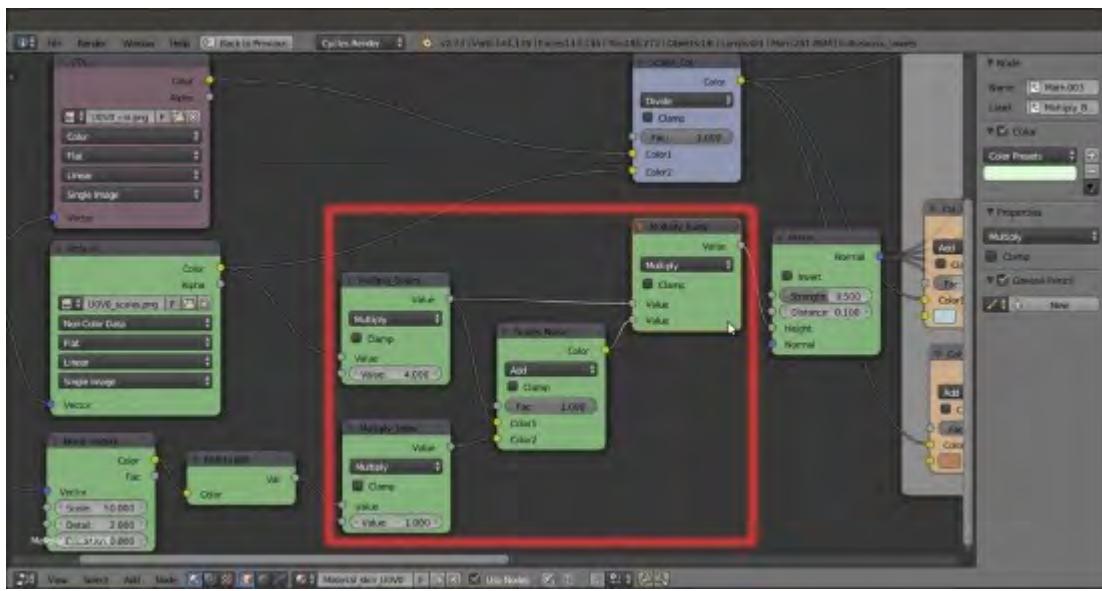
Adding some noise to the bump pattern part 1

43. Select the **Math** node and move it on the link connecting the **Noise Texture** node with the **Scales_Noise** node to paste it in between them: set the **Operation** to **Multiply**, the second **Value** to **1.000**, and label it as **Multiply_Noise**.
44. Press *Shift + D* to duplicate the **Multiply_Noise** node, change the label to **Multiply_Scales** and the second **Value** to **4.000**; paste it between the **SCALES** node and the **Scales_Noise** node.
45. Add an **RGB to BW** node (*Shift + A | Converter | RGB to BW*) and paste it between the **Noise Texture** node and the **Multiply_Noise** one:



Adding some noise to the bump pattern part 2

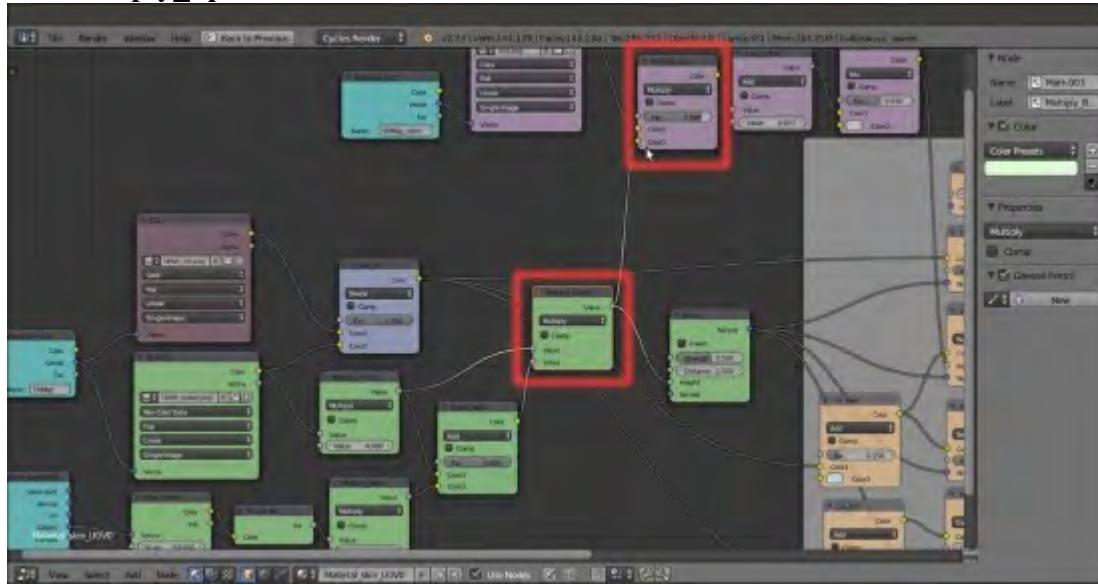
46. Press **Shift + D** to duplicate the **Multiply_Scales** node and change the duplicate label to **Multiply_Bump**; connect the output of the **Multiply_Scales** to the first **Value** input socket of the **Multiply_Bump** node and the output of the **Scales_Noise** node to the second **Value** input socket. Connect the output of the **Multiply_Bump** node to the **Height** input socket of the **Bump** node:



Adding some noise to the bump pattern part 3

47. Add a **MixRGB** node (**Shift + A | Color | MixRGB**) and paste it between the **VCOL** node and the **Spec_soften** node; label it as **Multiply_Spec**, set the **Blend Type** to **Multiply** and the **Fac**

value to **0.850**; connect the output of the **Multiply_Bump** node to the **Color2** input socket of the **Multiply_Spec** node:



Modulating the specularity with the aid of the bump pattern output

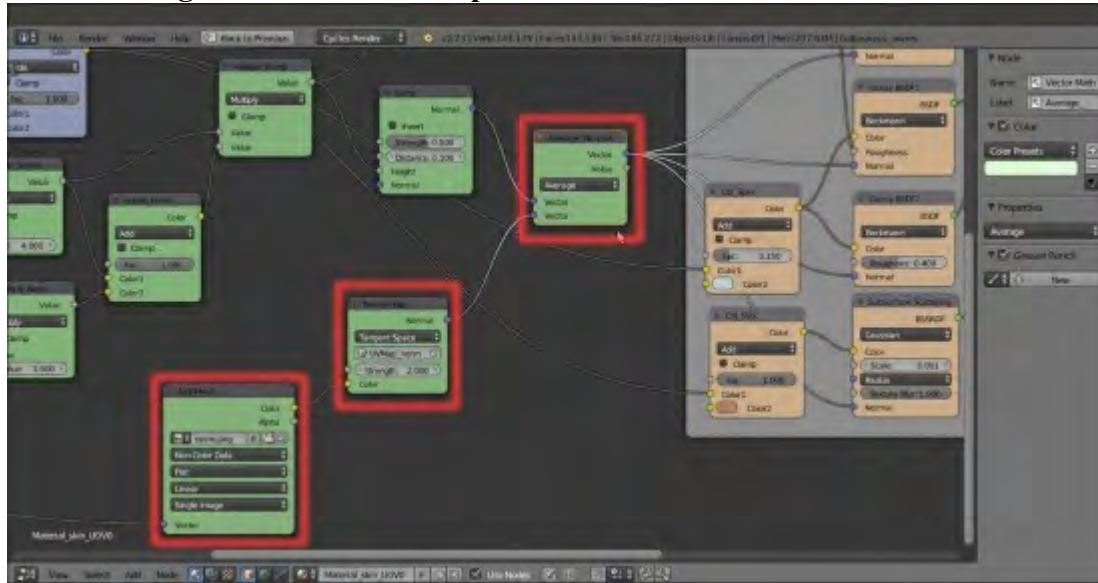
The overall bump effect is almost completed:



The new Rendered bump effect

What is still missing now is the normal map we obtained from the sculpted **Gidiosaurus** mesh in [Chapter 11, Refining the Textures](#).

48. Add a new **Image Texture** node (*Shift + A | Texture | Image Texture*) and a **Normal Map** node (*Shift + A | Vector | Normal Map*). Label the **Image Texture** node as **NORMALS**, then connect the **Vector** output of the **Attribute_UV2** node to the **Vector** input socket of the **NORMALS** node.
49. Connect the **Color** output of the **NORMALS** node to the **Color** input socket of the **Normal Map** node, then click on the **Open** button on the **NORMALS** node, browse to the **textures** folder and load the image **norm.png**. Set the **Color Space** of the **NORMALS** node to **Non-Color Data** and click on the empty slot in the **Normal Map** node to select the **UVMap_norm** coordinates layer.
50. Add a **Vector Math** node (*Shift + A | Converter | Vector Math*), label it as **Average_Normals** and paste it right after the **Bump** node; connect the output of the **Normal Map** node to the second **Value** input socket of the **Average_Normals** node.
51. Set the **Operation** of the **Average_Normals** node to **Average** and connect its **Vector** output to the **Vector** input sockets of the **Diffuse BSDF**, **Glossy BSDF1**, **Glossy BSDF2**, and **Subsurface Scattering** nodes.
52. Set the **Strength** of the **Normal Map** to **2.000**:



Adding the normal map output to the bump pattern

Finally we have completed the first skin material!

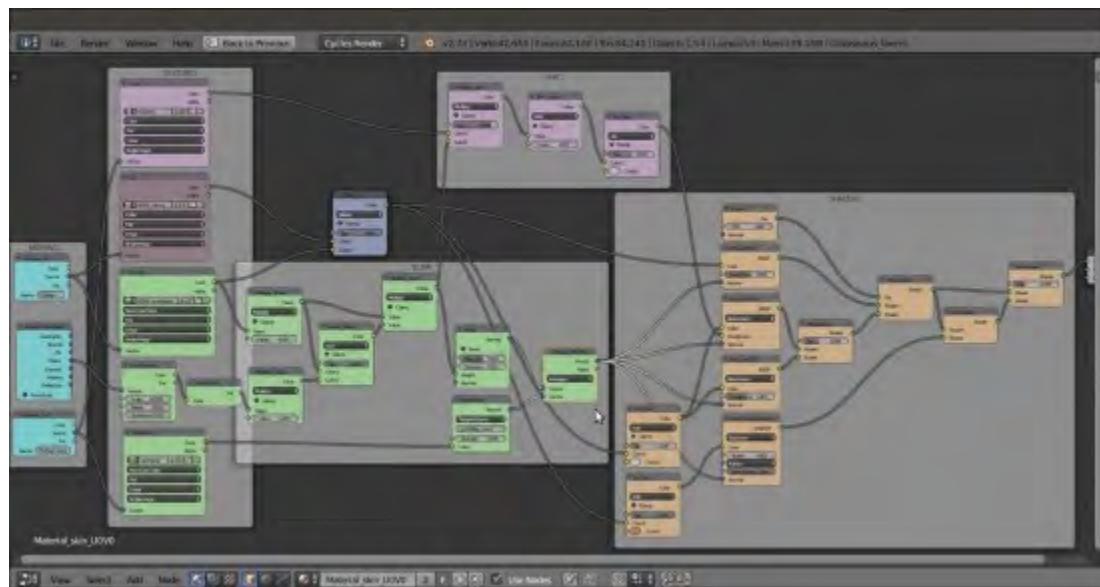


The completed Material_skin_U0V0

53. Save the file as Gidiosaurus_skin_Cycles.blend.

How it works...

This material can at first glance appear a bit complex, but actually the design behind it is quite simple as you can see in the following screenshot, where each component has been visually grouped by colors and frames (open the provided Gidiosaurus_skin_Cycles_01.blend file to have a better look):



The total skin material network

- From step 1 to step 18 we built the **SHADERS** part of the material, that is, the combination of the diffuse with the glossy component and the addition of the subsurface scattering effect.
- Note that the glossy component (the *specularity*) is obtained by mixing **two** glossy shaders with different roughness values; by setting the factor value of the **Mix Shader1** node to **0.350**, we give prevalence to the **Glossy BSDF1** node effect, which is to the node connected to the first top **Shader** input socket.
- Also, we added the subsurface scattering effect by the **Add Shader** node, and to further tweak the blending of the effect with the rest of the shader, we added the **Mix Shader3** node, to give prevalence to the output of the **Mix Shader2** node (that is the output of the diffuse plus the glossy components).
- From step 19 to step 24 we saw some not mandatory but useful tips for assigning colors to the nodes, in order to visually distinguish and/or group them and make the whole material network more easily readable.
- At step 25 we started to add the textures, first the diffuse color one and then the grayscale scales image that we used here to add details to the coloration (and later for the bump effect). By mixing the scales with the diffuse color through the **MixRGB** node set to a **Divide** blend type, we automatically obtained a scales pattern on the skin itself.
- From step 30 to step 33 we tweaked the diffuse color map to also affect the glossy and the subsurface scattering components, but with different hues.
- Note that at step 34 we used an **Attribute** node to set the UV coordinates layer to be used for the mapping of the textures. It would have been unnecessary in this case, with the **UVMap** coordinates layer being the first one and therefore the default one. Cycles, in fact, in the case of image textures, automatically uses any existing UV coordinates layer. But, because later we also used a different UV coordinates layer, it was better to specify it.
- From step 35 to step 38 we improved the glossiness effect of the skin, by using the output of the `vcol.png` image we had previously baked and tweaked through the nodes inside the **SPEC** frame.
- From step 39 to step 47 we built the **BUMP** effect, by using the output of the **SCALES** image texture added through a **MixRGB** node to the output of a procedural **Noise Texture**. The **RGB to BW** node simply converts the colored output of the procedural noise to a grayscale output (and if you think we could have used the **Fac** output instead, well, it's not the same thing), and the **Multiply_Scales** and **Multiply_Noise** nodes set the strength of the outputs before the adding process. Through the **Multiply_Bump** node we also added the grayscale output of the combined bump to the glossy component.
- From step 48 to step 52 we also added the effect of the normal map we baked from the sculpted high resolution **Gidiosaurus** mesh to the bump pattern. The normal map is averaged, through the **Vector Math** node, with the bump output. Because of this averaging, the strength value of the normal map had to be set to double (**2.000**) to have full effect.

There's more...

Still focusing on the character's **head**, there is a material we can obtain from the skin material with some modification, the material for the wet parts of the character's skin (inner **eyelids**, **tongue**, inner **nostrils**).

Going on from the previously saved file:

1. If you think this is the case, especially if your computer (like mine) isn't very powerful, temporarily disable the **Rendered** preview by moving the mouse cursor inside the 3D viewport and pressing *Shift + Z*.
2. In the **Material** window, click on the **Material_wet_U0V0** material to select it.
3. Put the mouse pointer inside the **Node Editor** window, select the default two nodes already assigned to the material and delete them by pressing the *X* key.
4. Now, in the **Material** window, re-select the **Material_skin_U0V0**; put the mouse in the **Node Editor** window, press *A* twice to select everything, and press *Ctrl + C*.
5. Re-select the **Material_wet_U0V0**, put the mouse pointer inside the empty **Node Editor** window and press *Ctrl + V* to paste the copied material nodes.

Now we have copied the nodes of the skin material to the material assigned to the parts that need to appear wet; it's enough now to tweak this material a bit to modify the bump pattern and the glossiness:

6. In the **Node Editor**, zoom to the **Noise Texture** node inside the **BUMP** frame; left-click on it to select it and then press the *X* key to delete it.
7. Press *Shift + A* and add a **Voronoi Texture** node (*Shift + A | Texture | Voronoi Texture*); left-click on the node and, by keeping the mouse button pressed, move the node a little bit on the frame, so it should automatically be parented to it.
8. Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Voronoi Texture** node and the **Color** output of this latter node to the **RGB to BW** node input socket; set the **Voronoi Scale** to **200.000**.
9. Add an **Invert** node (*Shift + A | Color | Invert*) and paste it between the **Voronoi Texture** and the **RGB to BW** nodes:



The different texture nodes of the "Material_wet_U0V0"

10. Scroll the **Node Editor** window a bit to the right to find the **Multiply_Noise** node: change the label to **Multiply_Voronoi** and the second **Value** to **0.025**.
11. Find the **Scales_Col** node and change **Blend Type** from **Divide** to **Multiply**.

- Now go to the **SHADERS** frame; change the **IOR** value of the **Fresnel** node to **15.000** and connect its output to the **Fac** input socket of the **Mix Shader1** node; change the **Distribution** of both the **Glossy BSDF1** and **Glossy BSDF2** nodes to **Ashikhmin-Shirley** and set the **Roughness** of the **Glossy BSDF2** node to **0.600**.

We substituted the **Noise Texture** node with a **Voronoi Texture** node to give a kind of organic look to the surface of the **tongue** of the creature.

In the following screenshot, we can see the result of the wet material; note that for the occasion I opened the mouth wide, to make the inside more visible:

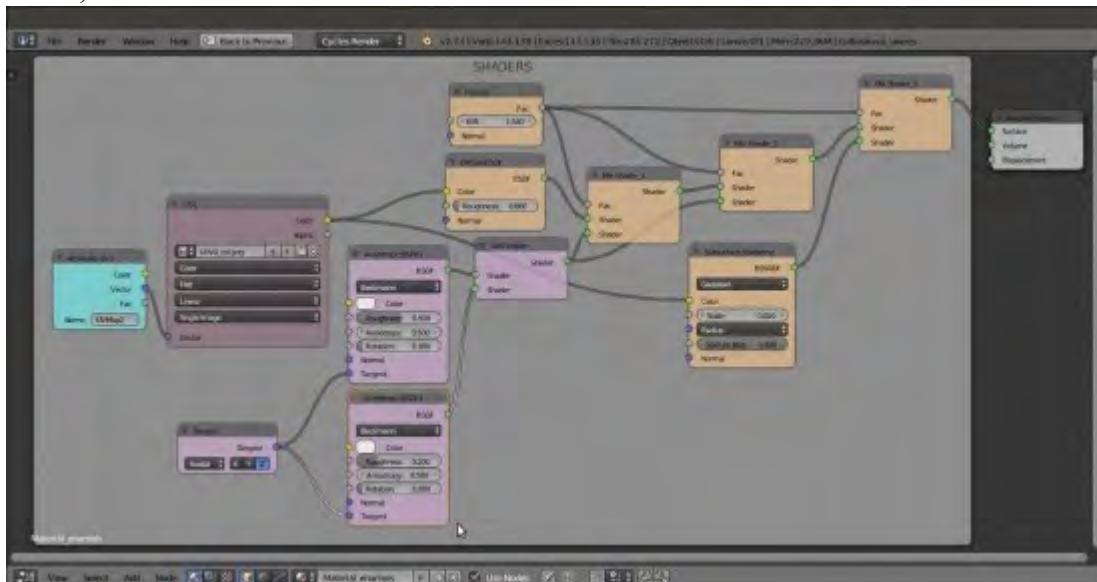


The rendered wet material

One more material we are going to create in this section of the recipe is the **Material_enamels** for **teeth** and **talons**; in this case, we just need mostly the **SHADERS** frame's nodes with the single contribution of the color image texture **U0V0_col.png**, here using the **UVMap2** coordinates layer to avoid having to create **5** different materials for the **talons** alone (originally distributed in different tiles). By the way, nothing is stopping you from creating several talon materials, if you prefer.

- Again, select, copy and paste the skin material to the enamels material slot through the **Node Editor** window, as we have already done in steps 3, 4 and 5.
- This time, just delete the unnecessary nodes, in short keeping only the **Attribute** node, the **COL** node and the **SHADERS** frame with its parented nodes.
- Change the UV coordinates layer in the **Name** slot of the **Attribute** node to **UVMap2** (and the label to **Attribute_UV3**). Lower the **Roughness** value of the **Diffuse BSDF** node to **0.000**.
- Go to the **SHADERS** frame; select and delete the **Col_Spec** and **Col_SSS** nodes, then connect the **Color** output of the **COL** node also to the **Color** input socket of the **Subsurface Scattering** node.
- Select and delete the **Glossy BSDF1** and the **Glossy BSDF2** nodes.

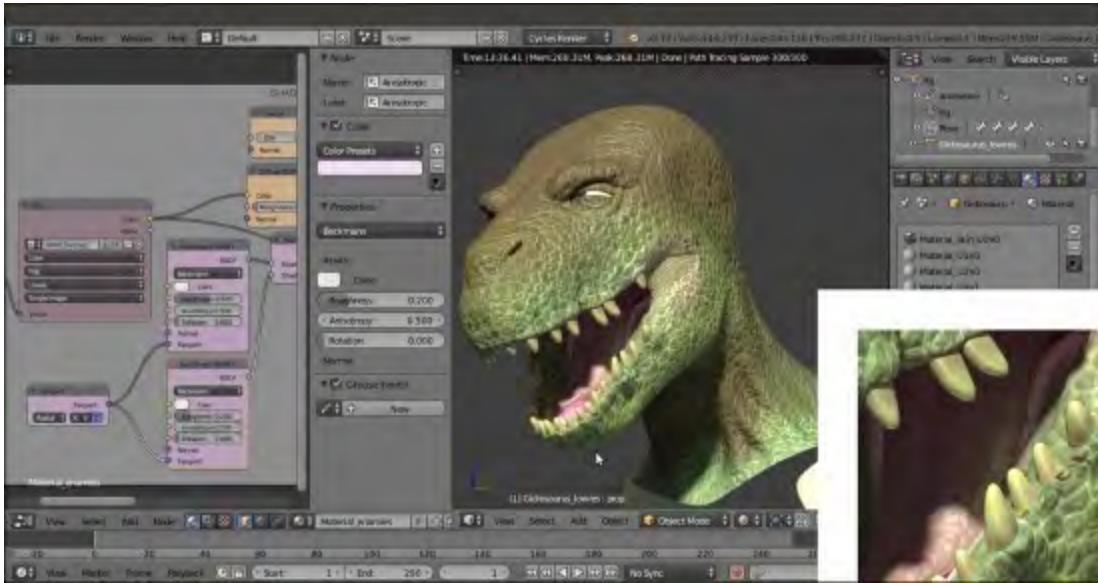
18. Add 2 **Anisotropic BSDF** shader nodes (**Shift + A | Shader | Anisotropic BSDF**), a **Tangent** node (**Shift + A | Input | Tangent**) and detach the **Add Shader** node from the **Mix Shader3** node.
19. Label the two **Anisotropic BSDF** shader nodes as **Anisotropic BSDF1** and **Anisotropic BSDF2** and connect them to the two **Shader** input sockets of the **Add Shader** node. Connect the output of the **Tangent** node to the **Tangent** input sockets of the two **Anisotropic** shader nodes.
20. Set the **Tangent** of the **Tangent** node to **Z**. Set the **Anisotropy** of both the **Anisotropic** nodes to **0.500**, the **Roughness** of the **Anisotropic BSDF1** node to **0.500** and the **Roughness** of the **Anisotropic BSDF2** node to **0.200**.
21. Connect the **Add Shader** output to both the second **Shader** input sockets of the **Mix Shader1** and **Mix Shader2** nodes.
22. Set the **IOR** value of the **Fresnel** node to **1.540** and connect the **Fresnel** output to the **Fac** input sockets of the **Mix Shader1**, **Mix Shader2**, and **Mix Shader3** nodes.
23. Connect the output of the **Diffuse BSDF** shader node to the first **Shader** input socket of the **Mix Shader1** node, then connect the output of the **Mix Shader1** node to the first **Shader** input socket of the **Mix Shader2** node.
24. Connect the output of the **Subsurface Scattering** node to the second **Shader** input socket of the **Mix Shader3** node.
25. In the **Subsurface Scattering** node, change the **Scale** to **0.020** and the **Radius** to **R 1.000, G 0.400, B 0.100**.



The "Material_enamels" network

26. Save the file.

Thanks to the two **Anisotropic** shaders with their different roughness values, we obtained a nice specularity effect along the length of the **teeth** (and therefore also of the **talons**):



The rendered preview of the teeth (and talons) shader

See also

- Shameless self-promotion—one other cookbook, published by Packt Publishing, explaining the logic behind Cycles materials and textures and with several material recipes (<https://www.packtpub.com/hardware-and-creative/blender-cycles-materials-and-textures-cookbook-third-edition>)
- The online documentation (<http://wiki.blender.org/index.php/Doc:2.6/Manual/Render/Cycles/Nodes>)

Making a node group of the skin shader to reuse it

Once we are satisfied with the reptile skin shader created for the character's **head**, we can copy it to the other parts of the **body**, that is to the other material slots, and then apply the necessary modifications. Those, in this case, just consist of different color and scales image textures.

This means that all the other shader parts can be reused as they are. In this recipe, in fact, we are going to make a node group of these parts so as to easily re-use the shader for the other materials slots.

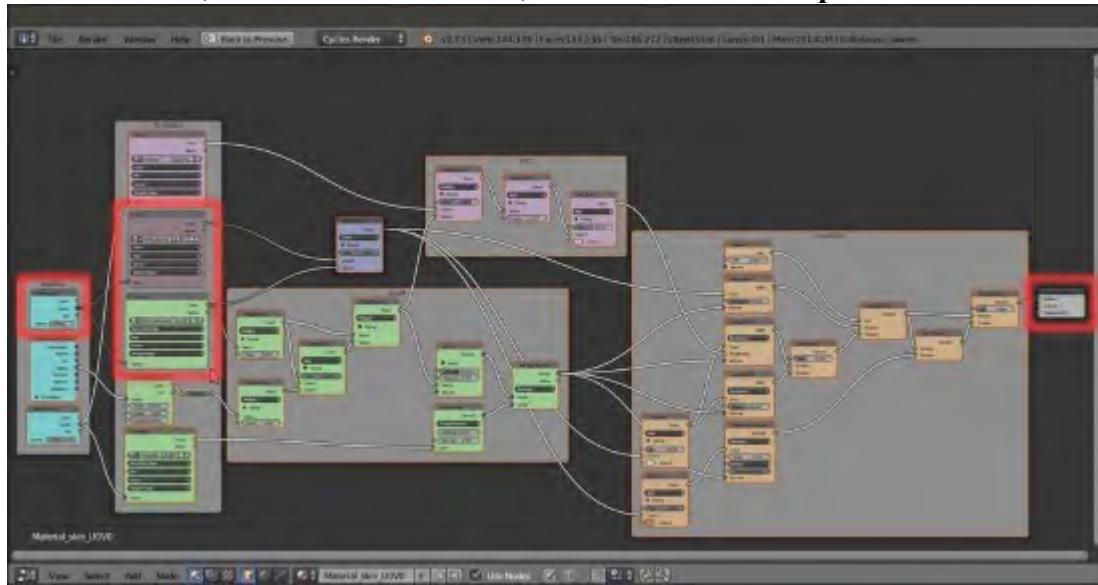
Getting ready

Just start Blender and re-open the previously saved `Gidiosaurus_skin_Cycles_01.blend` file.

How to do it...

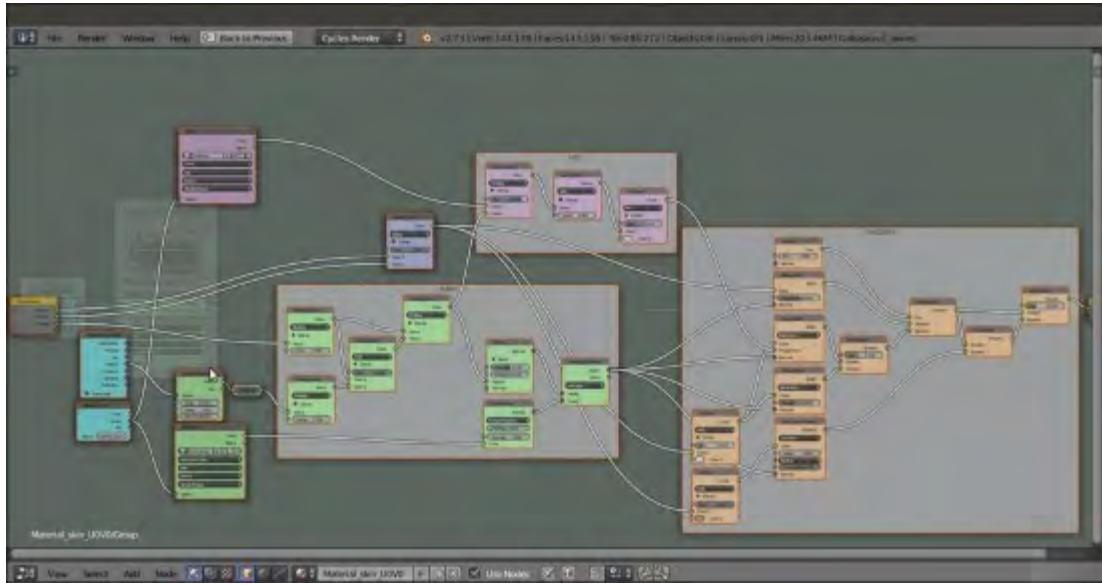
Let's start to create our skin node group:

1. In the **Material** window, select the slot of the `Material_skin_U0V0`.
2. Put the mouse pointer in the **Node Editor** window, press the *B* key and left-click to box-select all the nodes with their respective frames. Then, press the *Shift* key and right-click (twice for each one) to deselect the **Attribute_UV1** node, the **MAPPING** frame, the **COL** node, the **SCALES** node, the **TEXTURES** frame, and the **Material Output** node:



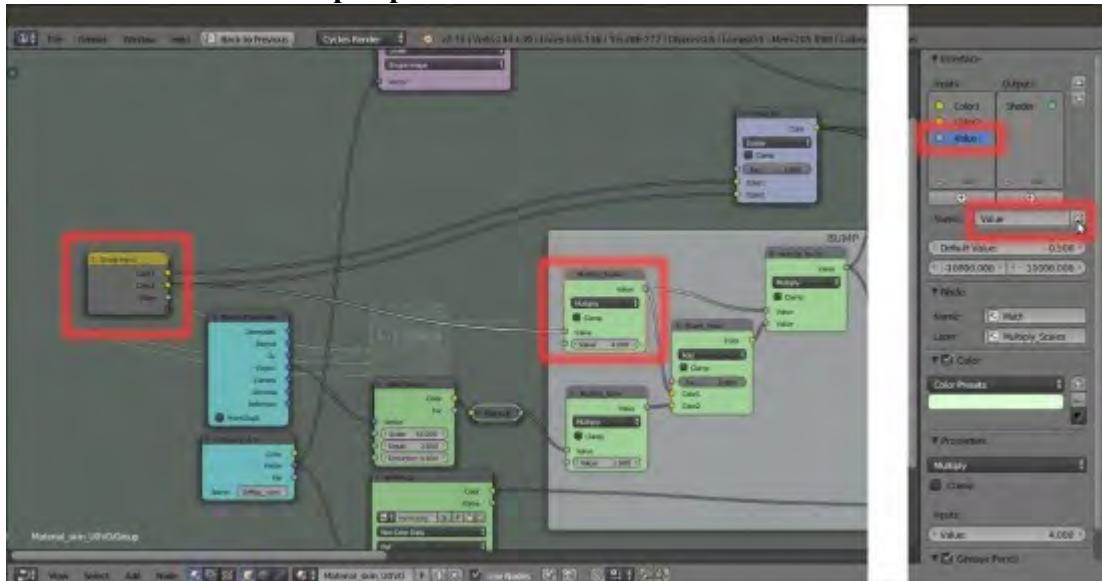
The box-selected nodes and the highlighted deselected ones

3. Press *Ctrl + G* to make a node group of the selected nodes; automatically you are inside the group in **Edit Mode**:



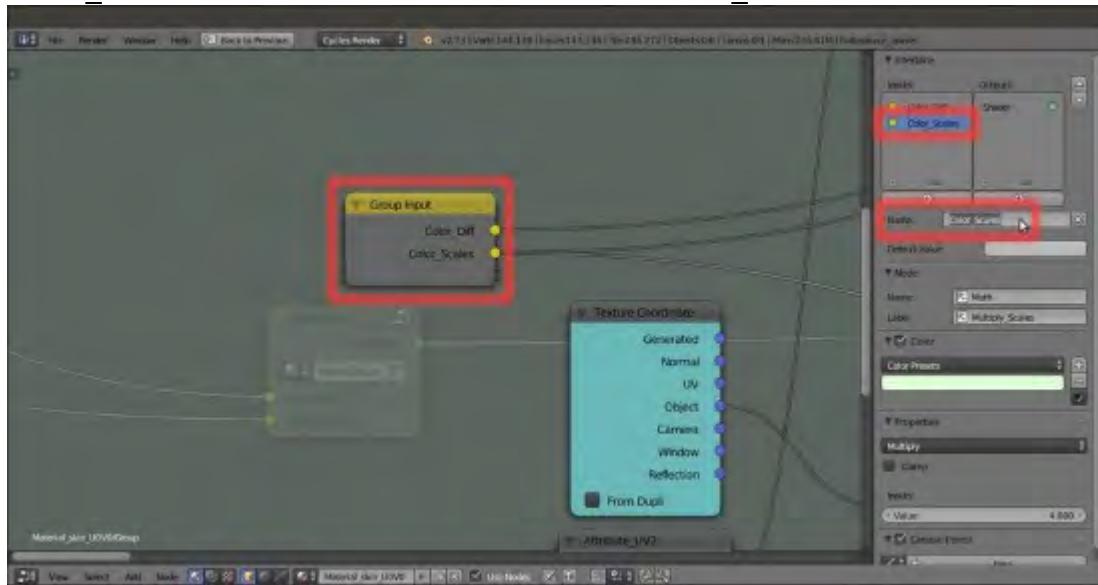
Inside the node group in Edit Mode

4. Click on the **Group Input** node to select it and zoom in on it, then press **N** to call the **Properties** sidepanel. Connect the **Color2** output socket to the first **Value** input socket of the **Multiply_Scales** node, replacing the connection coming from the **Value** output.
5. Go to the **Properties** sidepanel and in the **Interface** subpanel, click on the **Value** item inside the little **Inputs** window; then go down to the **Name** slot and click on the **X** icon button to delete the socket from the **Group Input** node:



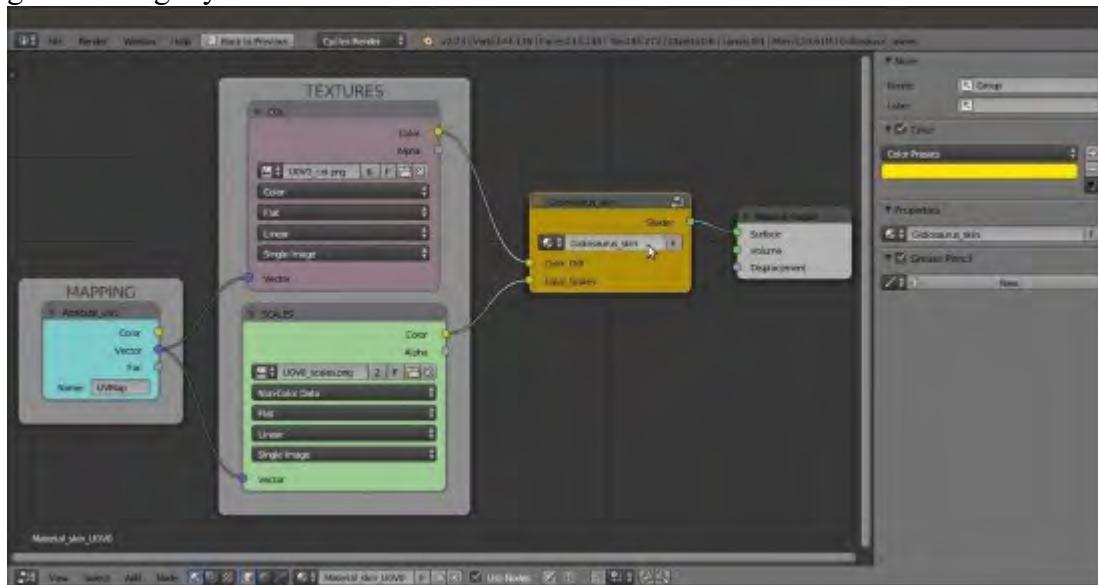
Tweaking the node group input socket connections

6. Still in the **Name** slot in the **Interface** subpanel, select the **Color1** item and rename it **Color_Diff**. Select the **Color2** item and rename it **Color_Scales**:



Renaming the node group input sockets

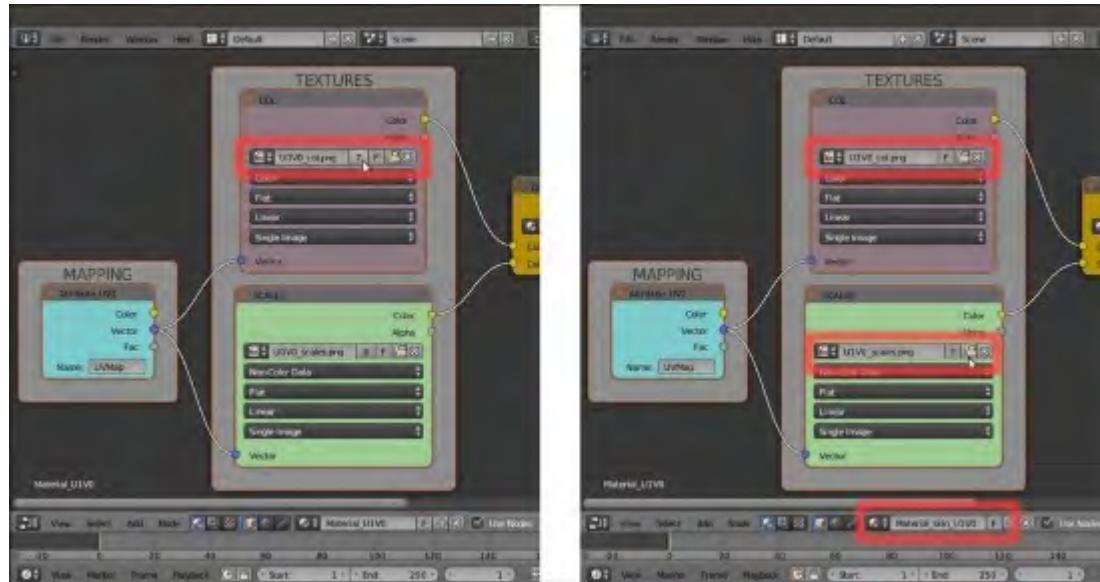
7. Press *Tab* to exit out of **Edit Mode** and close the node group; rename it **Gidiosaurus_skin** and give it a bright yellow color:



The "Gidiosaurus_skin" node group

8. Now select everything by pressing the *A* key twice and then press *Ctrl + C*.
9. In the **Material** window, select the **Material_U1V0**, then click on the **Use Nodes** button in the **Surface** subpanel.

10. Put the mouse pointer in the **Node Editor** window and delete the already selected default nodes (the **Diffuse BSDF** connected to the **Material Output** nodes), then press *Ctrl + V* to paste all the copied nodes inside the window.
11. Zoom on the **COL** and **SCALES** nodes. Click on the numbered button to the right side of the texture name to make them single users, then click on the folder icon buttons to browse to the textures folder and load the proper images according to the material name, that is, the **U1V0_col.png** and **U1V0_scales.png** image textures.
12. Rename the material as **Material_skin_U1V0**:



Making the copied textures single users and loading the right image textures for the "Material_skin_U1V0"

13. Repeat step 8 to step 12 for the other remaining **3** material slots and then save the file as **Gidiosaurus_skin_Cycles_02.blend**.



The "*Material_wet_UOV0*" network and the completed Gidiosaurus shading in the rendered preview

How it works...

Of course, it wasn't mandatory to make a group of the skin shader to reuse it for the other material slots; we could just have selected, copied and pasted all the nodes and frames as they were at the end of the previous recipe.

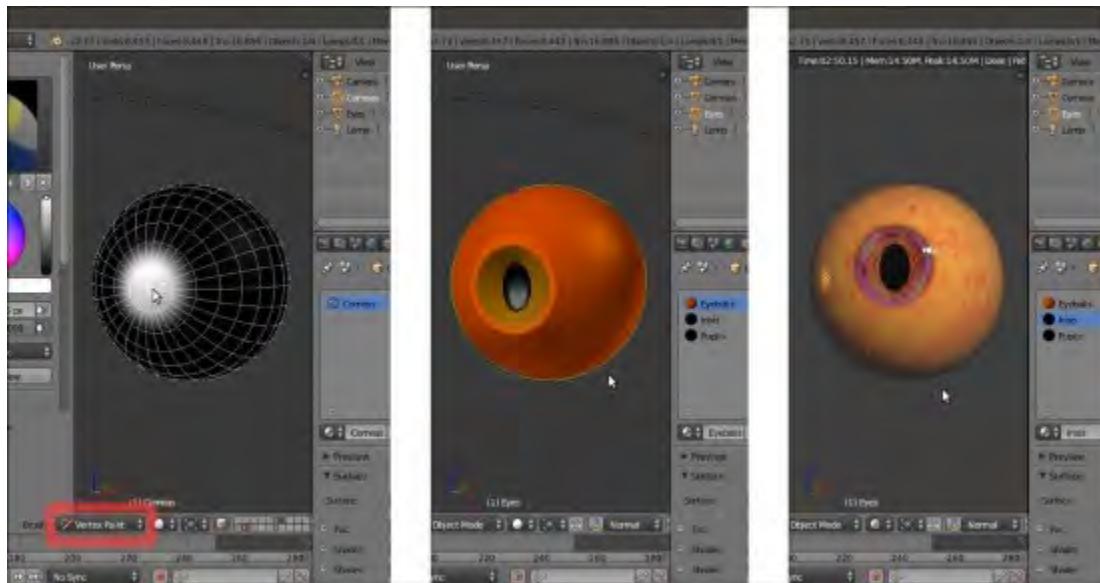
The (quite big) advantage in having a node group instanced in different materials is that if you need to change something in the internal network, you don't have to repeat the modifications in the node group of each material. It's enough to do it in **Edit Mode** in one of the instances, and all the internal modifications will be reflected in all the instances of the node group used by the other materials.

Building the eyes' shaders in Cycles

The character's eyes are made up of two **UV Spheres**, the **Corneas** and the **Eyes** objects: the bigger **Corneas** one enveloping a smaller **Eyes** sphere, which in turn is made up of **three** parts: the **eyeballs**, the **irises**, and the **pupils**.

The **Corneas** sphere was first painted with a totally black **Vertex Color** layer, then painted with a white color only to the vertices corresponding to the front crystalline lens.

The **Eyes** sphere has **three** different materials assigned to the **three** different parts:



The Corneas object in Vertex Paint mode, the Eyes object with its three materials and the Rendered preview of the textured objects together

Getting ready

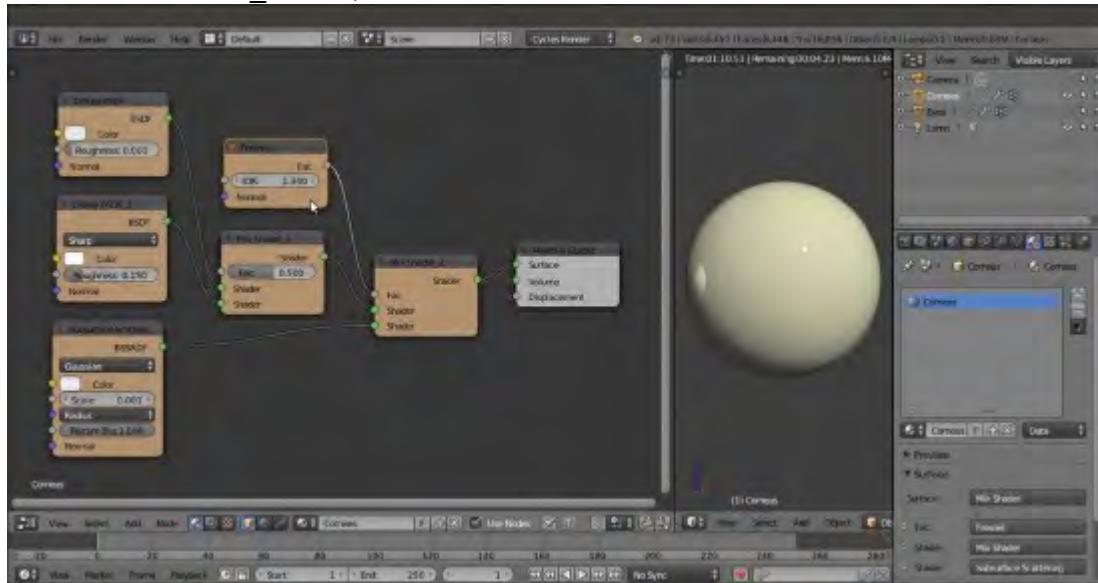
Start Blender and open the `Gidiosaurus_skin_Cycle_02.blend` file; save it as `Gidiosaurus_shaders_Cycles.blend`.

1. Enable only the **6th** and the **12th** scene layer, in order to have visible only the **Corneas**, the **Eyes** and the **Lamp** objects (actually the **Camera** is also on the **6th** scene layer, but it's hidden and at the moment we don't need it).
2. Zoom the 3D view onto the **Corneas** and **Eyes** objects, and press **Shift + Z** to start the **Rendered** preview.
3. In the **Outliner**, disable the *Restrict view-port visibility* button of the **Eyes** object to hide it.
4. Select the **Corneas** object and go to the **Material** window.

How to do it...

So, let's start with the **Corneas** material, first:

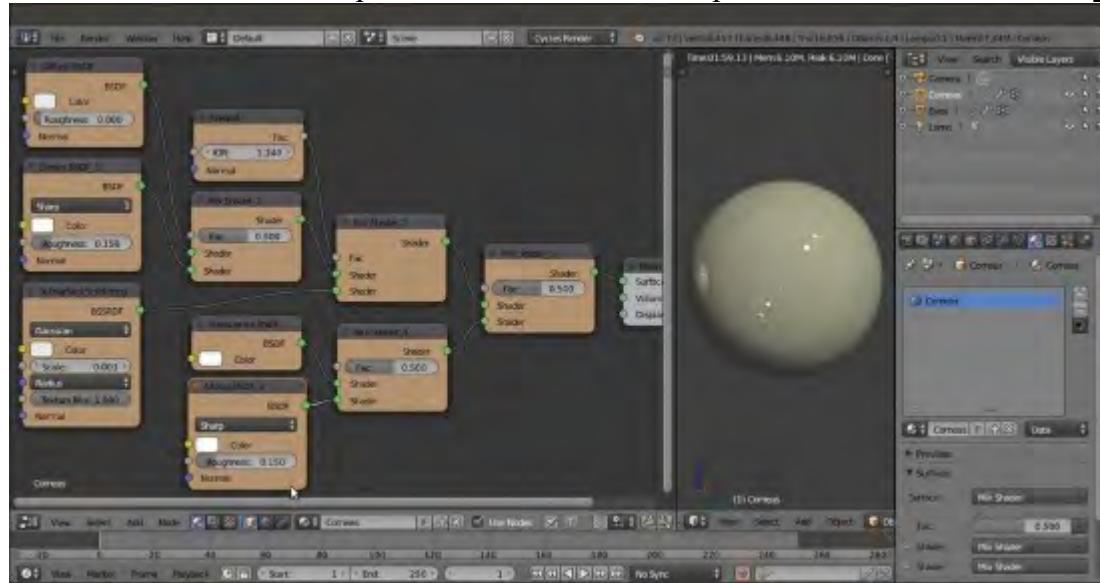
1. In the **Material** window, click on the **New** button in the **Surface** subpanel. Rename the material as **Corneas**.
2. In the **Material** window, switch the **Diffuse BSDF** shader node with a **Mix Shader** node (label it as **Mix Shader_1**). In the first **Shader** slot, select a **Diffuse BSDF** shader node and in the second one select a **Glossy BSDF** shader (label it as **Glossy BSDF1**).
3. Go to the **Node Editor** window and set the **Roughness** of the **Glossy BSDF1** shader to **0.150**, the **Color** to pure white and the **Distribution** to **Sharp**.
4. Select the **Mix Shader_1** node and press **Shift + D** to duplicate it. Label the duplicate as **Mix Shader_2**, then add a **Subsurface Scattering** node (**Shift + A | Shader | Subsurface Scattering**). Connect the output of the **Mix Shader_1** node to the first **Shader** input socket of the **Mix Shader_2** node and the output of the **Subsurface Scattering** shader node to the second **Shader** input socket.
5. Change the **Subsurface Scattering** falloff from **Cubic** to **Gaussian**, set the **Scale** to **0.001** and the **Radius** to **R 9.436, G 3.348, B 1.790**.
6. Add a **Fresnel** node (**Shift + A | Input | Fresnel**) and connect its output to the **Fac** input socket of the **Mix Shader_2** node; set the **IOR** to **1.340**:



The basic starting "Corneas" shader

7. Add a new **Mix Shader** node (**Shift + A | Shader | Mix Shader**), label it as **Mix Shader_3** and paste it between the **Mix Shader_2** and the **Material Output** node.
8. Add a new **Mix Shader** node (**Shift + A | Shader | Mix Shader**), label it as **Mix Shader_4** and connect its output to the second **Shader** input socket of the **Mix Shader_3** node.
9. Add a **Transparent BSDF** shader (**Shift + A | Shader | Transparent BSDF**) and connect it to the first **Shader** input socket of the **Mix Shader_4** node.

- Select and press **Shift + D** to duplicate the **Glossy BSDF1** node; label the duplicate as **Glossy BSDF2** and connect its output to the second **Shader** input socket of the **Mix Shader_4** node:



The "Corneas" shader with the added transparency nodes

- Add a **Layer Weight** node (**Shift + A | Input | Layer Weight**) and a **Math** node (**Shift + A | Converter | Math**); set the **Blend** factor of the **Layer Weight** to **0.300** and connect its **Facing** output to the first **Value** input socket of the **Math** node, then set the second **Value** to **0.100** and check the **Clamp** item.
- Connect the **Math** (labeled as **Add**) output to the **Fac** input sockets of the **Mix Shader_1** and **Mix Shader_4** nodes.
- Add an **Attribute** node (**Shift + A | Input | Attribute**) and a **ColorRamp** node (**Shift + A | Converter | ColorRamp**). In the **Name** slot of the **Attribute** node, type **Col**, then connect its **Color** output to the **Fac** input socket of the **ColorRamp** node.
- In the **ColorRamp** node, set the **Interpolation** to **B-Spline** and move the white color stop to position **0.100**. Connect its **Color** output to the **Fac** input socket of the **Mix Shader_3** node.



The "Corneas" shader with the transparency area located by the Vertex Color layer

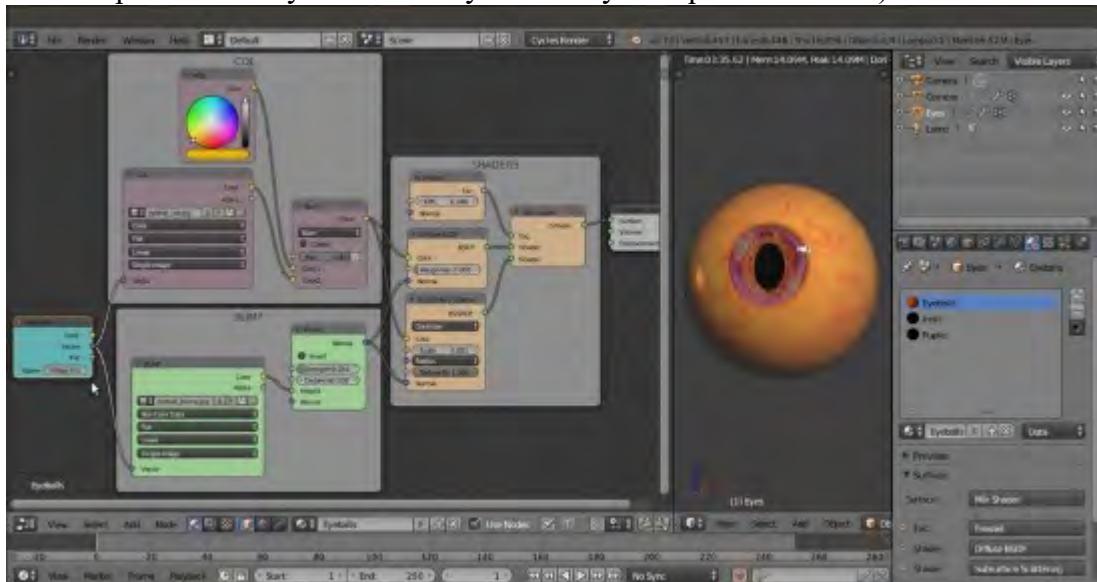
15. Add two **Image Textures** nodes (*Shift + A | Texture | Image Texture*) and label them respectively as **COL** and **BUMP**.
16. Add an **Attribute** node (*Shift + A | Input | Attribute*); in the **Name** slot type **UVMap.001** and connect its **Vector** output to the **Vector** input sockets of the two image texture nodes.
17. Add an **RGB** node (*Shift + A | Input | RGB*), a **MixRGB** node (*Shift + A | Input | MixRGB*) and a **Hue Saturation Value** node (*Shift + A | Input | Hue/Saturation*).
18. Click on the **Open** button of the **COL** node to browse to the **textures** folder and load the image **eyeball_col.jpg**.
19. Connect the **Color** output of the **COL** node to the **Color2** input socket of the **MixRGB** node and the output of the **RGB** node to the **Color1** input socket; set the **Blend Type** to **Burn** and the **Fac** value to **0.800**.
20. Connect the **Color** output of the **MixRGB** node to the **Color** input socket of the **Hue Saturation Value** node, and the output of this latter node to the **Color** input sockets of the **Diffuse BSDF** and **Subsurface Scattering** nodes.
21. Set the **RGB** node color to **R 0.800, G 0.466, B 0.000**; set the **Saturation** value of the **Hue Saturation Value** node to **0.900**.
22. Click on the **Open** button of the **BUMP** node to browse to the **textures** folder and load the image **eyeball_bump.jpg**; set **Color Space** to **Non-Color Data**.
23. Add a **Bump** node (*Shift + A | Vector | Bump*) and connect the **Color** output of the **BUMP** node to the **Height** input socket of the **Bump** node. Connect the **Normal** output of this latter node to the **Normal** input sockets of the **Diffuse BSDF**, **Glossy BSDF1**, and **Subsurface Scattering** nodes, and set the **Strength** to **0.050**.
24. If you wish, add frames and colors to the different components to make the shader more easily readable:



The textured "Corneas" material

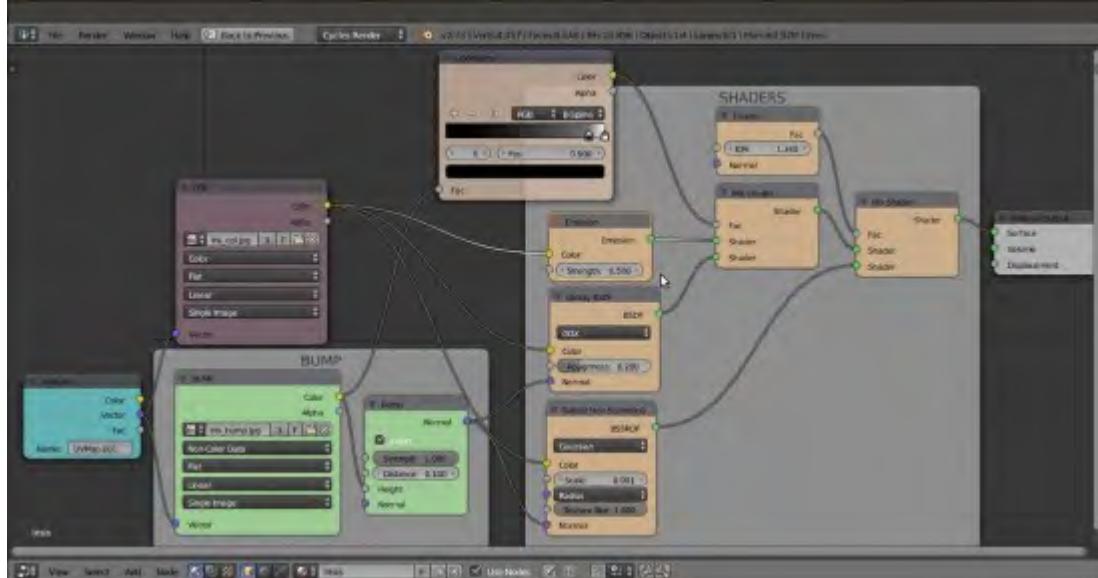
Now let's quickly see the materials for the Eyes object:

- As you can see in the following screenshot, the **Eyeballs** material is essentially the same as we just made for the **Corneas** except for the transparent part; this material, by the way, is obsolete because it's hidden behind the **Corneas**' opaque surface, so can be safely omitted (but I left it in place in case you want to try the totally transparent **Cornea**):



The "Eyeballs" material and the completed rendered eye

- The **Irides** material follows the same scheme; the only differences are in the fact that it uses different image textures (`iris_col.jpg` and `iris_bump.jpg`) and that a contrasted (by a **ColorRamp** node) version of the bump image is used as a factor for the mixing of an **Emission** shader; note that the color map is also connected to this **Emission** shader:



The "Irides" material network

- The Pupils are a simple, basic, black diffuse material.

To have a look at these materials, open the `Gidiosaurus_shaders_Cycles.blend` file and select the **Corneas** and **Eyes** objects in the **Outliner**.

How it works...

These shaders are quite simple; the more complex one is the shader for the **Corneas**, essentially because it's made up of **two** materials, one with a slight bump effect and one totally smooth, mixed on the ground of the black and white **Vertex Color** layer that takes care also of the distribution of the transparent and opaque materials on the **Corneas** object itself.

If you are wondering why we didn't use the **Eyeball** material on the underlying **Eyes** sphere, leaving the **Corneas** object totally transparent, the reason is simple: in Cycles, to have a material transparent but also reflecting the environment, you need to use a **Transparent** shader mixed with a **Glass** or a **Glossy** shader node, that inevitably will make whatever material is behind appear darker; sometimes this can look right, in this case I preferred to use a different approach.

Note

Note that the transparent part in front of the iris of the cornea, to be anatomically correct, should be a convex, bulging half sphere; instead, we modeled the cornea as a simple spherical sheath around the eyeball to avoid complications with the open/closed movements of the eyelids.

Building the armor shaders in Cycles

The last thing to do, for this chapter, is to create the shaders for the **Armor** object, made up of metallic plates and leather **tiers**.

Getting ready

Continuing from the previously saved blend file:

1. Enable the **6th** and the **13th** scene layer and select the **Armor** object in the **Outliner**.
2. Put the mouse pointer in the 3D viewport and press the **0** key on the numpad to go into **Camera** view; fit the window into the field of view.
3. Go to the **Material** window and press the **+** icon button to the right side to add **four** empty material slots to the **armor**. Select the first material slot and click on the **New** button in the **Surface** subpanel, and rename the material **Armor_U0V0**.
4. Select the **second** material slot, click on the **New** button and rename the material as **Armor_U1V0**; repeat for the **third** slot and rename the material as **Leather** and repeat also for the **fourth** slot and rename the material **Armor_rivets**.
5. Switch the **Node Editor** window temporarily with a **UV/Image Editor** window, then press **Tab** to go into **Edit Mode**; go to the **UV Maps** subpanel under the **Object Data** window to be sure you have the **UVMAP** coordinates layer (the first one) as the active one, then enable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar.
6. In the **Node Editor** window, box-select the UV islands of the **U1V0** tile, then in the **Material** window, select the **Armor_U1V0** material and click on the **Assign** button.
7. Still in **Edit Mode**, select all the tiers vertices and then select the **Leather** material, click on the **Assign** button; repeat the operation by selecting all the rivets and assigning them to the **Armor_rivets** material.
8. Disable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar, go out of **Edit Mode** and switch the **UV/Image Editor** window back to the **Node Editor** window.
9. Put the mouse pointer inside the 3D viewport and press **Shift + Z** to start the **Rendered** preview.

How to do it...

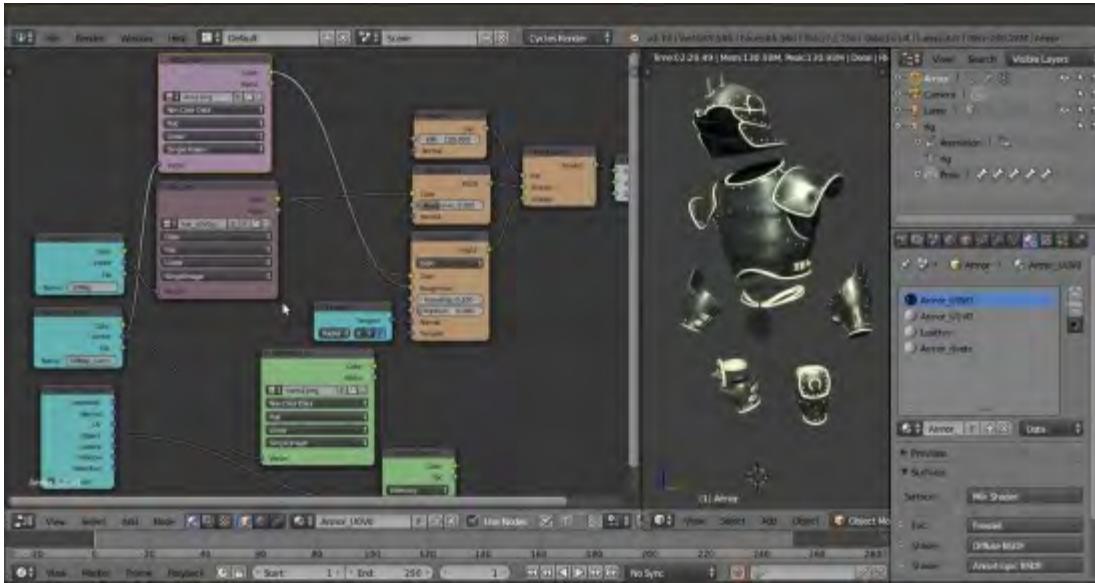
We are first going to create the shader for the metal **plates**:

1. In the **Material** window, select the **Armor_U0V0** material slot.
2. Go to the **Node Editor** window and switch the **Diffuse BSDF** shader node with a **Mix Shader** node; in the first **Shader** slot, select a **Diffuse BSDF** shader node and in the second one, select an **Anisotropic BSDF** shader.
3. Go to the **Node Editor** window and set the **Roughness** of the **Diffuse BSDF** shader to **0.300** and the **Anisotropy** of the **Anisotropic BSDF** shader to **0.300**.
4. Add a **Fresnel** node (**Shift + A | Input | Fresnel**) and connect its output to the **Fac** input socket of the **Mix Shader** node; set the **IOR** to **100.000**.
5. Add a **Tangent** node (**Shift + A | Input | Tangent**) and connect its output to the **Tangent** input socket of the **Anisotropic BSDF** shader node; set the **Tangent** to **Z**.



Starting to build the metal shader for the armor

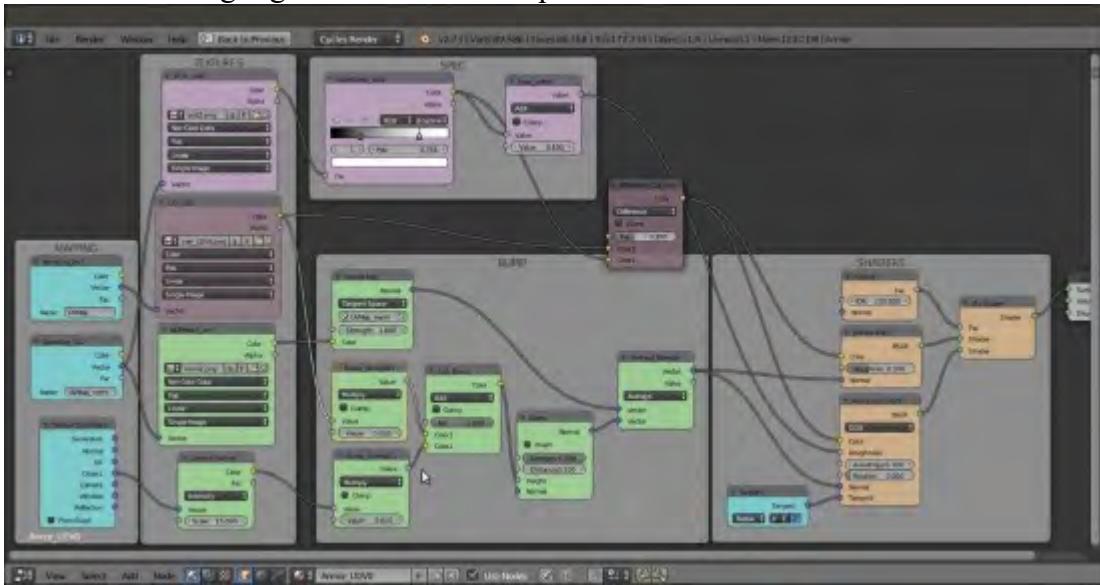
6. Add a **Frame** (*Shift + A | Layout | Frame*) and parent the nodes, except the **Material Output**, to it, then label it as **SHADERS**.
7. Add three **Image Textures** nodes (*Shift + A | Texture | Image Texture*) and a **Voronoi Texture** node (*Shift + A | Texture | Voronoi Texture*), then add two **Attribute** nodes (*Shift + A | Input | Attribute*) and a **Texture Coordinate** node (*Shift + A | Input | Texture Coordinate*).
8. Label the **Attribute** nodes as **Attribute_UV1** and **Attribute_UV2**. Label the **Image Texture** nodes as **COL_iron**, **NORMALS_iron**, and **VCOL_iron**.
9. Connect the **Vector** output of the **Attribute_UV1** node to the **Vector** input socket of the **COL_iron** node. Connect the **Vector** output of the **Attribute_UV2** to the **Vector** input sockets of both the **VCOL_iron** and **NORMALS_iron** nodes. Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Voronoi Texture** node.
10. Click on the **Open** button of the **VCOL_iron** node, browse to the **textures** folder and load the image **vcol2.png**. Set the **Color Space** to **Non-Color Data**. Connect its **Color** output to the **Roughness** input socket of the **Anisotropic BSDF** shader node.
11. Click on the **Open** button of the **COL_iron** node, browse to the **textures** folder and load the image **iron_U0V0.png**. Connect its **Color** output to the **Color** input sockets of the **Diffuse BSDF** and **Anisotropic BSDF** shader nodes.
12. Click on the **Open** button of the **NORMALS_iron** node, browse to the **textures** folder and load the image **norm2.png**. Set the **Color Space** to **Non-Color Data**.
13. Set the **Scale** of the **Voronoi Texture** to **15.000**:



Adding the textures to the "Armor_U0V0" material

14. Add a **ColorRamp** node (*Shift + A | Converter | ColorRamp*) and a **Math** node (*Shift + A | Converter | Math*). Paste the **ColorRamp** node right after the **VCOL** node, and the **Math** node right after the **ColorRamp**.
15. Label the **ColorRamp** as **ColorRamp_Vcol** and set the **Interpolation** to **B-Spline**, then move the black color stop to position **0.245** and the white color stop to position **0.755**.
16. Label the **Math** node as **Spec_soften** and set the second **Value** to **0.100**.
17. Add a **MixRGB** node (*Shift + A | Color | MixRGB*) and label it as **Difference_Col_iron**; set the **Blend Type** to **Difference** and the **Fac** value to **0.300**.
18. Connect the **Color** output of the **COL** node to the **Color1** input socket and the **Color** output of the **ColorRamp_Vcol** node to the **Color2** input socket. Connect the **Color** output of the **Difference_Col_iron** node to the **Color** input sockets of the **Diffuse BSDF** and the **Anisotropic BSDF** shader nodes, replacing the old connections.
19. Add a **Normal Map** node (*Shift + A | Vector | Normal Map*), a **Bump** node (*Shift + A | Vector | Bump*), and a **Vector Math** node (*Shift + A | Converter | Vector Math*).
20. Connect the **Color** output of the **NORMALS_iron** node to the **Color** input socket of the **Normal Map** node; click on the empty slot (*UV Map for tangent space maps*) on this latter node to select the **UVMap_norm** item.
21. Connect the **Normal** output of the **Normal Map** node to the first **Vector** input socket of the **Vector Math** node; label this latter as **Average_Normals** and set the **Operation** to **Average**, then connect its **Vector** output to the **Normal** input sockets of the **Diffuse BSDF** and **Anisotropic BSDF** shader nodes.
22. Add a **MixRGB** node (*Shift + A | Color | MixRGB*), label it as **Add_Bump**, set the **Blend Type** to **Add** and the **Fac** value to **1.000**. Connect the **Color** output of the **COL** node to the **Color1** input socket of the **Add_Bump** node also, and the **Color** output of the **Voronoi Texture** node to the **Color2** input socket.

23. Connect the **Color** output of the **Add_Bump** node to the **Height** input socket of the **Bump** node, and the **Normal** output of this latter node to the second **Vector** input socket of the **Average_Normals** node. Set the **Strength** of the **Bump** node to **1.000**.
24. Add two **Math** nodes (*Shift + A | Converter | Math*), label them respectively as **Bump_strength1** and **Bump_strength2**; set the **Operation** to **Multiply** for both, then paste the **Bump_strength1** node between the **COL_iron** and the **Add_Bump** nodes and set the second **Value** to **0.020**. Paste the **Bump_strength2** node between the **Voronoi_Texture** and the **Add_Bump** nodes, and set the second **Value** to **0.010**.
25. Add frames to highlight the different components:



The completed "Armor_U0V0" material

- The first **Armor** shader is ready! Now it's very easy to obtain the others:
26. Press **A** twice to select all the nodes, then press **Ctrl + C** to copy them.
 27. In the **Material** window, select the **Armor_U1V0** material slot and in the **Node Editor** window, delete the default **Diffuse** and **Material Output** nodes; then press **Ctrl + V** to paste the nodes copied from the other material.
 28. Zoom to the **COL** node and click on the numbered button to the right side of the texture name slot to make it single user, then click on the folder icon button to browse to the texture folder and load the image **iron_U1V0.png**.
 29. Reselect the **Armor_U0V0** material slot and repeat the step 26 and 27, this time pasting the nodes inside the **Armor_rivets** material slot:



The "Armor_rivets" material and the rendered completed armor

30. Save the file.

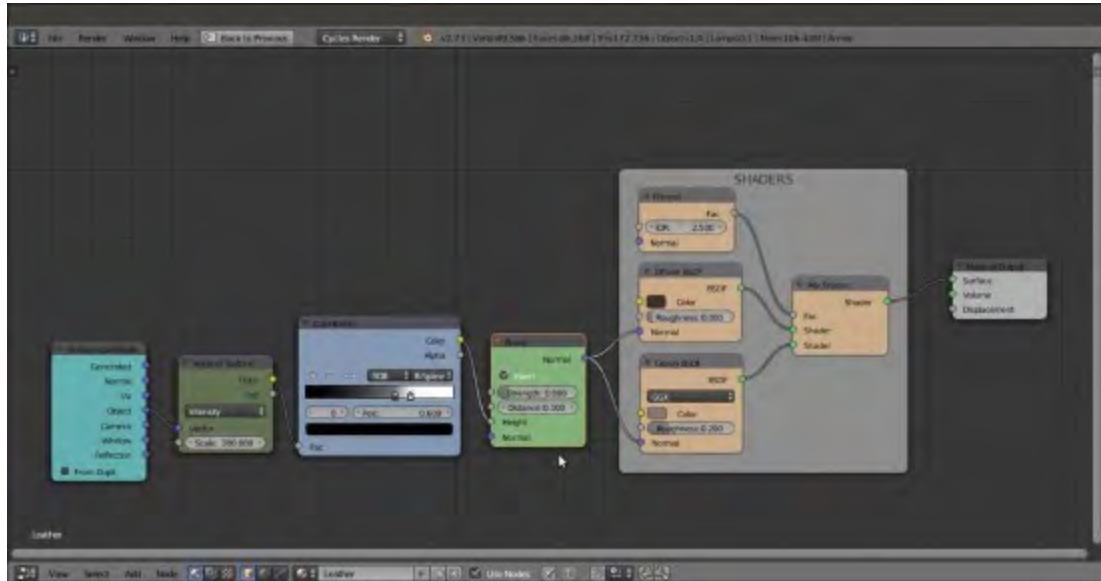
How it works...

The construction of the metallic **armor plates** material follows basically the same scheme we used for the other materials:

- First the shaders were produced, where the metallic look is mainly due to the **Anisotropic BSDF** shader mixed with the diffuse component with a quite high **IOR** value (metals can often have values from **20.000** to **200.000**; we used a midway value of **100.000**).
- The shininess of the metallic surface has been modulated through the output of the **vcol2.png** image, a **Dirty Vertex Color** layer we had previously baked to an image.
- The color of the **Armor** surface has been modulated as well through a **Difference** node with the same **vcol2.png** image.
- The bump pattern works by first adding the **Voronoi** and the **color map** output and then averaging the result with the **normal map** output.

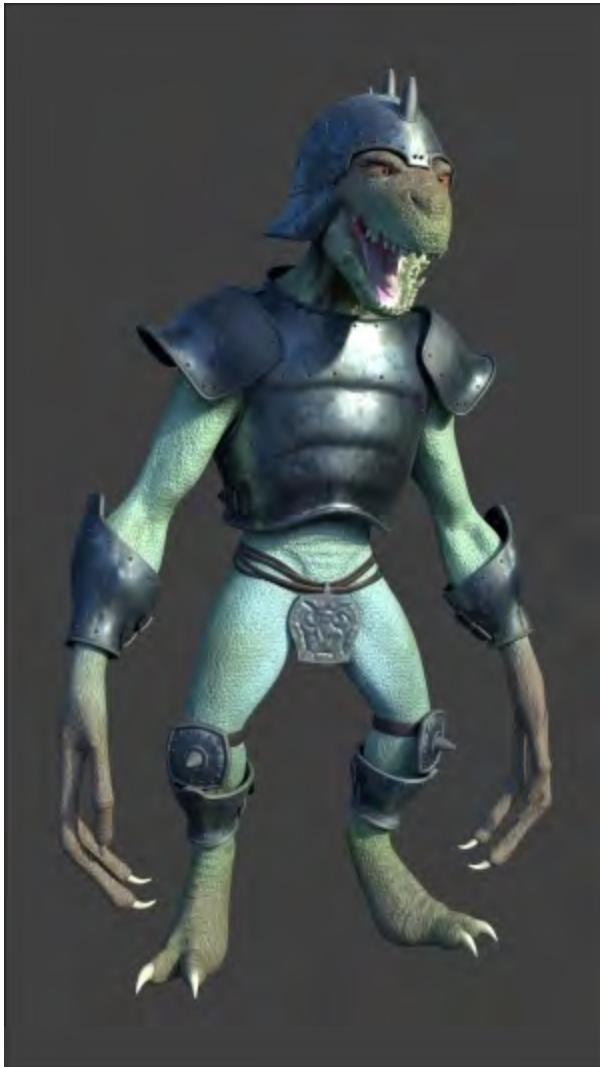
There's more...

The last material created for our character is a very simple leather material made mainly from the output of a **Voronoi Texture** node, contrasted, inverted, and used as bump pattern:



The simple "Leather" material

This completes the creation of the **Gidiosaurus** shaders in **Cycles**:



The completed Gidiosaurus character in Cycles

Of course, reflecting materials, for example, the metallic **armor** surface or the **corneas** (but to some extent also the reptile **skin**), need something to reflect to show them at their best; we'll see this in the last chapter of this cookbook.

In the next chapter, which is the penultimate chapter, we'll see the creation of the same materials in **Blender Internal**.

Chapter 13. Creating the Materials in Blender Internal

In this chapter, we will cover the following recipes:

- Building the reptile skin shaders in Blender Internal
- Building the eyes' shaders in Blender Internal
- Building the armor shaders in Blender Internal

Introduction

In this chapter we'll see how to set up the materials for the **Gidiosaurus** and the **Armor** in the **Blender Render** engine; in fact, although not exactly of the same quality as in **Cycles**, it is also possible to obtain quite similar shader results in **Blender Internal**:



Comparison of the Gidiosaurus character rendered in Cycles (left) and Blender Internal (right)

If you are wondering why we should re-do in the **Blender Render** engine, which is quite old and no longer developed and/or supported, the same thing we have already done in **Cycles**, there are several possible reasons: for example, no doubt **Cycles** is superior in quality but, compared with the scanline **BI**, its rendering is (and, being a path-tracer, always will be) slower; even with the aid of a render-farm, rendering times are still a *money* issue in the production of animations.

The previous screenshot shows, for comparison, only the top parts of two full shot renderings of the **Gidiosaurus** character: the **Cycles** rendering to the left took around **1 hour and 20 minutes** (**1920 × 1080** resolution CPU rendering with *Intel Core 2 Duo T6670 2.20 GHz* and **4 GB** of RAM, in Ubuntu 12.04 64-bit); the **Blender Internal** rendering to the right took only **26 minutes**.

One other reason is that **Cycles'** normals baking capabilities are still not as good as in **Blender Internal** (at the moment, it bakes only the real geometry, contrary to **Blender Internal**, which can also bake the bump output of textures to normal maps), or that it's not as flexible for **Non-Photorealistic Rendering (NPR)** as the **Blender Render** engine.

Just a quick note: normally, materials under the **Blender Render** engine are created directly in the slots inside the **Material** window, often switching to the **Texture** window and back; in the following screenshot, you can see the **Rendered** preview of a generic Red *mono* material assigned to a **UV Sphere**:



A generic "mono" Blender Internal material

But, it's also possible to use node materials in **Blender Internal**, created and connected inside the **Node Editor** window; basically, let's say that *two or more materials can be mixed through nodes* to obtain more advanced results. In the following screenshot, for example, the mono Red material is mixed with a mono Green material through the output of a **Voronoi** texture connected to the **Fac** input socket of a **MixRGB** node:



Two mono materials mixed in the Node Editor window

This is the way we are going to create the **Blender Internal** shaders.

Building the reptile skin shaders in Blender Internal

Because we want to keep the materials we already created for **Cycles** in the same blend file (and the reason will be clear in the next chapter), before we start with the creation of the **Blender Internal** shaders, we must prepare the file a bit.

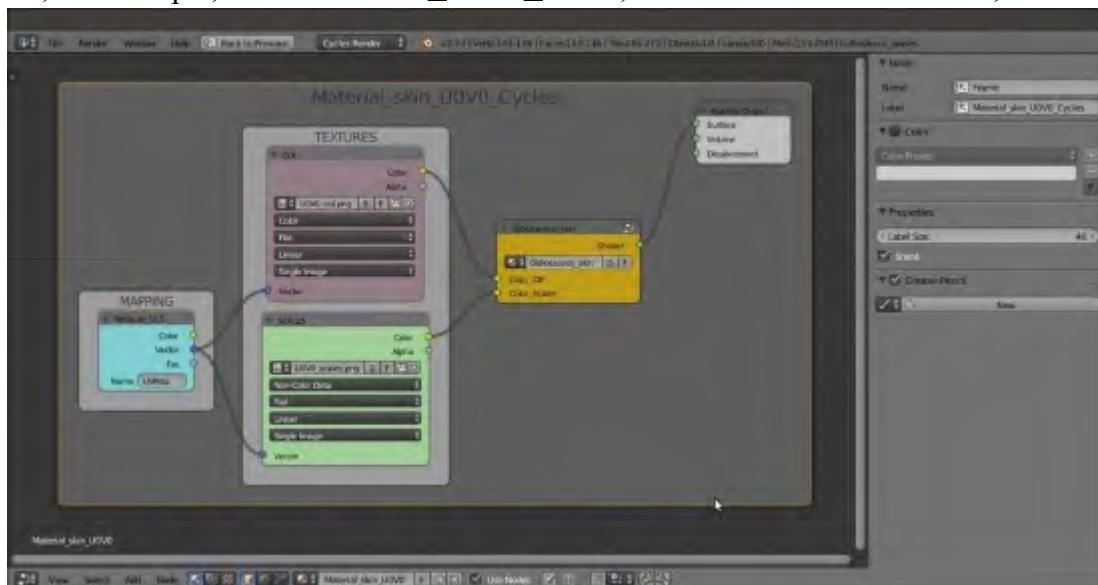
Getting ready

The first thing to do is to open the last saved blend file, add **Frames** to each material in the **Node Editor** window, and label them with the material name followed by the suffix **_Cycles**; this is to later distinguish them from the material we will build for **BI**.

Therefore:

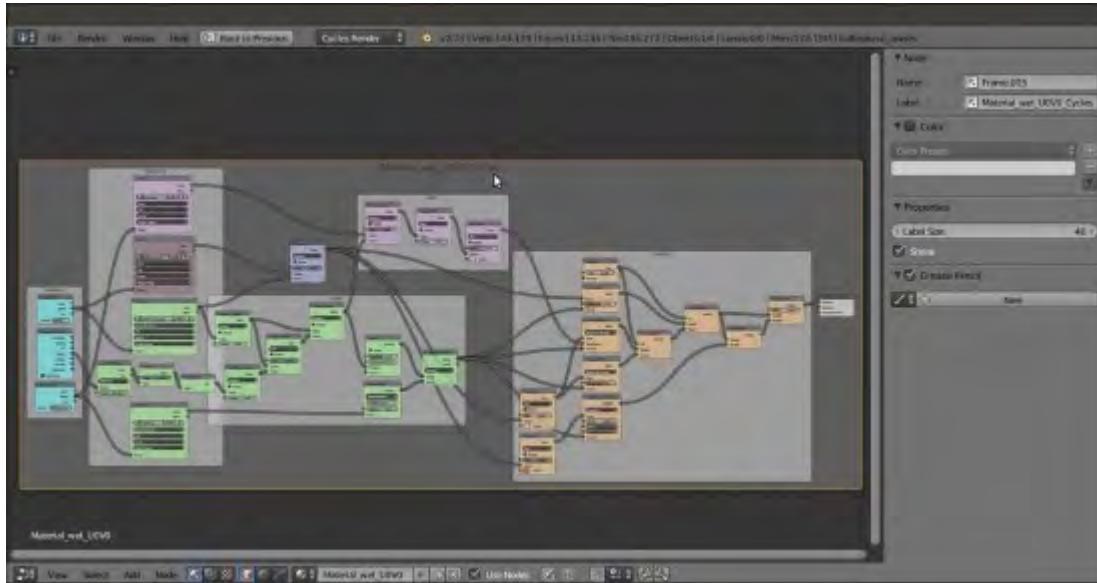
1. Start Blender and load the `Gidiosaurus_shaders_Cycles.blend` file.
 2. In the **Outliner**, select the **Gidiosaurus_lowres** mesh, go to the **Material** window and click on the **Material_skin_U0V0** slot; put the mouse pointer inside the **Node Editor** window and press *Shift + A* to add a **Frame** (*Shift + A | Layout | Frame*).
 3. Press *A* to select all the nodes (the added **Frame**, already selected, becomes the active one) and then press *Ctrl + P* to parent them to the active **Frame**.
 4. Select only the **Frame** and press *N* to call the **Properties** sidepanel; in the **Label** slot under the **Name** subpanel, type **Material_skin_U0V0_Cycles**, then go down to the **Properties** subpanel and increase the **Label Size** to **40**.
 5. Repeat the procedure for all the Cycles' **Gidiosaurus_lowres** materials, for the **Eyes** and **Corneas** and for the **Armor** materials.

So, for example, the `Material_skin_U0V0`, in the **Node Editor** window, becomes this:



A "framed" Cycles material

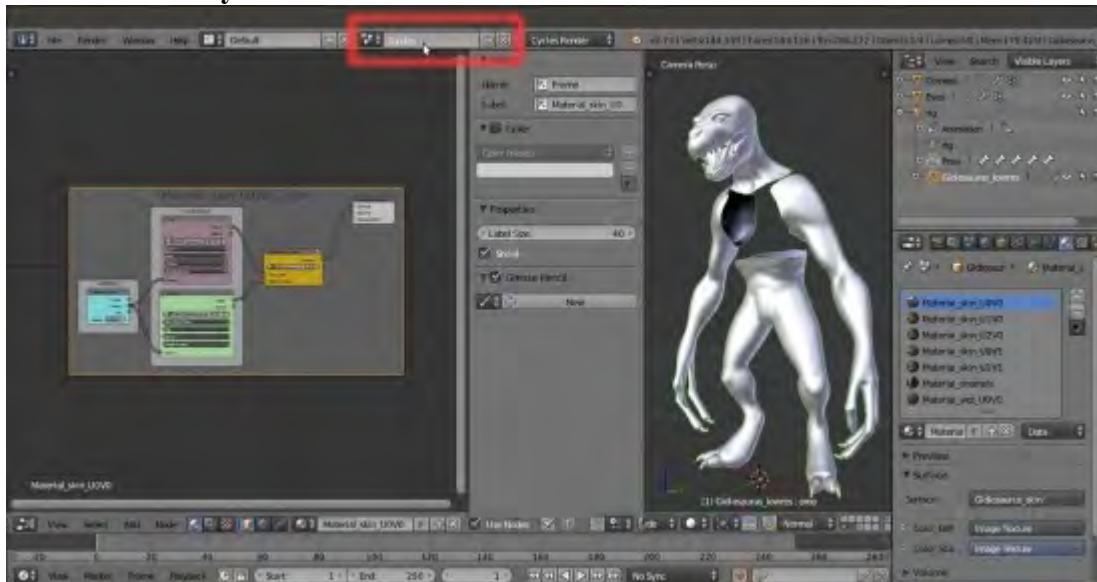
Also, the Material_wet_U0V0, becomes this:



Another "framed" Cycles material

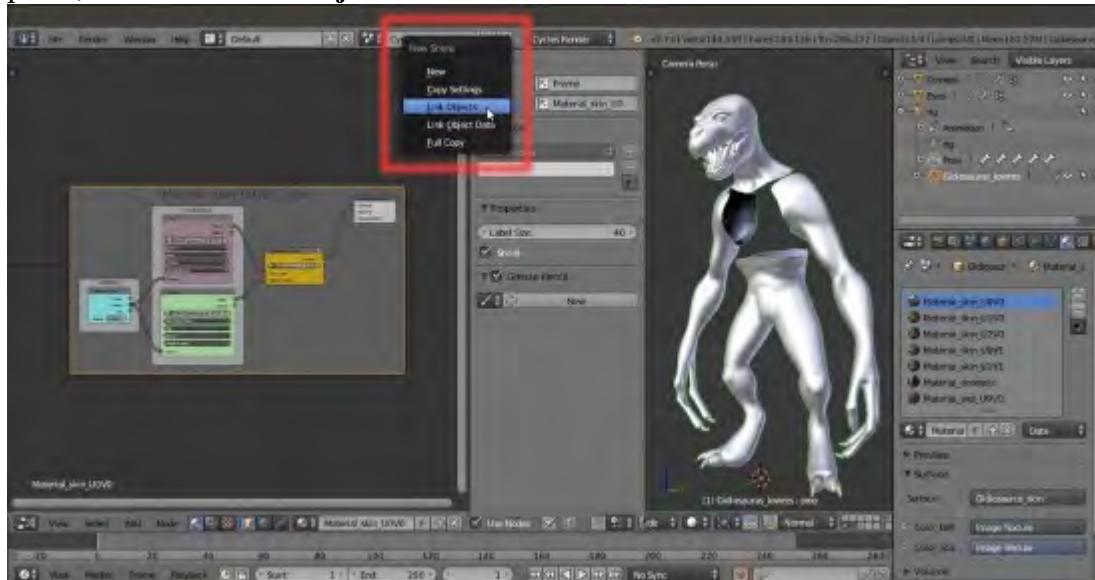
Note that the name of the material is the same as before, the only difference is that a **Frame** labeled with the **_Cycles** suffix has been added in the **Node Editor** window to visually group all the Cycles' nodes that are a constituent of the shader.

6. Now go to the *Scene data block* button on the main top header; left-click on it and rename the **Scene** label as **Cycles**:



Renaming the Scene label

7. Click on the + icon button to the right of the datablock name; in the pop-up little **New Scene** panel, select the **Link Objects** item:



Adding a new scene with linked objects

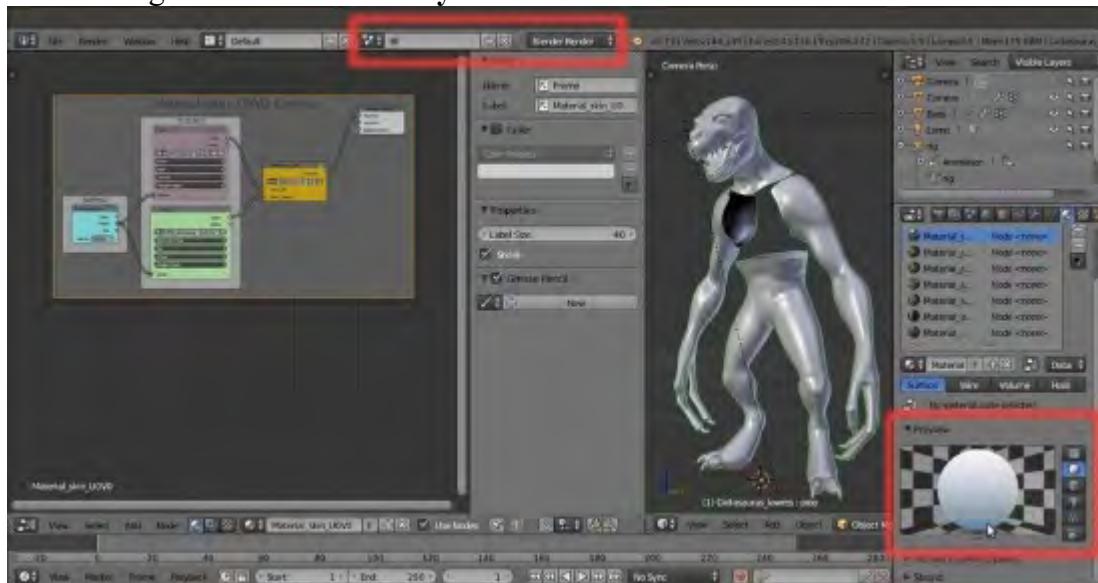
At this point we have created a new scene (automatically labeled as **Cycles.001**) that is sharing the same objects of the other (**Cycles**) scene (be aware of this: the objects in one scene are not a copy of the others, *they are the same objects shared/linked between the two scenes*); you can say which objects are actually linked from one scene to another, by their blue pivot point (for example, look at the highlighted pivot point of the **Gidiosaurus_lowres** object in the following screenshot):



A new scene with linked objects

The advantages of creating new scenes with linked objects are obvious: we can have totally different rendering engines, or different worlds or lamps, in the different scenes and use the same objects and meshes data; so, for example, any modification to a linked object in one scene will automatically be transferred to the other scenes.

- Furthermore, avoid duplicating the objects for each scene; this will help to keep a small file size.
8. Rename the scene from **Cycles.001** to **BI**, then move to the *Engine to use for rendering* button a bit to the right and switch from **Cycles Render** to **Blender Render**.



Switching to the Blender Render engine and the "empty material" preview

Note

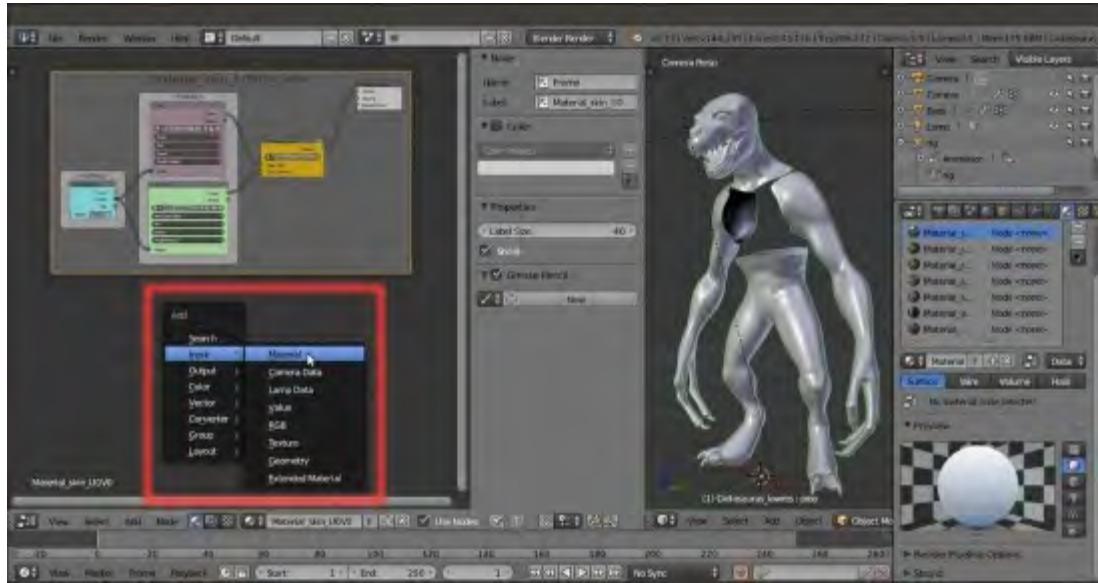
Note that the **Preview** subpanel of the **Material** window shows an *empty material*, to point out that under the current **Blender Render** engine, the material slot, although filled with the **Cycles** material, doesn't have anything to render yet.

9. In the **Outliner**, select the **Lamp** (be sure to have enabled both the **11th** and the **6th** scene layers); go to the **Object Data** window, set the energy to **14.000** and the color to **R 1.000, G 1.000, B 0.650**; under the **Shadow** subpanel, enable the **Buffer Shadow** item, **Filter Type** to **Gauss, Soft = 12.000, Size = 4000**, and **Samples = 16**. Set **Clip Start = 9.000** and **Clip End = 19.000**.
 10. Go to the **World** window and enable the **Ambient Occlusion** by checking the item in the subpanel of the same name; leave the **Blend Mode** to **Add** and set the **Factor** to **0.35**.
 11. Go further down to the **Gather** subpanel and click on the **Approximate** button: check the **Pixel Cache** item and then check also the **Falloff** checkbox under the **Attenuation** item; set the **Strength** to **0.900**.
 12. Enable the **Indirect Lighting** item just above and set the **Factor** to **0.65**.
- These **World** settings are to obtain a sort of **Global Illumination** effect in the **Blender Render** engine; to learn more, have a look at http://www.blender.org/manual/render/blender_render/world/index.html.
13. Save the file as `Gidiosaurus_shaders_Blender_Internal.blend`.

How to do it...

Let's start with the first top **Gidiosaurus** skin material, so:

1. Be sure to have the **Gidiosaurus_lowres** object selected and, back in the **Material** window, click on the **Material_skin_U0V0** slot.
2. Put the mouse pointer inside the **Node Editor** window and press **Shift + A** (**Shift + A | Input | Material**) to add a **Material** node to the window; then press again **Shift + A** and add an **Output** node (**Shift + A | Output | Output**):



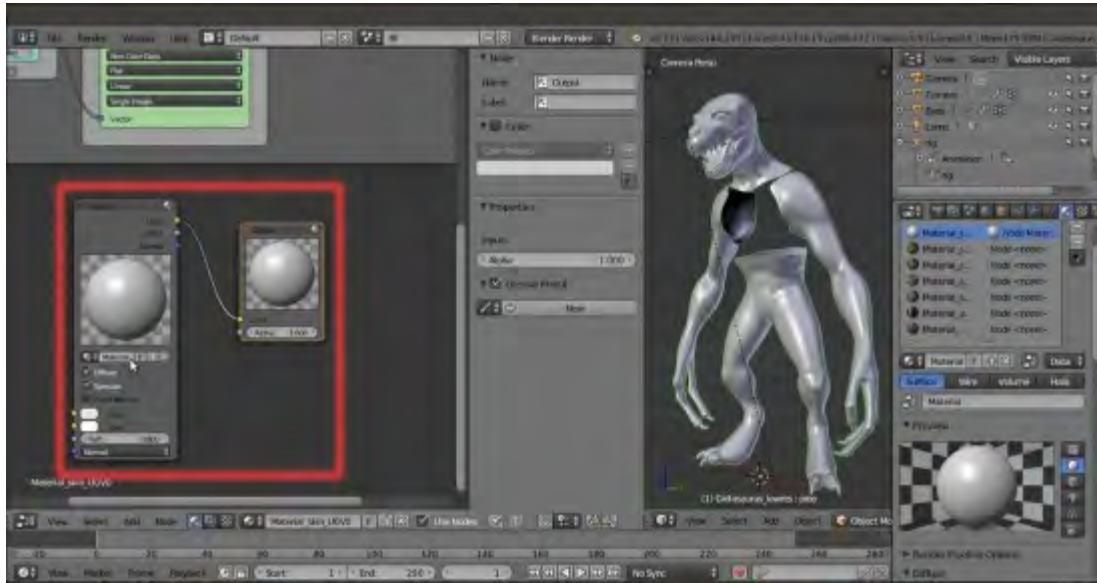
Adding a first material node in the Node Editor window

3. Connect the **Color** output of the **Material** node to the **Color** input socket of the **Output** node:



Connecting the material node to the output node

4. Now click on the **New** button on the **Material** node to create a new default **Blender Internal** material:



Creating a default "mono" material by clicking on the New button in the material node

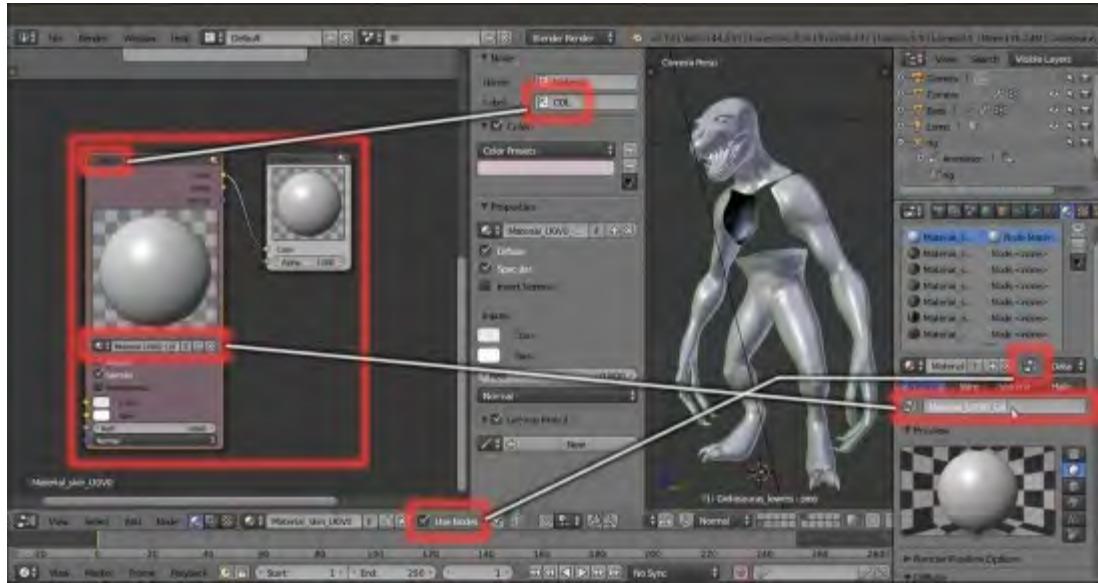
5. In the **Properties** sidepanel of the **Node Editor** window (**N** key to call it) label the **Material** node as **COL** and assign a color.

If you look now at the **Material** window, close to the right side of the material datablock (the name of the material), there is an already enabled and squared button with the symbol of the nodes.

In our case, that button is already enabled because we are already using material nodes; because it's enabled, a second material datablock slot has appeared just further down: that's the datablock slot for any node selected inside the **Node Editor** window and that is part of a material node.

The purpose of this second datablock slot is to let us know which material is the selected one and we are therefore going to edit it by tweaking all the values in the subpanels below.

6. Go to the **Material** window to find the second material name slot: rename the material selected in the **Node Editor** window as **Material_U0V0_Col**; you can do the same thing by clicking on the name datablock on the **COL** node interface.



The corresponding datablock slots in the node interface and in the Material window

7. In the **Node Editor** window, or in the **N Properties** sidepanel, deselect the **Specular** item.
8. Go to the top of the **Material** window and click on the pin icon to the left of the contest; by doing this only the selected material is shown in the window.
9. Go to the **Diffuse** sidepanel and click on the **Diffuse Shader Model** button to select the **Oren-Nayar** item; then go down to the **Shading** subpanel and enable the **Cubic Interpolation** item:



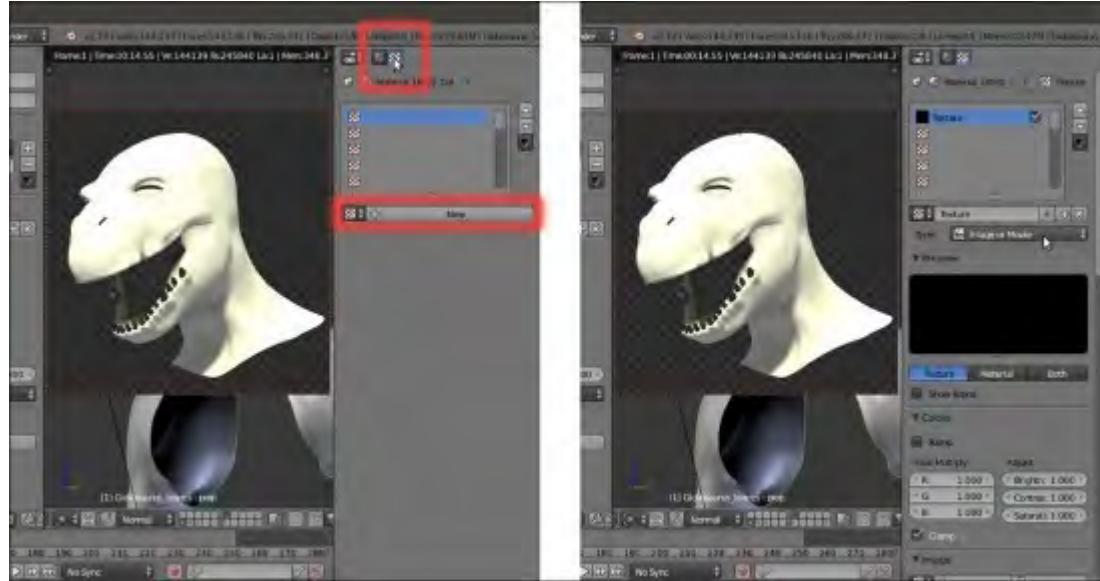
The Specular item to be disabled in the Node Editor window and the shader's parameters to be tweaked in the Material window

- At this point, press **Shift + B** to draw a box around the character's head in the **Camera** view and then zoom to it. If your computer is powerful enough to allow you to work without slowing down, put the mouse pointer inside the 3D viewport and press **Shift + Z** to start the **Rendered** preview; in any case, you can easily enable or disable the preview every time you need it:



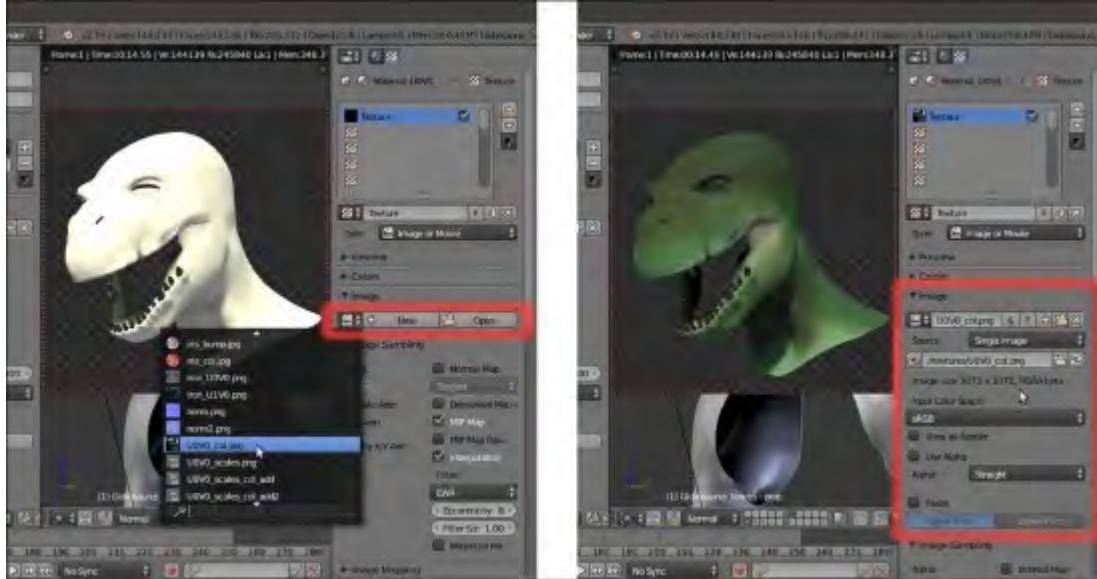
Cropping and starting the rendered preview

- Click on the **Texture** window icon at the top right of the main **Properties** panel, just above the contest, be sure to have the **first** top texture slot selected and click on the **New** button to automatically load a default **Image or Movie** texture panel:



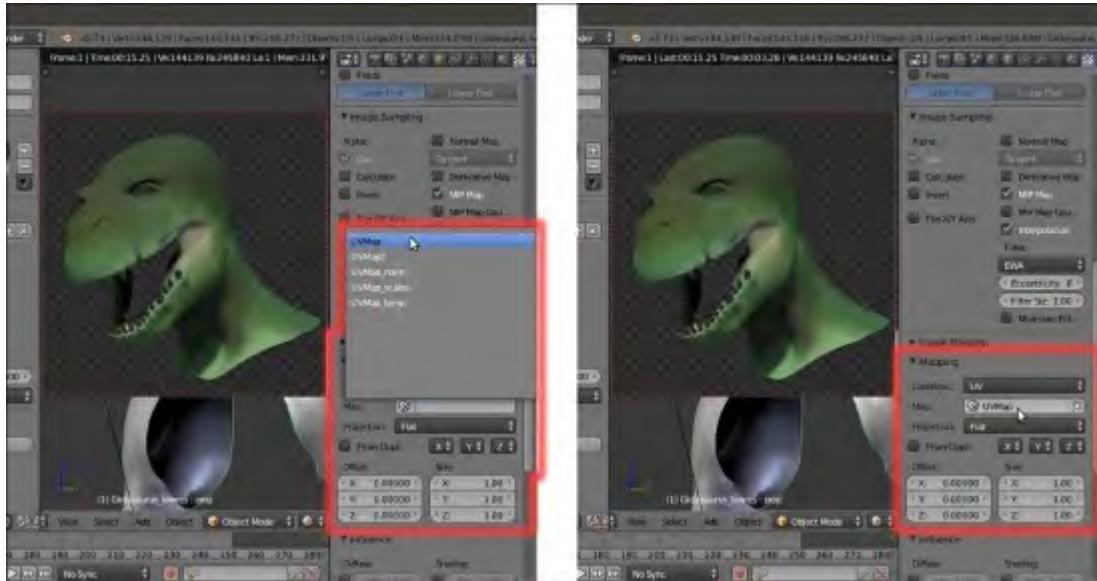
Adding a first texture slot to the material

12. Collapse the **Preview** and the **Colors** subpanels, which at this moment we don't need, and click on the double little arrows to the left side of the **New/Open** buttons in the **Image** subpanel (remember that we have already loaded inside the blend file all the image textures we need, because of the **Cycles** shaders!): in the pop-up menu, select the **U0V0_col.png** item:



Selecting the right image texture from the drop-down list

13. Go further down to find the **Mapping** subpanel: be sure to have the **Coordinates** set to **UV**, the **Projection** to **Flat** (default settings) and click on the **Map** empty slot to select the **UVMap** item.



Selecting the right UV coordinates mapping

14. Go even further down to find the **Influence** subpanel: be sure that the diffuse **Color** channel is the one enabled and that the slider is set to **1.000** (again, default settings):



The Influence settings subpanel for the texture

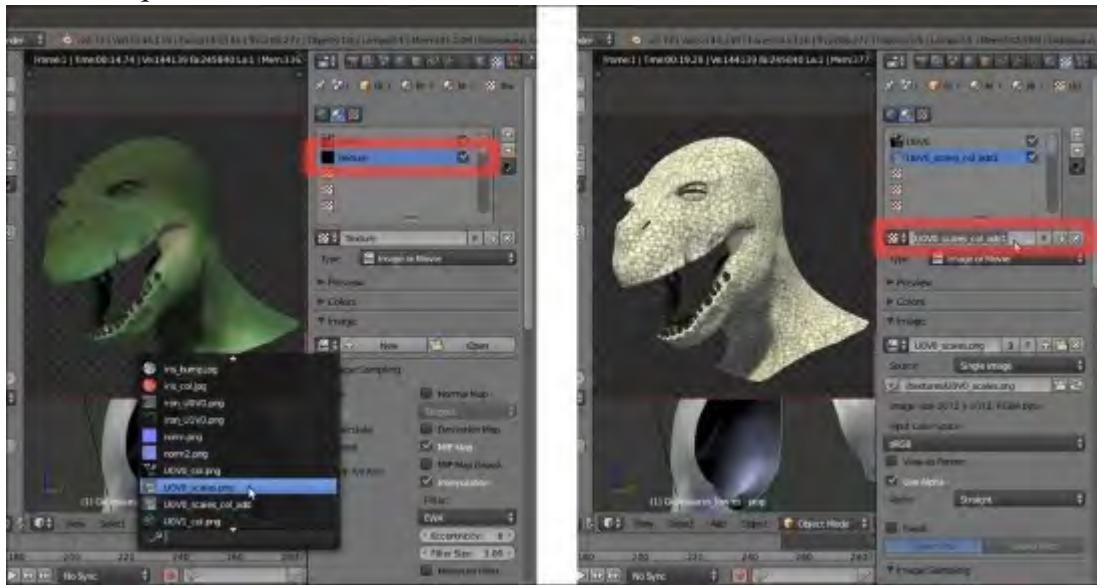
15. Scroll back to the top of the **Texture** window and click on the *Unique datablock ID name* slot, where the generic **Texture** name is written; rename it as **U0V0** (as you can see in the following screenshot, this is the name that also appears in the textures list window):



Renaming the texture datablock

16. Now click on the empty **second** slot: again, click on the **New** button, click on the double arrows and this time load the image **U0V0_scales.png**.

17. In the *Unique datablock ID name* slot, rename it as U0V0_scales_col_add1.

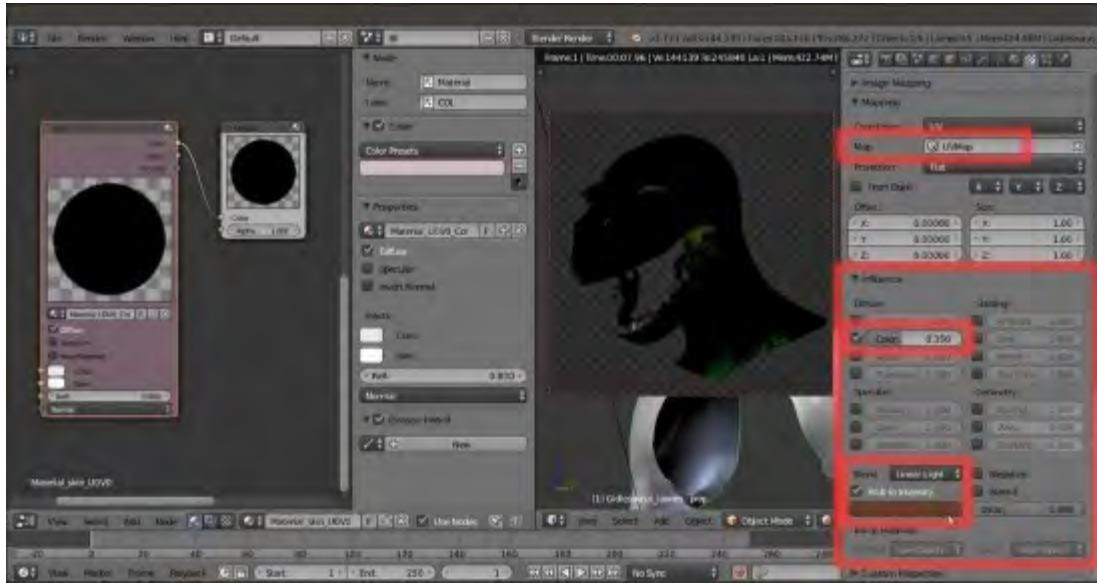


Adding a new texture slot, loading a new image texture and renaming it accordingly

Note

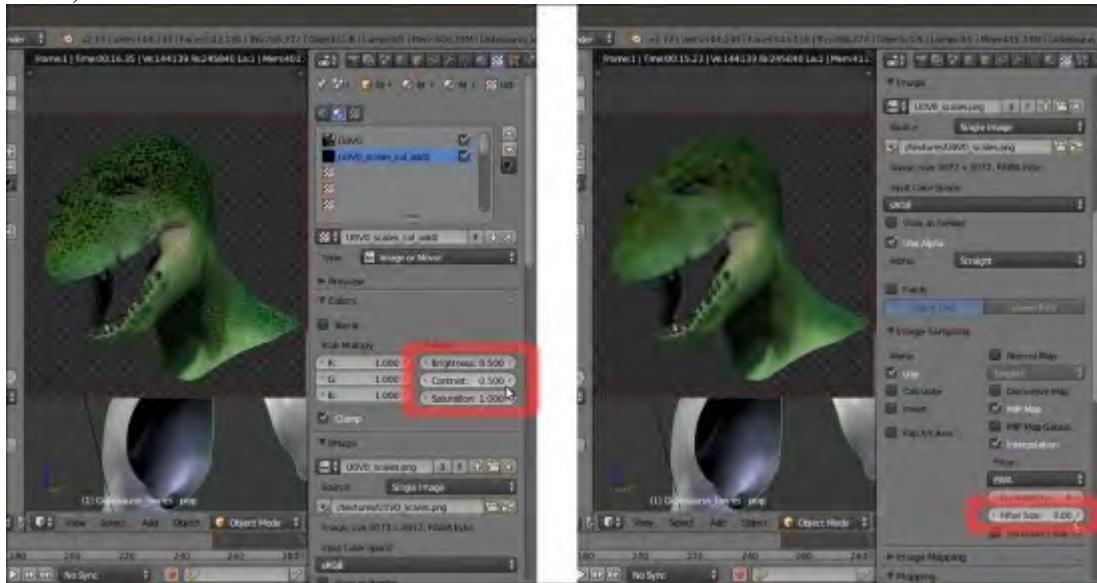
Note that as we load the U0V0_scales.png image in the **second** texture slot, the **Rendered** preview changes to show the grayscale image mapped on the model; this is because, by default, the **Influence** of any new added texture is set to the **Color** channel with a value **1.000** and **Blend Type** to **Mix**.

18. In the **Input Color Space** slot under the **Image** subpanel, change the default **sRGB** to **Non-Color**; then, scroll down to the **Mapping** panel to set the **UVMap** coordinates item and then go to the **Influence** subpanel: leave the **Color** channel enabled but move the slider to the lower value of **0.350**, then change the **Blend Type** to **Linear Light** (for the **Blender Internal** materials, the **Blend Type** works as the layer system of a 2D image editor such as **Photoshop** or **Gimp**); enable the **RGB to Intensity** item and change the pink color to **R 0.130, G 0.051, B 0.030**.



Tweaking the Influence settings for the second texture

19. Go up to expand the **Colors** subpanel: set the **Brightness** and the **Contrast** to **0.500**, to make the texture less bright and less contrasted. Go down to the **Image Sampling** subpanel and set the **Filter Size** to **3.00**, to blur the image (values beyond **1.00** start to blur the image more and more):



Modifying the appearance of the second image texture

20. Select the **third** empty texture slot and repeat the procedure, again loading the **U0V0_scales.png** image; in the *Unique datablock ID name* slot, rename it as **U0V0_scales_col_add2**.

21. Scroll down to the **Mapping** panel to set the **UVMap** coordinates item and then in the **Influence** subpanel, leave the **Color** channel enabled at value **1.000** but change the **Blend Type** to **Subtract**. Set the **Brightness** to **0.100** and the **Contrast** to **1.500**. Again, set the **Filter Size** to **3.00**.
22. Select the **fourth** texture slot, load again the U0V0_scales.png image, rename it U0V0_scales_col, set the **UVMap** coordinates layer, **Color = 1.000** and **Blend Type = Divide**:



Adding more texture slots with different settings

23. Select the **fifth** texture slot, load the vcol.png image again, rename it vcol, set the **UVMap_norm** coordinates layer, **Color = 0.800** and **Blend Type = Screen**:



Adding the baked Vertex Color image texture as well

At this point we have completed the **first** component of the skin shader, that is, the **diffuse color** component; in the following *F12* render you can see the final result:



The completed diffuse color component of the Blender Internal skin material

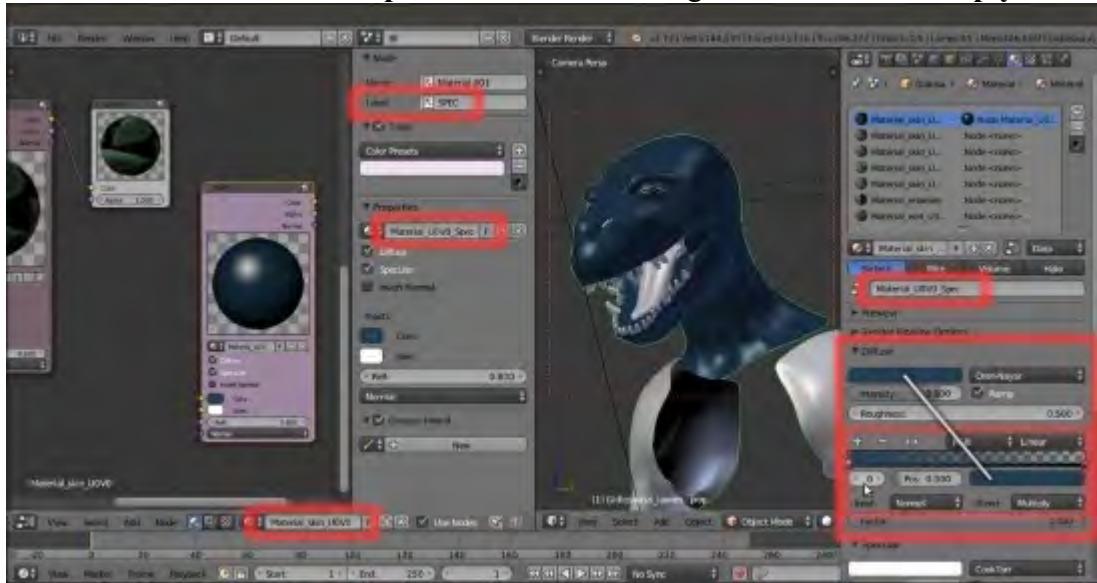
Note that this *F12* render result is quite different from the **Rendered** real-time preview; this is probably due to the complexity of using several textures inside a node material system with the (sadly) bad real-time viewport performances of Blender.

Note

Also note that the only parts of the **Gidiosaurus** mesh that appear in the rendered image are actually the parts we assigned a **Blender Internal** material to; in fact, the **teeth** and the **tongue** are rendered as blank shapes (even working as a mask).

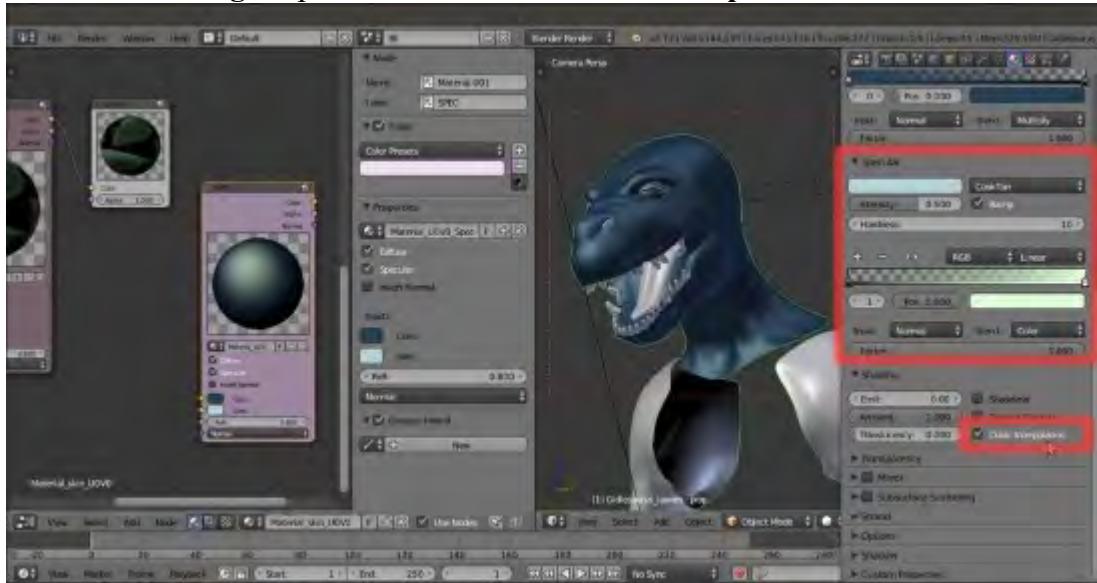
- Now, we can carry on with building the second component of the shader, the **glossy** component.
24. Put the mouse pointer inside the **Node Editor** window and add a new **Material** node (*Shift + A | Input | Material*); label it as **SPEC** and then click on the **New** button to create a new material: rename it **Material_U0V0_Spec**.
 25. Go to the **Material** window; in the **Diffuse** sidepanel, change the shader model to **Oren-Nayar**, then change the color to a deep blue **R 0.020, G 0.051, B 0.089**.
 26. Enable the **Ramp** item: in the slider, switch the positions of the two color stops (that is: white color stop to position **0.000** and black color stop to position **1.000**), then select the white color stop; put the mouse on the deep blue color slot of the **Diffuse** subpanel and press *Ctrl + C* to copy it; put the mouse pointer on the color slot of the selected color stop and press *Ctrl + V* to paste the deep blue color.

27. Click on the **Diffuse Ramp Input** button at the bottom of the subpanel to select the **Normal** item and on the **Diffuse Ramp Blend** button to the right to select the **Multiply** item:



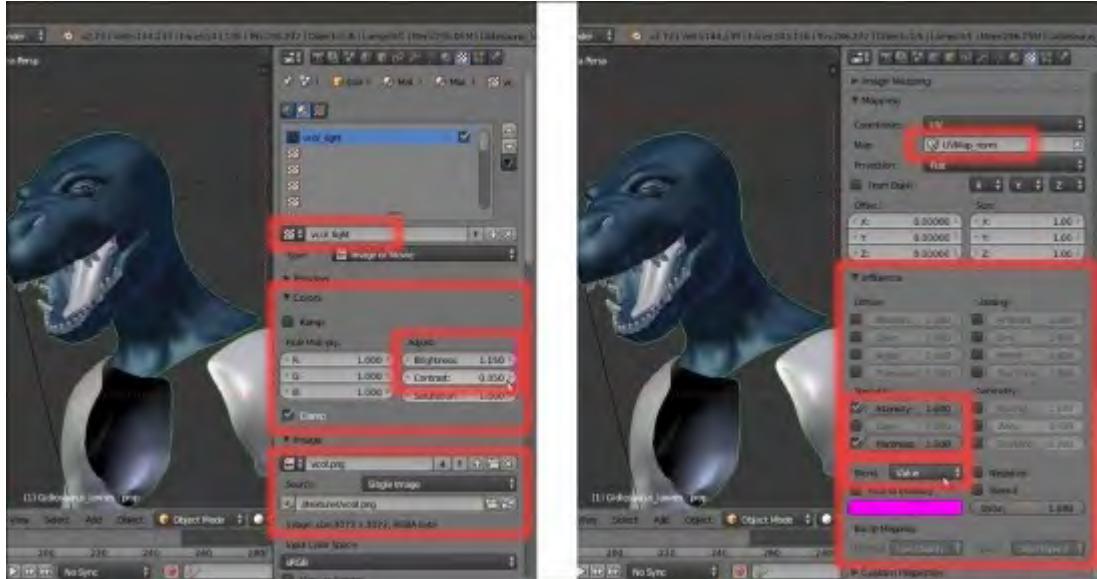
The "Material_U0V0_Spec", to be used inside the "Material_skin_U0V0" node material

28. Scroll down to the **Specular** subpanel: change the color to a light blue **R 0.474, G 0.642, B 0.683**; set the **Intensity** to **0.600** and the **Hardness** to **10**.
 29. Enable the **Ramp** item: select the white color stop and change the color to **R 0.761, G 1.000, B 0.708**, then set the **Diffuse Ramp Input** button to **Normal** and the **Diffuse Ramp Blend** to **Color**.
 30. Go to the **Shading** subpanel and enable the **Cubic Interpolation** item:



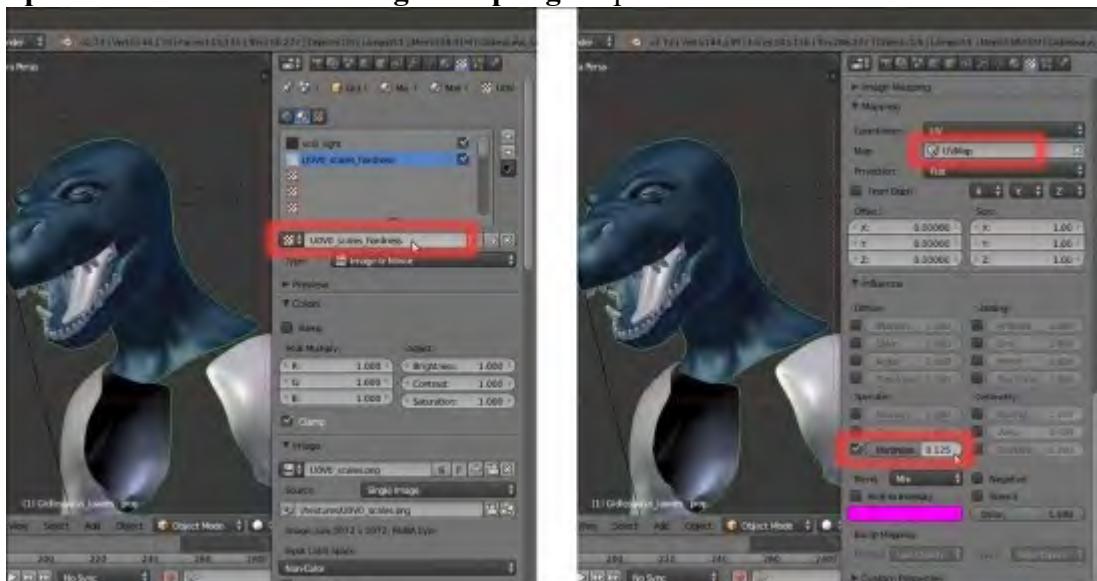
Setting the parameters of the specularity component

31. Go to the **Textures** window; select the top **first** empty texture slot and click on the **New** button. Load the image `vc01.png`, rename the ID datablock as `vc01_light` and go to the **Colors** subpanel: set the **Brightness** to **1.150** and the **Contrast** to **0.850**. Go down to the **Mapping** subpanel and set the **UVMap_norm** coordinates layer, then in the **Influence** subpanel disable the diffuse **Color** channel and enable both the **Intensity** and the **Hardness** channels under **Specular**; set the **Blend Type** to **Value**:



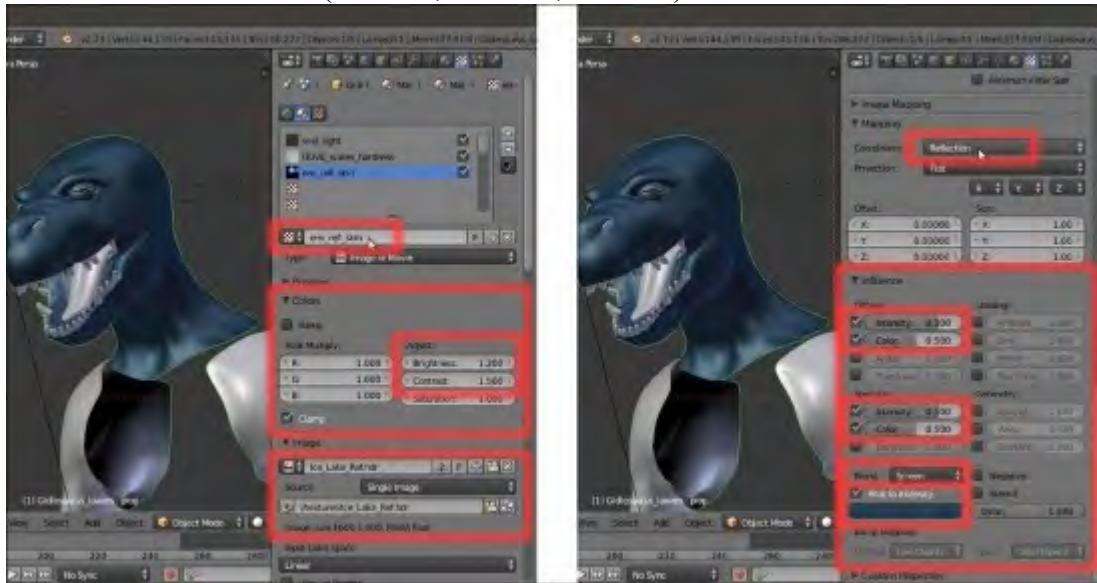
The settings for the specularity first texture

32. Go to the **second** slot and load the image `U0V0_scales.png`; rename it as `U0V0_scales_hardness`, in the **Mapping** subpanel, set the **UVMap** coordinates layer, in the **Influence** subpanel disable the diffuse **Color** and enable the **Hardness** channel under **Specular** to **0.125**. In the **Image Sampling** subpanel set the **Filter Size** to **5.00**.



Re-using the "UOV0_scales.png" image texture for the specularity hardness

33. In the **third** slot load the image `Ice_Lake_Ref.hdr`, a free high dynamic range image licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License from the **sIBL Archive** (<http://www.hdrlabs.com/sibl/archive.html>); there is a reason we are now using the **hdr** image, and it's explained in the *How it works...* section.
34. Rename the image ID datablock as `env_refl_skin` and in the **Colors** subpanel, set the **Brightness** to **1.200** and the **Contrast** to **1.500**; go to the **Mapping** subpanel and set the **Texture Coordinates** to **Reflection**. Down in the **Influence** subpanel, enable both the **Intensity** channel under **Diffuse** and **Specular** and set their sliders to **0.500**; enable also, the **Color** channel under **Specular** and set the sliders of both the **Color** channels to **0.500** as well. Set the **Blend Type** to **Screen**, enable the **RGB to Intensity** item and set the color to the same deep blue of the diffuse color (**R 0.020, G 0.051, B 0.089**):



Using the environment hdr image as reflection map

If you want to see the effect of the single components in the **Rendered** preview as we build the shader, just temporarily disconnect the **COL** node link to the **Output** node and replace it with the **Color** output of the **SPEC** node (in this case):



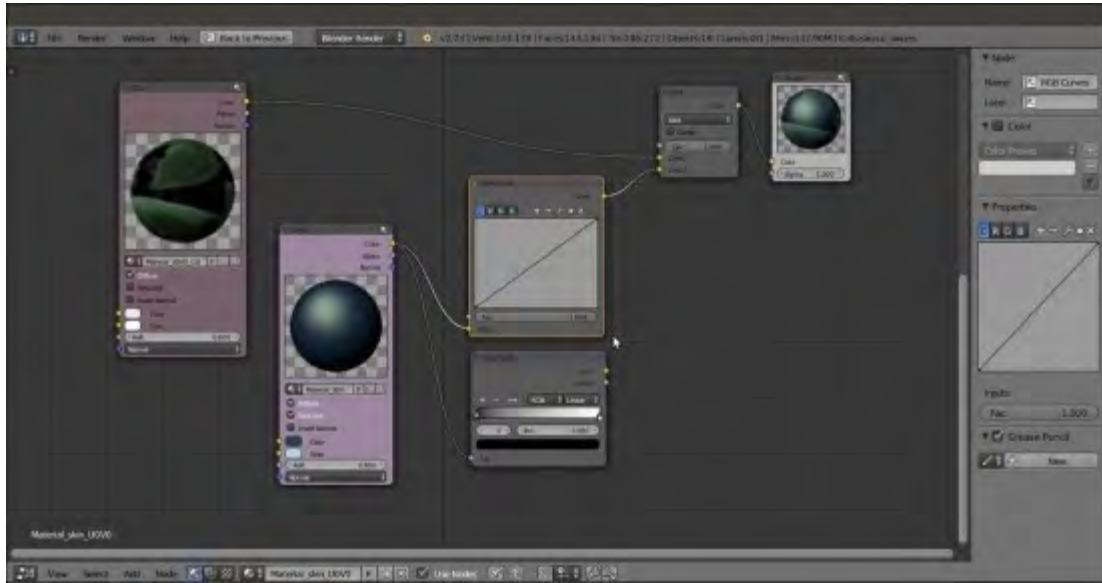
Testing the specularity component material in the rendered preview

35. At this point, add a **MixRGB** node (*Shift + A | Color | MixRGB*) and move it on the link connecting the **COL** node to the **Output** node, to automatically paste it between them; then connect the **Color** output of the **SPEC** node to the **Color2** input socket of the **MixRGB** node, set the **Blend Type** of this latter node to **Add** and its **Fac** value to **1.000**:



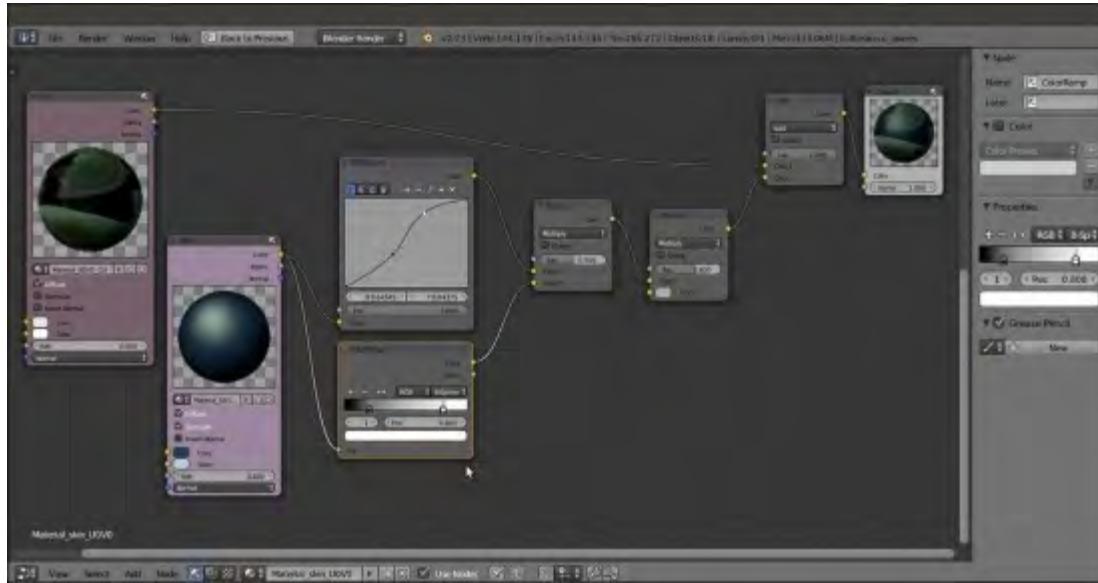
Finally adding the specularity component to the diffuse component

36. Add a **RGB Curves** node (*Shift + A | Color | RGB Curves*) and a **ColorRamp** node (*Shift + A | Converter | ColorRamp*); paste the **RGB Curves** node between the **SPEC** and the **MixRGB** node, then connect the **Color** output of the **SPEC** node also to the **ColorRamp** input socket:



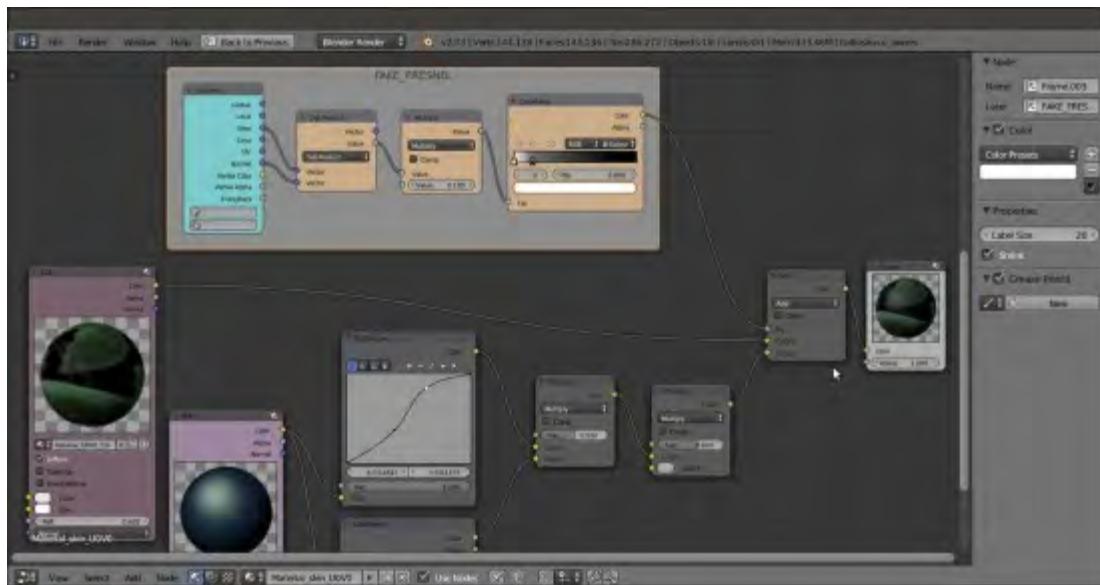
Adding new nodes

37. Press **Shift + D** to duplicate the **MixRGB** node, change the **Blend Type** of the duplicate to **Multiply** and paste it between the **RGB Curves** and the first **Add-MixRGB** nodes: set the **Fac** value to **0.500**. Connect the **Color** output of the **ColorRamp** node to the **Color2** input socket of the **Multiply-MixRGB** node.
38. Press **Shift + D** to duplicate the **Multiply-MixRGB** node and paste the duplicate between the first **Multiply-MixRGB** node and the **Add-MixRGB** node. Set the **Fac** value of the last **Multiply-MixRGB** node to **0.600** and the **Color2** to **R 0.347, G 0.462, B 0.386**.
39. Go to the **RGB Curves** node and left-click inside the interface window to add a point; set its coordinates to **X = 0.38636** and **Y = 0.36875**. Add a second point and set its coordinates to **X = 0.64545** and **Y = 0.84375**.
40. Go to the **ColorRamp** node and set the **Interpolation** to **B-Spline**, then move the black color stop to position **0.195** and the white color stop to position **0.800**:



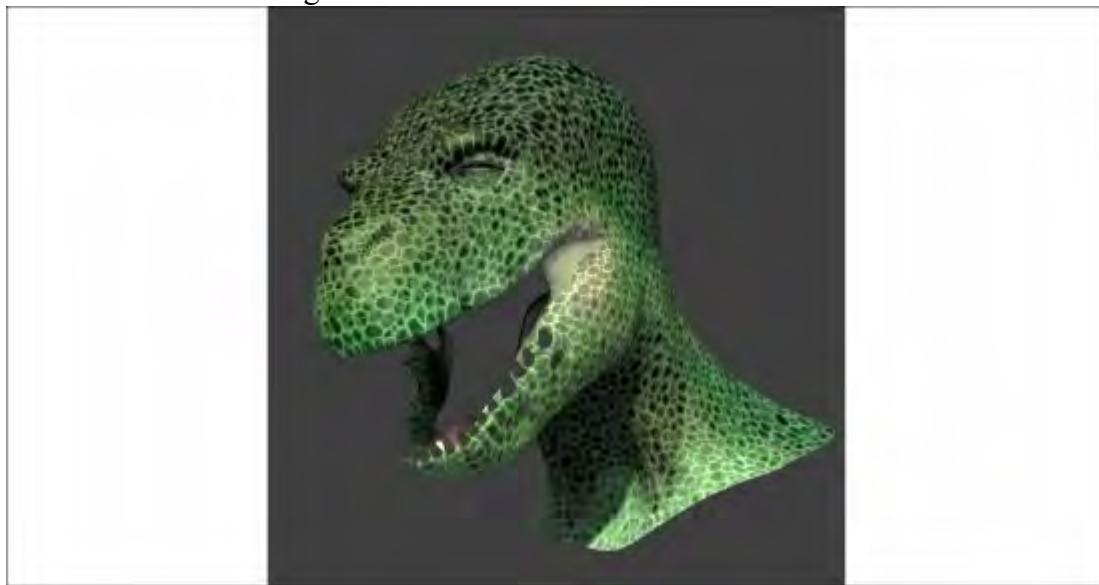
Tweaking the specularity component through the new nodes

41. Add a **Geometry** node (*Shift + A | Input | Geometry*), a **Vector Math** node (*Shift + A | Converter | Vector Math*), a **Math** node (*Shift + A | Converter | Math*), and a **ColorRamp** node (*Shift + A | Converter | ColorRamp*).
42. Add a **Frame** (*Shift + A | Layout | Frame*), label it as **FAKE_FRESNEL** and parent the last **four** added nodes to it.
43. Set the **Operation** of the **Vector Math** node to **Dot Product**, then connect the **View** output of the **Geometry** node to the first **Vector** input socket of the **Vector Math** node, and the **Normal** output of the **Geometry** node to the second **Vector** input socket of the **Vector Math** node.
44. Connect the **Value** output of the **Dot Product** node to the first **Value** input socket of the **Math** node; set the **Operation** of this latter node to **Multiply** and the second **Value** to **0.100**.
45. Connect the output of the **Math** node to the **Fac** input socket of the **ColorRamp** node; set the **Interpolation** of this latter node to **B-Spline**, then move the black color stop to position **0.150** and the white color stop to position **0.000**.
46. Connect the **Color** output of the **ColorRamp** node to the **Fac** input socket of the **Add-MixRGB** node:



Adding the output of a fake Fresnel as factor for the blending of the two components

Let's do a *F12* rendering to see the result so far:



The F12 rendered result so far

47. Now, select the **COL** material node and press *Shift + D* to duplicate it. Label the duplicated one as **SSS**, then through the **Node Editor** window, enable the **Specular** item. Click on the **2** icon button to the right side of the material name datablock to make it single user and rename the new copy of the material as **Material_U0V0_SSS**.

48. Go to the **Material** window and change the **Diffuse Shader Model** to **Minnaert** and the **Diffuse** color to **R 0.439, G 0.216, B 0.141**. Move down to the **Specular** subpanel and change the color to the same **R 0.439, G 0.216, B 0.141** brownish hue (copy and paste), then set the **Intensity** to **0.600** and the **Hardness** to **12**.
49. Go down to the **Subsurface Scattering** subpanel and enable it by checking the checkbox: set the **IOR** value to **3.840**, the **Scale** to **0.001**, copy and paste the brownish color also in the scattering color slot, set the **Color** slider to **0.000** and the **Texture** slider to **1.000**. Set the **RGB Radius**: **R 9.436, G 3.348, B 1.790**.
50. Go to the **Texture** window and set to **0.300** the **Color** channel sliders of the **U0V0**, **U0V0_scales_col_add2**, and **U0V0_scales_col** texture slots, then set to **0.117** the **Color** channel slider of the **U0V0_scales_col_add1** texture slot.
51. Select the last **vcol** texture slot and click on the **X** icon (*Unlink datablock*) button to clear it; click on the double little arrows to the left side of the **New** button and select the **vcol_light** item from the pop-up menu. Set the **Mapping** to **UVMap_norm**, and under the **Influence** subpanel, the **Color** of **Diffuse** to **0.267** and the **Blend Type** to **Screen**.



Testing the output of the SSS component material

52. Add a new **MixRGB** node (*Shift + A | Color | MixRGB*), paste it between the **Add-MixRGB** and the **Output** node and set the **Fac** value to **0.250**.
53. Press *Shift + D* to duplicate this **Mix-MixRGB** node; change the **Blend Type** of the duplicated one to **Screen**, set the **Fac** value to **1.000** and connect the output of the **Add-MixRGB** also to the **Color1** input socket of the **Screen-MixRGB** node; connect the output of the **SSS** node to the **Color2** input socket of the **Screen-MixRGB** node.
54. Connect the output of the **Screen-MixRGB** node to the **Color2** input socket of the first **Mix-MixRGB** node.



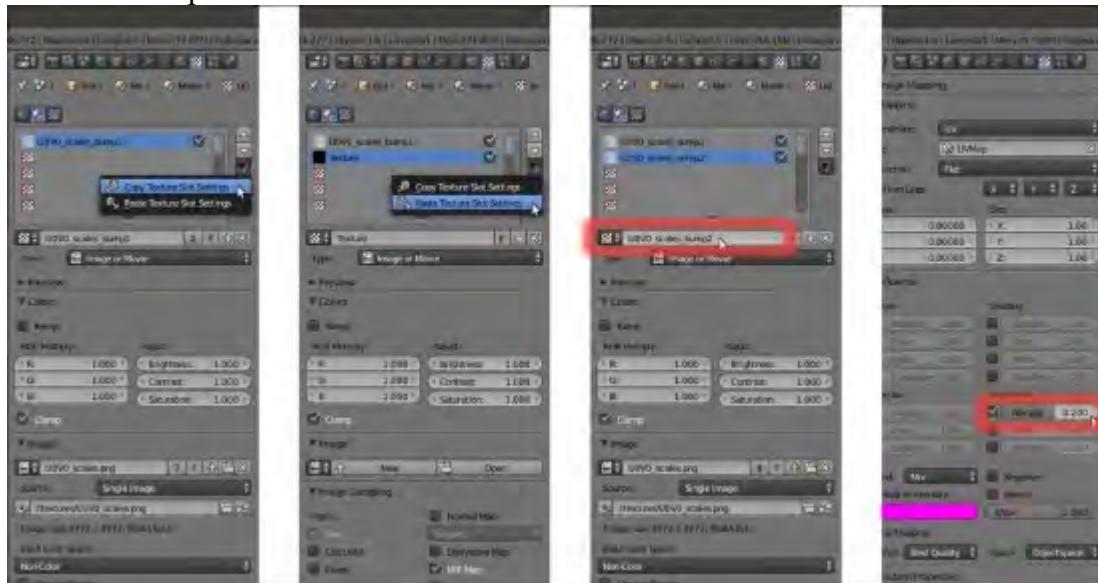
Adding the SSS component to the rest of the shader

55. Add a new **Material** node (**Shift + A | Input | Material**) and click on the **New** button to create a new material; label the node as **Scales_bump** and rename the material as **Material_U0V0_Scales_bump**.
56. In the **Material** window set the **Diffuse Shader Model** to **Oren-Nayar** and the **Specular Shader Model** to **Blinn**, **Intensity = 0.100** and **Hardness = 5**. In the **Shading** subpanel enable the **Cubic Interpolation** item.
57. Go to the **Texture** window and in the **first** slot load the **U0V0_scales.png** image; rename the ID datablock as **U0V0_scales_bump1**. In the **Image Sampling** subpanel set the **Filter Size** to **5.00**, the **Mapping** to **UVMap** and in the **Influence** subpanel disable the **Color** channel and enable the **Normal** channel under **Geometry**: set the slider to **0.100** and go to the bottom to click on the **Bump Method** slot and select **Best Quality**:



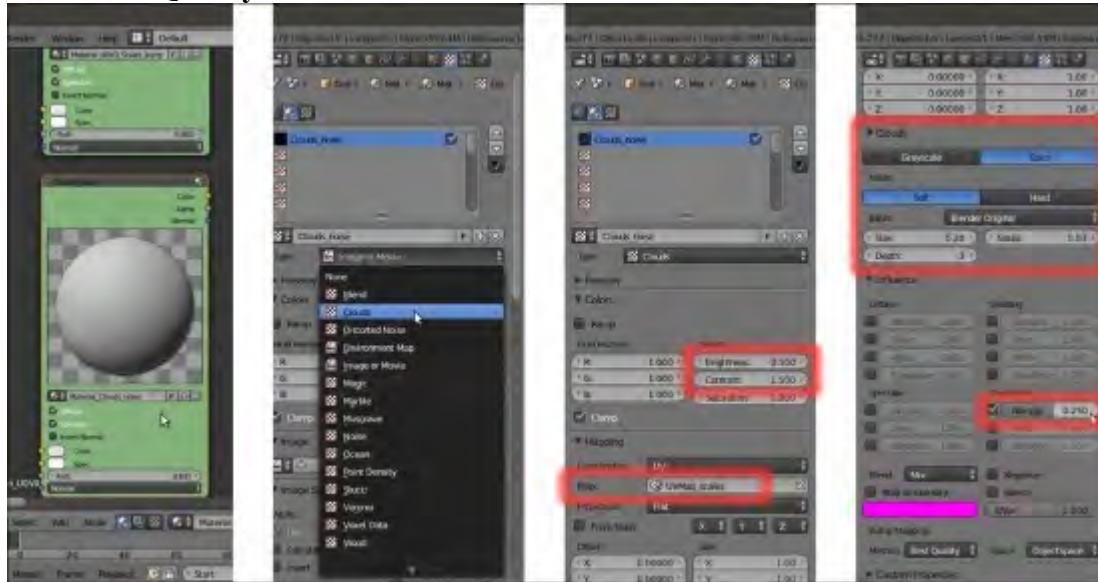
The bump material node

58. Go back to the top of the panel and click on the big black arrow to the right of the texture window; select the **Copy Texture Slot Settings** item.
59. Select the empty **second** texture slot and click on **New** to add a generic texture, then click again on the black arrow to select the **Paste Texture Slot Settings** item.
60. In the *ID datablock* slot, click on the **2** icon button to make it single user and rename it **U0V0_scales_bump2**.
61. Go down to the **Image Sampling** subpanel and set the **Filter Size** to **1.00**, then go down to the **Influence** subpanel and set the **Normal** slider to **0.200**.



Copying and pasting a texture slot

62. Press **Shift + D** to duplicate the node; make the material of the duplicated one single user and rename it as **Material_Clouds_noise**, then label the node as **Clouds_noise**.
63. In the **Texture** window, delete (unlink) **U0V0_scales_bump1** and **U0V0_scales_bump2**, and then select the **first** slot and click on the **New** button: change the automatic **Texture.001** ID datablock name with **Clouds_noise**, click on the **Type** button and in the pop-up menu select the **Clouds** item.
64. In the **Colors** subpanel set the **Brightness** to **0.500** and the **Contrast** to **1.500**, in the **Mapping** subpanel set the **UVMap_scales** coordinates layer, in the **Clouds** subpanel switch from **Grayscale** to **Color**, set the **Size** to **0.20**, the **Depth** to **0.3** and the **Nabla** to **0.05**.
65. In the **Influence** subpanel disable the **Color** channel and enable the **Normal** channel under **Geometry**: set the slider to **0.250** and go to the bottom to click on the **Bump Method** slot and select **Best Quality**:



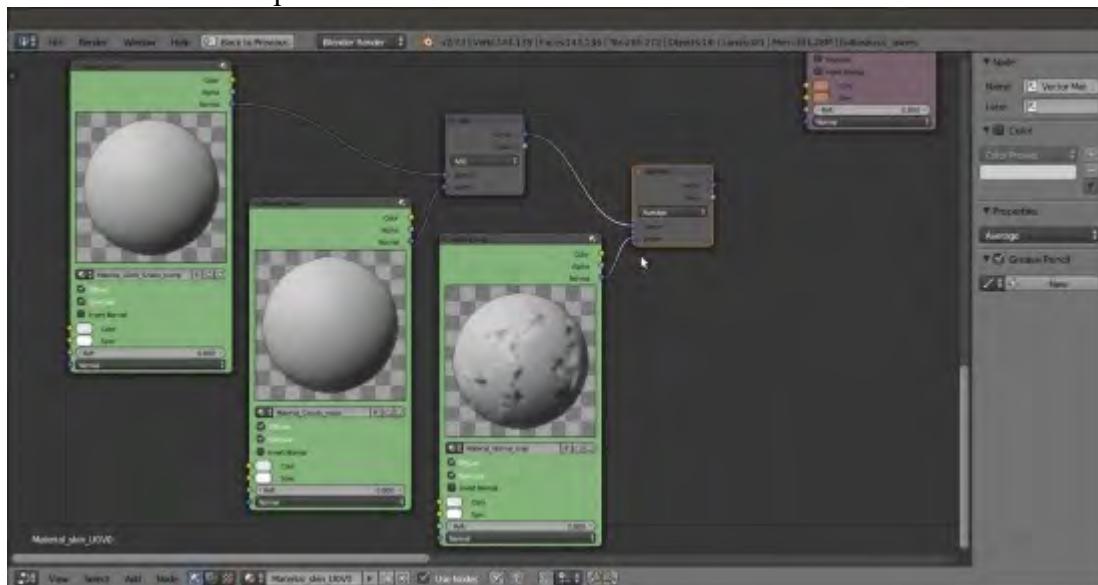
The second bump material node

66. Press **Shift + D** to duplicate the **Scales_bump** node; make the material of the duplicated one single user and rename it as **Material_Normal_map**, then label the node as **Normal_map** as well.
67. In the **Texture** window, delete (unlink) the **U0V0_scales_bump1** and **U0V0_scales_bump2**; select the **first** slot and click on the **New** button: change the automatic **Texture.001** ID datablock name to **normal** and then load the **norm.png** image.
68. In the **Mapping** subpanel set the **UVMap_norm** coordinates layer, then go to the **Image Sampling** subpanel and enable the **Normal Map** item; go to the **Influence** subpanel, disable the **Color** channel and enable the **Normal** channel: set the slider to **1.000** (higher values don't have an effect with normal maps in **Blender Internal**).



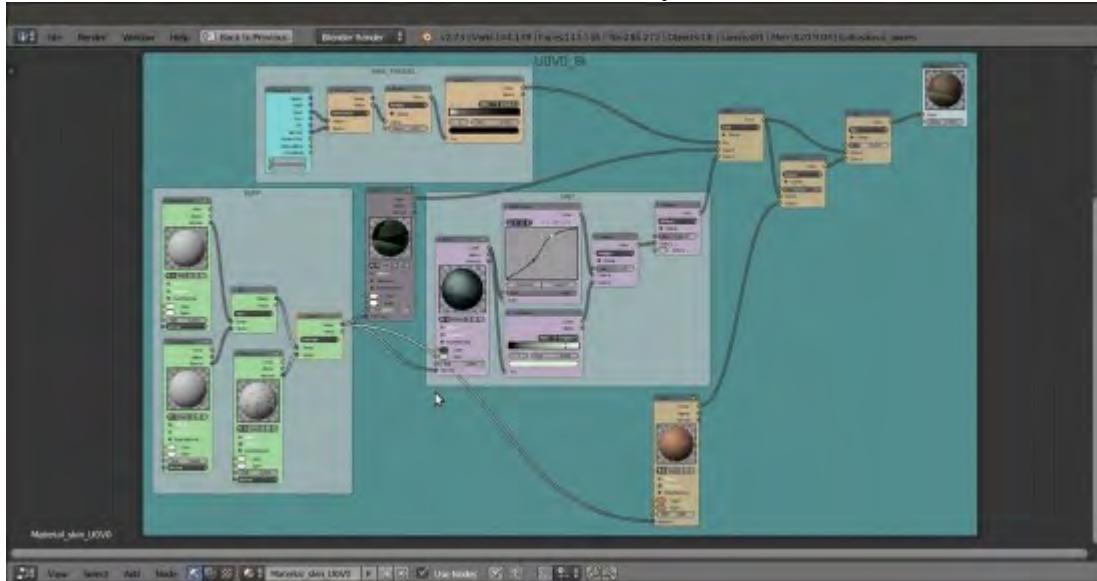
The normal map material node

69. Add a **Vector Math** node (*Shift + A | Converter | Vector Math*) and connect the **Normal** (blue) output of the **Scales_bump** node to the first **Vector** input socket of the **Vector Math** node, and the **Normal** output of the **Clouds_noise** node to the second **Vector** input socket.
70. Press *Shift + D* to duplicate the **Vector Math** node, set the **Operation** of the duplicate to **Average** and connect the **Vector** output of the **Add-Vector Math** node to the first **Vector** input socket of the **Average-Vector Math** node, and the **Normal** output of the **Normal_map** node to the second **Vector** input socket.



Connecting the outputs of the three bump nodes

71. Connect the **Vector** output of the **Average-Vector Math** node to the **Normal** input sockets of the **COL**, **SPEC**, and **SSS** material nodes.
72. Add frames everywhere to make things clear but especially to visually group and separate the nodes of the **Blender Internal** material from the **Cycles** ones.



The completed "U0V0_BI" node material

73. Save the file.

How it works...

You have probably noticed that a few of the nodes we can find in the **Cycles** material system are also available for the material nodes in **Blender Internal**; sadly, some are still missing (and probably forever will be), as, for example, a **Fresnel** node that, in fact, we had to approximate with a combination of other different nodes.

Anyway, although not all the same nodes are at our disposal, we had enough of them to try to obtain a result as close as possible as the result we obtained in the **Cycles** material (in the previous [Chapter 12, Creating the Materials in Cycles](#)).

Note

One thing you should absolutely keep in mind when loading the textures into the material nodes in **Blender Internal** is their order in the texture stack. This is important and must be taken into consideration according to the result we need, because a texture can totally overwrite the texture in the above slot (with the default **Mix** blend type) but can also be added, subtracted, multiplied, divided, and so on; the textures stack works the same as the layer stack system of a 2D graphic editor (**Gimp**, for instance), with the order from the top to the bottom and the different blending options (the **Blend Type** items).

Having said that, let's see the steps:

- From step 1 to step 9 we created a basic **Blender Internal** material node by using both the **Node Editor** and the **Material** window.
- From step 10 to step 23 we assigned the proper textures to the basic material node that becomes, in this case, the **Material_U0V0_Col**, the basic **diffuse color** component of the shader.

These steps have been described in the most detailed way possible because they are the same steps for all the textures added to the materials; of course, the values and the settings can be different, but basically:

- We add a texture (image or procedural)
- We set a mapping orientation
- We set the influence value on the selected channel (also more than one at a time)
- Because the same texture can be used (with different settings) more than once, we always rename the *Unique datablock ID* name to make them easily recognizable in the pop-up menu list.
- The **Filter Size** value in the **Image Sampling** subpanel is really useful for blurring an image texture: a value of **1.00** is the default sharpness, while a higher value makes the image more and more blurred.
- From step 24 to step 30 we created the **glossy/specular** **Material_U0V0_Spec** node.
- From step 31 to step 34 we added the textures to the **Material_U0V0_Spec** material. This material should represent the **glossy/specular/mirror** component of the shader, that is probably the most important thing for obtaining a correct visual result; in **Cycles** the **Glossy BSDF** shader node provides the result perfectly, while in **Blender Internal**, we have two options: one, by enabling the (slow and imperfect) internal ray-tracing **Mirror** item, or by faking it. We faked it by setting an image (the same **hdr** we'll use in the next chapter in the **World** both for **Cycles** and **BI**) on the **Reflection** channel of the shader, hence giving the impression of an environment (slightly) mirrored by the character's skin.
- At step 35 we added together the outputs of the **COL** and of the **SPEC** nodes.
- From step 36 to step 40 we tweaked the output of the **SPEC** nodes to obtain a more realistic output/distribution of the glossiness on the mesh's surface, trying to mimic, as much as possible, the glossy output of the **Cycles** shader version.
- From step 41 to step 46 we built a fake **Fresnel** to work as a factor for the blending of the **glossy** and the **diffuse** components; this works by calculating the dot product of the vectors of the point of view and the mesh's normals. Be aware that it isn't actually working as the real **Fresnel** node that you find in **Cycles**, and that it has several limitations. By varying the second **Value** of the **Math** node and/or the black color stop position of the **ColorRamp** node, we can obtain several nice effects in some way visually similar to the real output. If you are wondering why we don't use the output of a **BI** material with a **Fresnel** diffuse shader model, sadly it doesn't seem to work correctly (actually, it doesn't seem to work at all).
- From step 47 to step 51 we built the **SSS** material node by duplicating the **COL** node, making a new copy of the material and renaming it as **Material_U0V0_SSS**, then enabling **Subsurface Scattering** and modifying the influence values of the textures; the values of the subsurface scattering (**IOR**, **Scale**, **RGB Radius**), instead, were borrowed from the **Cycles** version of the shader.

- From step 52 to step 54 we added the output of the **SSS** node to the rest of the shader by using a **Blend Type** set to **Screen** plus a **Mix** one set to a low **Fac** value; basically, the exact copy of what we did in **Cycles**.
- From step 55 to step 71 we created the **bump pattern**, divided into **three** different material nodes to give us more flexibility in adding and averaging them together in a way that is as similar as possible to **Cycles**:

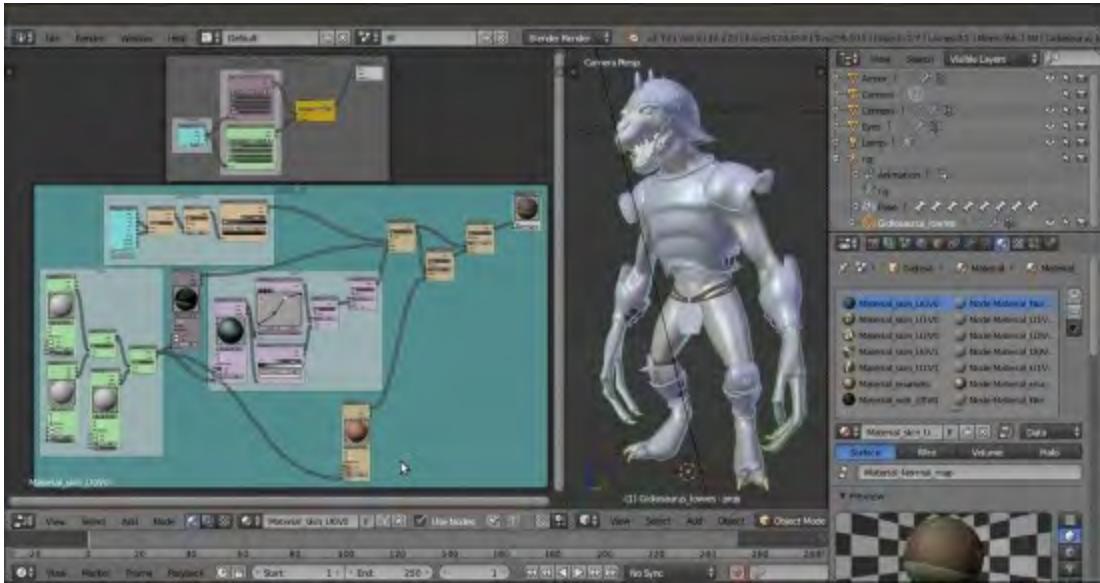


The F12 rendered final result in Blender Internal

There's more...

The other missing shaders for the **Gidiosaurus** skin are solved in exactly the same way we used in the previous chapter for the other **Cycles** skin shaders: by selecting the entire **BI** frame with all the parented nodes and pressing **Ctrl + C** to copy them, then selecting the different material slots and pressing **Ctrl + V** to paste everything; in **Blender Internal**, we have to make the single materials inside the nodes single user one by one and then substitute the textures according to the **UDIM** tile the material corresponds to (**U1V0_col.png**, **U1V0_scales.png**, **U2V0_col.png**, and so on).

So, in the end, each material will have *two sets of nodes*, one for the **Cycles** shader and one for the **Blender Internal** shader, and each one works under the respective render engine; this will be useful, as we'll see in the next chapter, for the rendering stage.



Two different sets of nodes for the same material

See also

- http://www.blender.org/manual/render/blender_render/materials/index.html
- http://www.blender.org/manual/render/blender_render/textures/index.html

Building the eyes' shaders in Blender Internal

We'll now see how to make the shaders for the **Gidiosaurus's eyes**; they are composed of two objects, the **Corneas** and the **Eyes** objects, so let's start with the first one.

Getting ready

Enable the **6th** and the **12th** scene layers and select the **Corneas** object; in the **Outliner**, disable the **Eyes** object's visibility in the viewport to hide it, put the mouse pointer inside the **Camera** view, zoom to one of the **eyeballs** and then start the **Rendered** preview.

How to do it...

Let's start to create the Corneas material:

1. Put the mouse pointer in the **Node Editor** window and add: a **Material** node (*Shift + A | Input | Material*), a **Geometry** node (*Shift + A | Input | Geometry*), a **MixRGB** node (*Shift + A | Color | MixRGB*) and an **Output** node (*Shift + A | Output | Output*).
2. Connect the **Color** output of the **Material** node to the **Color** input socket of the **Output** node, then click on the **New** button on the **Material** node to create a new material and rename it **Cornea_bump**.
3. In the **Material** window, expand the **Render Pipeline Options** subpanel and enable the **Transparency** item; in the **Diffuse** subpanel set the shader model to **Oren-Nayar** and the color to a bright orange = **R 0.930, G 0.386, B 0.082**. In the **Specular** subpanel set the shader model to **WardIso** and the **Slope** to **0.070**. In the **Shading** subpanel enable the **Cubic Interpolation** item.
4. Go down to expand the **Transparency** subpanel; set the **Fresnel** value to **1.380** and the **Blend** to **1.700**.
5. Go further down to enable the **Mirror** item in the subpanel with the same name, set the **Reflectivity** to **0.200**, the **Fresnel** to **1.380** and the **Blend** to **1.500**.
6. Go to the **Texture** window and in the **first** slot load the image **eyeball_col.jpg**, rename the ID datablock as **eyeball_col** and set the **UVMap.001** as the coordinates layer; in the **Influence** subpanel set the diffuse **Color** channel to **0.200** and the specular **Color** channel to **0.200** as well, set the **Blend Type** to **Color**.
7. In the **second** texture slot load the image **eyeball_bump.jpg**, rename the ID datablock as **Eyeball_bump**, set **UVMap.001** as the coordinates layer and disable the **Color** channel to enable the **Normal** one at **0.007**; set the **Bump Method** to **Best Quality**:



The "Corneas" material nodes

8. Paste the **MixRGB** node between the **Material** and the **Output** nodes; then, connect the **Vertex Color** output of the **Geometry** node to the **Fac** input socket of the **MixRGB** node. Click on the last empty field at the bottom of the **Geometry** node to select the **Col** item from the pop-up list (it's the name of the **Vertex Color** layer that has been created, and mentioned, at the beginning of the *Building the eyes' shaders in Cycles* recipe in [Chapter 12, Creating the Materials in Cycles](#)).



The gray color of the Color2 socket of the MixRGB node showing in the rendered preview at the location established by the Vertex Color layer output used as factor

In the preceding screenshot, the effect of the **Vertex Color** layer is visible: the two gray dots on the eyeballs are actually the crystalline lens areas filled, only at the moment, with the gray color of the empty **Color2** input socket of the **MixRGB** node.

9. Press **Shift + D** to duplicate the **Material** node, click on the **2** icon button to the right side of the name datablock of the duplicated node to make the material single user, rename the new material, simply, **Cornea** and, in the **Texture** window, select the **second** slot texture, **eyeball_bump**, to click on the **X** icon button and delete it.
10. Connect the **Color** output of the second **Material** node to the **Color2** input socket of the **MixRGB** node.



The completed "Corneas" material

Now, go to the **Outliner** and enable the **Eyes** object visibility in the viewport to show it:

11. With the **Corneas** object still selected, put the mouse pointer on the first **Material** node in the **Node Editor** window and press **Ctrl + C** to copy it.
12. Select the **Eyes** object and, in the **Material** window, select the **Eyeballs** material slot; put the mouse pointer in the **Node Editor** window and press **Ctrl + V** to paste the material node we copied before.
13. Click on the **2** icon button to make the material single user and rename it **Eyes**. Add an **Output** node (**Shift + A** | **Output** | **Output**) and connect the **Color** output of the **Eyes** material node to the **Color** input socket of the **Output** node.
14. Go to the **Material** window and disable the **Transparency** item in the **Render Pipeline Options** subpanel; go to the **Mirror** subpanel and disable it.
15. Enable the **Subsurface Scattering** subpanel: set the **IOR** to **1.340**, the **Scale** to **0.001**, the scattering color to the orange **R 0.930, G 0.386, B 0.082** and the **RGB Radius** to the **R 9.436, G 3.348, B 1.790** values.



The "Eyeballs" material SSS settings

16. Go to the **Texture** window; select the **eyeball_col** texture and set the diffuse **Color** to **0.855**, disable the specular **Color** channel and set the **Blend Type** to **Linear Light**; select the **eyeball_bump** texture and set the **Normal** channel slider to **0.005**.



The "Eyeballs" material texture settings

Now let's see the **iris**:

17. Box-select both the Eyes material node and the connected **Output** node and press **Ctrl + C** to copy them; go to the **Material** window and select the **Irides** material slot, then put the mouse pointer in the **Node Editor** window and press **Ctrl + V** to paste them.
18. Make the duplicated node's material single user and rename it **Iris**; change the **Diffuse** subpanel color to **R 0.429, G 0.153, B 0.000**, then go to the **Shading** subpanel and set the **Emmit** value to **0.07**. Go to the **Subsurface Scattering** subpanel and change the scattering color to **R 0.220, G 0.033, B 0.032**.
19. Go to the **Texture** window and delete (unlink) the two texture slots. In the **first** slot, load the image **iris_col.jpg**, rename the ID datablock **iris_col**, and set **UVMap.001** as the UV coordinates layer. In the **second** slot, load the image **iris_bump.jpg**, rename as **iris_bump**, set **UVMap.001** as the UV coordinates layer and in the **Influence** subpanel disable the **Color** channel and enable the specular **Intensity** and **Hardness** channels with value **1.000**, then enable also the **Normal** channel with value **1.000**. Set the **Bump Method** to **Best Quality**.
20. In the **third** texture slot, load again the **iris_bump.jpg** image; rename the ID datablock as **iris_ST**. In the **Image Sampling** subpanel, under **Alpha**, disable the **Use** item and enable the **Calculate** item. Set the **UVMap.001** coordinates layer and in the **Influence** subpanel disable the **Color** channel and enable only the **Stencil** item at the bottom:



The "Irides" material and the Stencil item

21. In the **fourth** texture slot, load the **iris_col.jpg** image, rename the ID datablock as **iris_emit**. Set the **UVMap.001** coordinates layer and in the **Influence** subpanel disable the **Color** channel and enable the **Emmit** channel at value **1.000**.



The "Irises" material emitting (fake) light

Regarding the **Pupils** material, it's a simple basic material with pure black as **Diffuse** color and the **Intensity** slider under the **Specular** subpanel set to **0.000** to be totally matte.

22. Save the file.

How it works...

For the **Corneas** object: we created two copies of the same transparent material, but then we removed the bump from one of them, because usually a **cornea** has bumps due to the veins on the **eyeball** but not on the **crystalline lens**, that is smooth; the two materials are mixed, exactly as in the **Cycles** version, through the output of the **Col Vertex Color** layer.

Regarding the **Irises** material: the **iris_ST** texture, set as a **stencil map**, works as a mask for the following texture to appear through its black areas.

Although it could have been solved by simply leaving a blank material slot, I assigned a black matte material to the **pupils**; I preferred to assign a material anyway, to avoid possible issues in the following stages such as, for example, in the rendering of the character against an alpha backdrop, of the separated passes and in the compositing.

Note that the **Corneas** is the only material where I enabled the ray-tracing mirror, which in the character's **skin** and in the **armor** are instead faked, to obtain faster rendering times (the **eyes** are really a small surface to be rendered).

Building the armor shaders in Blender Internal

We arrive finally at making the **Armor** shaders under the **Blender Internal** engine; we have **four** materials, here: the two **UDIM** plate shaders, the rivets shaders and the leather material for the tiers.

Getting ready

Enable the **6th** and the **13th** scene layers and select the **Armor** object; if your computer is powerful enough, use the **Rendered** preview while you are working.

How to do it...

Let's start with the first **UDIM** tile material creation, the main **armor** plates:

1. Put the mouse pointer in the **Node Editor** window and add a **Material** node (*Shift + A | Input | Material*), a **MixRGB** node (*Shift + A | Color | MixRGB*) and an **Output** node (*Shift + A | Output | Output*). In the **N Properties** sidepanel, label the **Material** node as **COL**.
2. Connect the **Color** output of the **COL** node to the **Color** input socket of the **Output** node, then click on the **New** button on the **Material** node to create a new material and rename it **Armor_U0V0_col**; in the **Node Editor** window, disable the **Specular** item.
3. Go to the **Material** window and find the **Diffuse** subpanel; change the shader model to **Oren-Nayar**, set the color to **R 0.817, G 0.879, B 1.000** and the **Roughness** value to **0.313**.
4. Go down to the **Specular** subpanel: set the shader model to **WardIso**, the **Intensity** to **1.000**, the **Slope** to **0.270**, and the color to **R 0.381, G 0.527, B 0.497**. In the **Shading** subpanel enable the **Cubic Interpolation** item.



Starting the "Armor_U0V0" material in Blender Internal

5. Go to the **Texture** window and in the **first** texture slot, load the image `iron_U0V0.png`; rename the ID datablock as `iron_U0V0` and set the **UVMap** coordinates layer, then go to the **Influence** subpanel and enable the diffuse **Intensity** channel at value **1.000**, leave the **Color** channel as it is and change the **Blend Type** to **Multiply**:



Adding the first texture image

6. In the **second** texture slot, load the image `vcol2.png`, rename the ID datablock as `vcol2` and set the **UVMap_norm** UV coordinates layer; go to the **Colors** subpanel, enable the **Ramp** item and set the **Interpolation** to **B-Spline**, then move the black color stop to position **0.245** and the white color stop to position **0.755**. In the **Image Sampling** subpanel set the **Filter Size** to **1.10** and in the **Influence** subpanel enable the diffuse **Intensity** channel at value **0.500**, the diffuse **Color** channel at **0.300**, set the **Blend Type** to **Difference** and enable the **Negative** item.
7. In the **third** texture slot, load the image `Ice_Lake_Ref.hdr` and rename the ID datablock as `env_refl_armor`. Set the **Mapping** coordinates to **Reflection** and, in the **Image Sampling** subpanel, the **Filter Size** to **6.00**; in the **Colors** subpanel set the **Brightness** to **1.800** and the **Contrast** to **2.000**, then move to the **Influence** subpanel and set both the diffuse **Intensity** and **Color** channels to **0.600** and the **Blend Type** to **Multiply**:



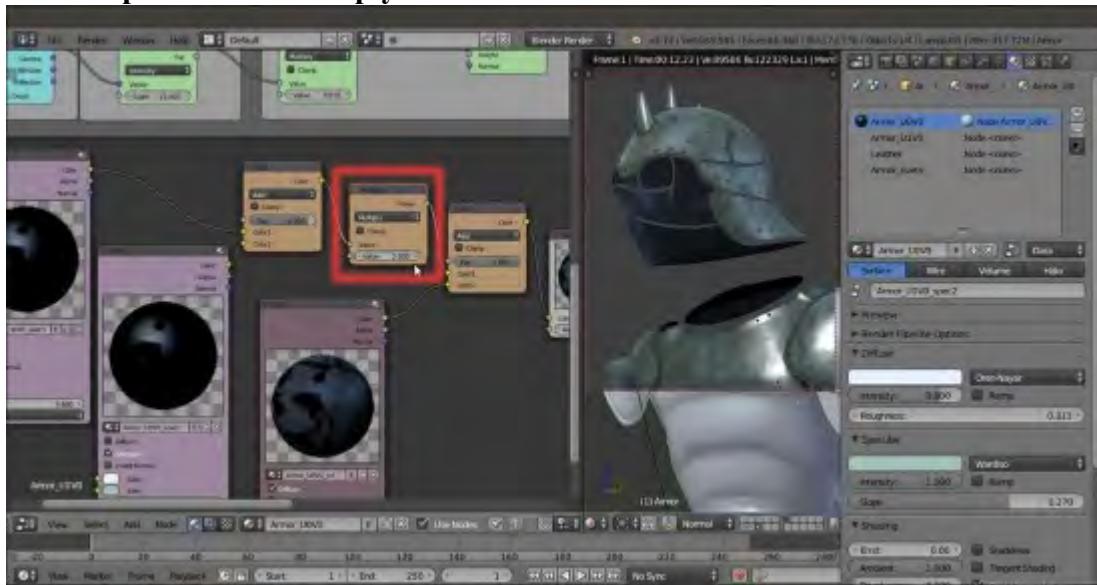
Adding the hdr image as reflection map

8. Go to the **Node Editor** window and press **Shift + D** to duplicate the **COL** node; label the duplicate as **SPEC1**, then make the material single user and rename it **Armor_U0V0_spec1**; disable the **Diffuse** item on the node interface and enable back, the **Specular** one.
9. Go to the **Texture** window; select the first **iron_U0V0** texture slot and go straight to the **Influence** subpanel: disable the diffuse **Intensity** and **Color** channels and enable the specular **Intensity**, **Color** and **Hardness** channels at **1.000**. Set the **Blend Type** to **Mix**.
10. Select the **second** **vcol2** texture slot, disable the diffuse **Intensity** and **Color** channels and enable the specular **Intensity** channel at **0.300**.
11. Select the **third** **env_refl_armor** texture slot, disable the diffuse **Intensity** and **Color** channels and enable the specular **Intensity** and **Color** channels at **0.500**.
12. Press **Shift + D** to duplicate the **SPEC1** node and label the duplicated one as **SPEC2**: make the material single user and rename it as **Armor_U0V0_spec2**. In the **Material** window go to the **Specular** subpanel and set the **Slope** to the maximum = **0.400**.
13. Now, connect the **Color** output of the **SPEC1** material node to the **Color1** input socket of the **MixRGB** node and the **Color** output of the **SPEC2** node to the **Color2** input socket; set the **Blend Type** of the **MixRGB** node to **Add** and the **Fac** value to **0.900**.
14. Press **Shift + D** to duplicate the **MixRGB** node and connect the output of the first **MixRGB** node to the **Color1** input socket of the duplicated **MixRGB** node, and the **Color** output of the **COL** node to the **Color2** input socket; set the **Fac** value of the second **MixRGB** node to **1.000** and connect its output to the **Color** input socket of the **Output** node.



Adding the specular component

- Add a **Math** node (**Shift + A | Converter | Math**) and paste it between the two **MixRGB** nodes; set the **Operation** to **Multiply** and the second **Value** to **2.000**.



Enhancing the specularity

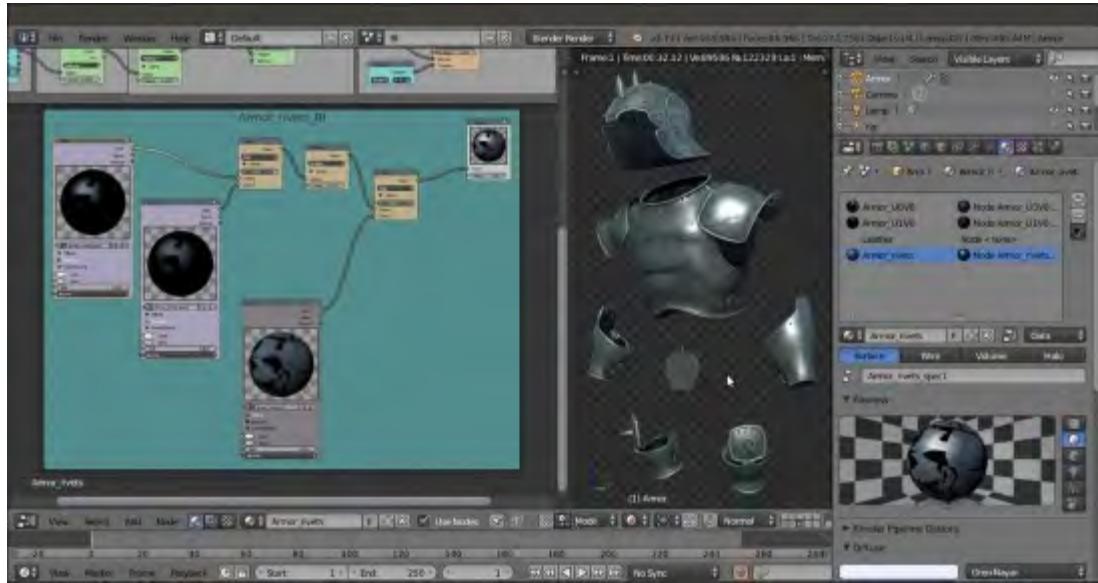
- Add a **Material** node (**Shift + A | Input | Material**) and label it as **BUMP**; create a new material and rename it as **Armor_U0V0_normals**; in the **Shading** subpanel enable the **Cubic Interpolation** item.
- Go to the **Texture** window and in the **first** texture slot, load the image **norm2.png**, rename the ID datablock as **norm2** and in the **Image Sampling** subpanel enable the **Normal Map** item; in

- the **Mapping** subpanel set the **UVMap_norm** coordinates layer and in the **Influence** subpanel disable the **Color** channel and enable the **Normal** one at **0.500**.
18. In the **second** texture slot, load the image **iron_U0V0.png**, mapping to **UVMap** layer and **Influence to Normal** at **0.010**; set the **Bump Method** to **Best Quality**.
 19. Go to the **Node Editor** window and connect the **Normal** output of the **BUMP** node to the **Normal** input sockets of the **SPEC1**, **SPEC2**, and **COL** nodes.



Adding the bump pattern

20. Box-select and press **Ctrl + C** to copy all these nodes, go to the **Material** window to select the **Armor_U1V0** material slot and, back in the **Node Editor** window, paste the copied nodes: then make the materials inside the nodes as single users, rename them accordingly and go to the **Texture** window to substitute the **iron_U0V0.png** image with the **iron_U1V0.png** image.
21. Copy and paste again, the nodes for the **Armor_rivets** material slot, but don't substitute the texture image: instead, simply delete the **BUMP** node, which wouldn't be of any use in such small parts.



The completed armor shaders in Blender Internal

22. Save the file.

How it works...

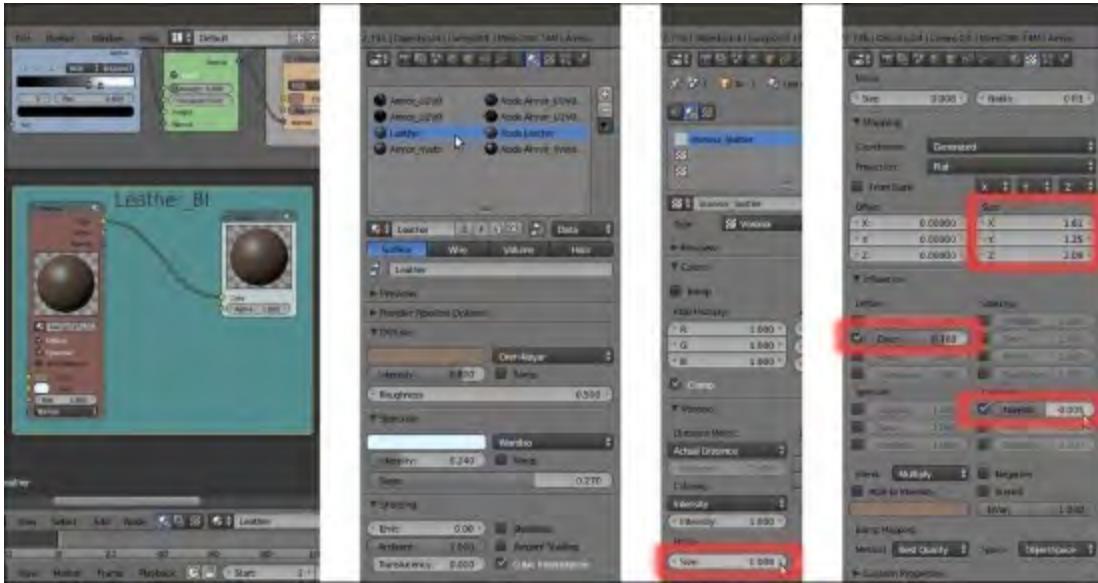
These shaders work, and have been built, exactly the same way as for the **Gidiosaurus'** skin and eyes; the only thing worth noting here is the order of the normal map and of the texture used for the bump pattern: in fact, to work together, in **Blender Internal**, the normal map must be placed higher in the texture stack, otherwise it will overwrite the effect of the bump map.

There's more...

The Leather material is a simple basic **Oren-Nayar** diffuse shader with the usual **WardIso** specular shader model, provided with a bump effect obtained through a **Voronoi** procedural texture with default values, except for the **Size**.

Note that the **Voronoi** texture influences the **Color** channel with the **Multiply** blend type and the **Normal** channel with a **negative** low value to obtain an actual bulging out pattern, instead of a concave one; negative values, in fact, reverse the *direction* of the bump.

The size of the procedural texture along the three axes is also further tweaked in the **Mapping** subpanel, scaling the three axes differently to resemble the dimensions of the **Texture Space** (basically the mesh bounding box, than can be made visible through the subpanel of the same name in the **Object Data** window), in order to avoid stretching along the mesh.



The "Leather" material in Blender Internal

Note also that in all these **Blender Internal** materials, simple mono materials (as for example the Leather BI material shown here earlier) are loaded inside a **Material** node and then connected to an **Output** node in the **Node Editor** window even if this wouldn't be necessary for the material itself to work: but, to let the **Blender Internal** and the **Cycles** render engines work together (through the compositor, as we'll see in the next chapter), it is mandatory to have all the shaders as nodes.

Chapter 14. Lighting, Rendering, and a Little Bit of Compositing

In this chapter, we will cover the following recipes:

- Setting the library and the 3D scene layout
- Setting image based lighting (IBL)
- Setting a three-point lighting rig in Blender Internal
- Rendering an OpenGL playblast of the animation
- Obtaining a noise-free and faster rendering in Cycles
- Compositing the render layers

Introduction

In this last chapter, we are going to see recipes about the more common stages needed to render the complete final animation: lighting techniques in both the render engines, fast rendering previews, rendering settings, and the integrated compositing.

But first, let's see the necessary preparation of the 3D scene layout.

Setting the library and the 3D scene layout

In this recipe, we are going to prepare a little both the file to be used as the library and the *hero* blend file, which is the file that will output the final rendered animation.

Getting ready

Start Blender and load the `Gidiosaurus_shaders_Blender_Internal.blend` file:

1. Go to the **Object Modifiers** window and check that the **Armature** modifiers are correctly enabled for *all the objects* (that is, the **Armature** modifiers must be enabled both for the rendering and for the 3D viewport visibility), then save the file.
2. Press *Ctrl + N* and click on the **Reload Start-Up File** pop-up panel to confirm a new brand file: immediately save it as `Gidiosaurus_3D_layout.blend`.

Tip

Saving the file at this point is necessary to automatically have a *relative path* for all the assets we are going to link.

How to do it...

Let's load the assets as links in the file:

1. Select and delete (*X* key) the default **Cube** primitive in the middle of the 3D scene, then *Shift*-select both the **Camera** and the **Lamp** and move them (*M* key) to the **6th** scene layer.
2. Still in the **1st** scene layer, click on the **File** item in the top main header and then navigate to select the **Link** item; or else, just press the *Ctrl + Alt + O* keys shortcut.

In the blend files provided with this cookbook, I moved a copy of the `Gidiosaurus_shaders_Blender_Internal.blend` file to the `4886OS_14_blendfiles` folder, to simplify the process, but anyway:

3. Browse to the folder where the `Gidiosaurus_shaders_Blender_Internal.blend` file is saved; click on it and browse further to click on the **Group** item/folder, then select the **Gidiosaurus** item and click on the top right **Link from Library** button.

The linked **Gidiosaurus** character appears at the **3D Cursor** position, in our case in the middle of the scene:



Linking the Gidiosaurus group

4. Zoom to the **Gidiosaurus** object and press *Ctrl + Alt + P* to make a proxy; in the pop-up menu panel that appears, select the proxied **rig** item, then *Shift*-enable the **11th** scene layer and move the **rig** on that scene layer (*M* key):



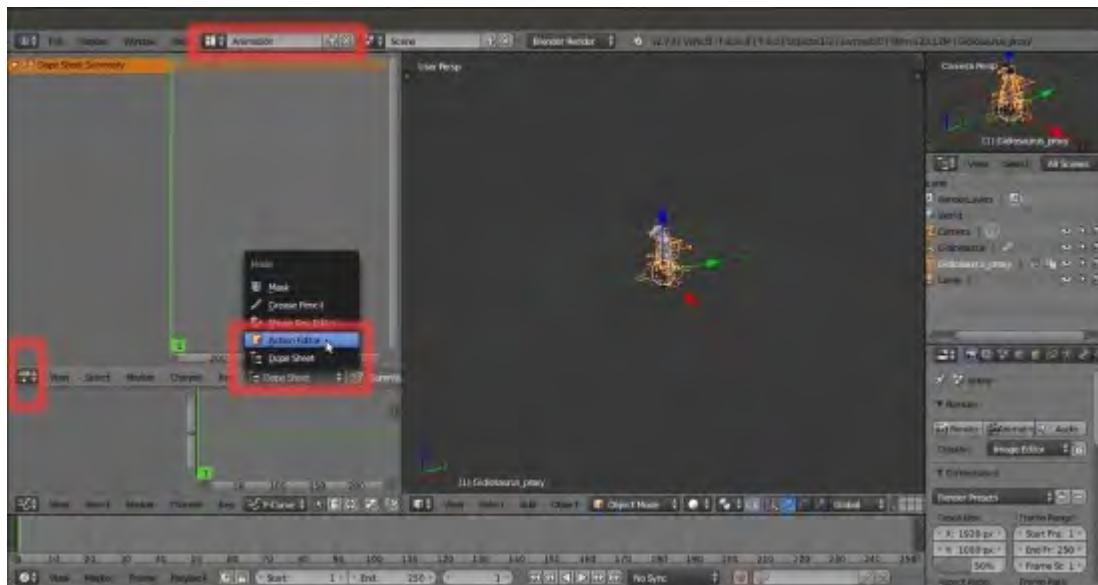
Making a proxy of the rig and moving it to the 11th scene layer

5. Click on the *Screen datablock* button on the top main header to switch from the **Default** screen layout to the **Animation** screen layout:



Switching to the Animation screen layout

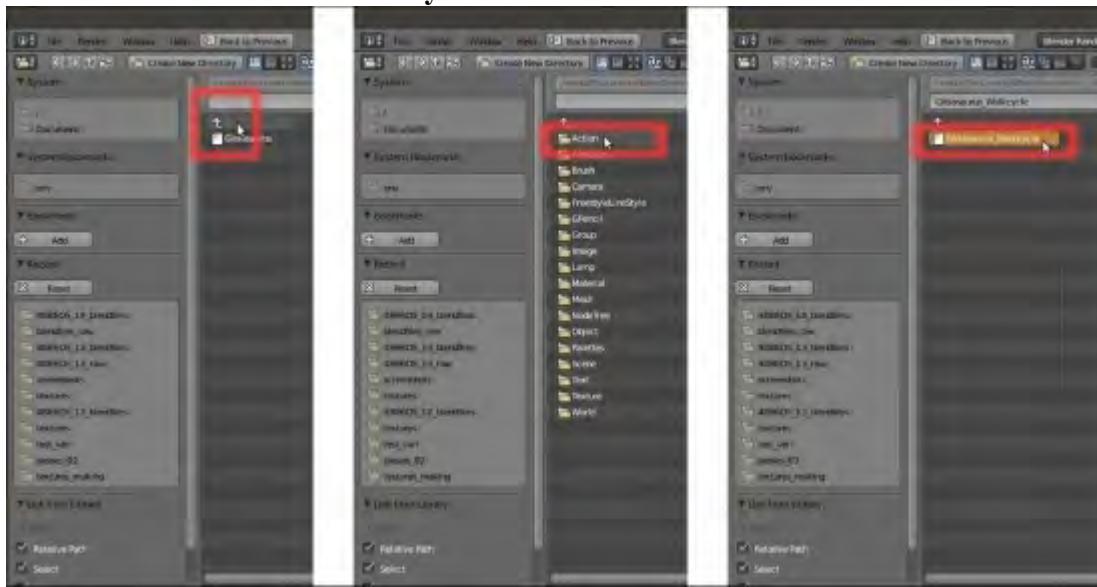
6. Click on the **Mode** button in the Dope Sheet toolbar and switch from the **Dope Sheet** window to the **Action Editor** window:



Switching to the Action Editor window

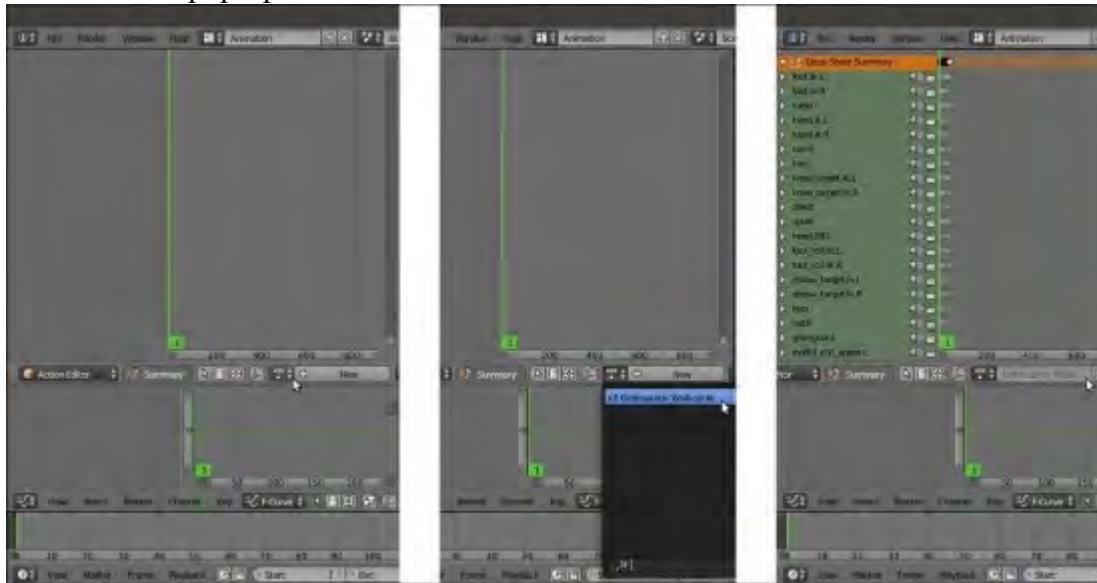
7. Click on the **File** item in the top main header and then navigate to select the **Link** item, or just press the **Ctrl + Alt + O** keys shortcut; the screen opens automatically at the last location we previously browsed to.

8. Click on the two dots above the **Gidiosaurus** item to navigate backward (to go up one level) and click on the **Action** item/folder to select the **Gidiosaurus_walkcycle** item; then click as before on the **Link from Library** button:



Browsing to the Action folder directory

9. Scroll a bit to the left of the **Action Editor** window's toolbar to reveal the **New** button; click on the double arrows to the left side of the **New** button to select the **LF Gidiosaurus_walkcycle** item from the pop-up menu.



Loading the linked action in the Action Editor window

10. Go back to the **Default** screen and click on the **Play Animation** button in the **Timeline** toolbar.

Depending on the power of your system, you will see the animated character start to move, more or less fluidly, in the 3D viewport; the frame-rate (number of frames per second) played by Blender in real-time is shown in red at the top left corner of the 3D view-port:



The 3D view-port showing the animation and the frame-rate at the top left corner

It should be around **24** frames per second; in my case, it barely arrives at **0.70** to **0.80...** so an arrangement must be found to show a faster and natural-looking movement.

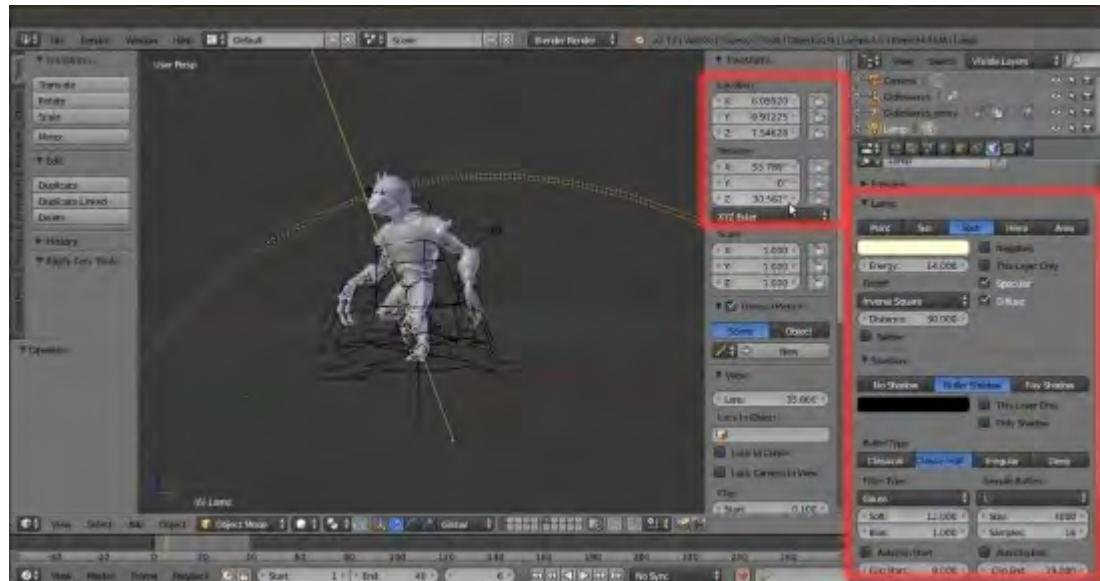
11. Go to the **Scene** window and enable the **Simplify** subpanel: set the **Subdivision** level to **0**.



The Simplify subpanel under the Scene window in the main Properties panel

Without subdivision levels, even on my old laptop the real-time frame-rate is now **24** frames per second.

12. Go to the **Render** window and, in the **Dimensions** subpanel, under **Frame Range**, set the **End Frame** to **40**; under **Resolution** switch the X and Y values, that is **X = 1080 px** and **Y = 1920 px**.
13. Go to the **Outliner** and click on the **Display Mode** button to switch from **All Scenes** to **Visible Layers**. Then *Shift*-enable the **6th** scene layer and select the **Lamp**.
14. Go to the **Object Data** window and, in the **Lamp** subpanel, change the lamp type from **Point** to **Spot**; set the color to **R 1.000, G 1.000, B 0.650** and the **Energy** to **14.000**.
15. Put the mouse pointer in the 3D viewport and press **N** to call the side **Properties** panel; go to the top **Transform** subpanel and set **Location** as **X = 6.059204, Y = -9.912249, Z = 7.546275** and **Rotation** as **X = 55.789°, Y = 0°, Z = 30.562°**.
16. Back in the **Object Data** window, go down to the **Shadow** subpanel and switch from **Ray Shadow** to **Buffer Shadow**; under **Filter Type** set the **Shadow Filter Type** to **Gauss**, the **Soft** to **12.000**, the **Size** to **4000** and the **Samples** to **16**. Set the **Clip Start** value to **9.000** and the **Clip End** value to **19.000**.



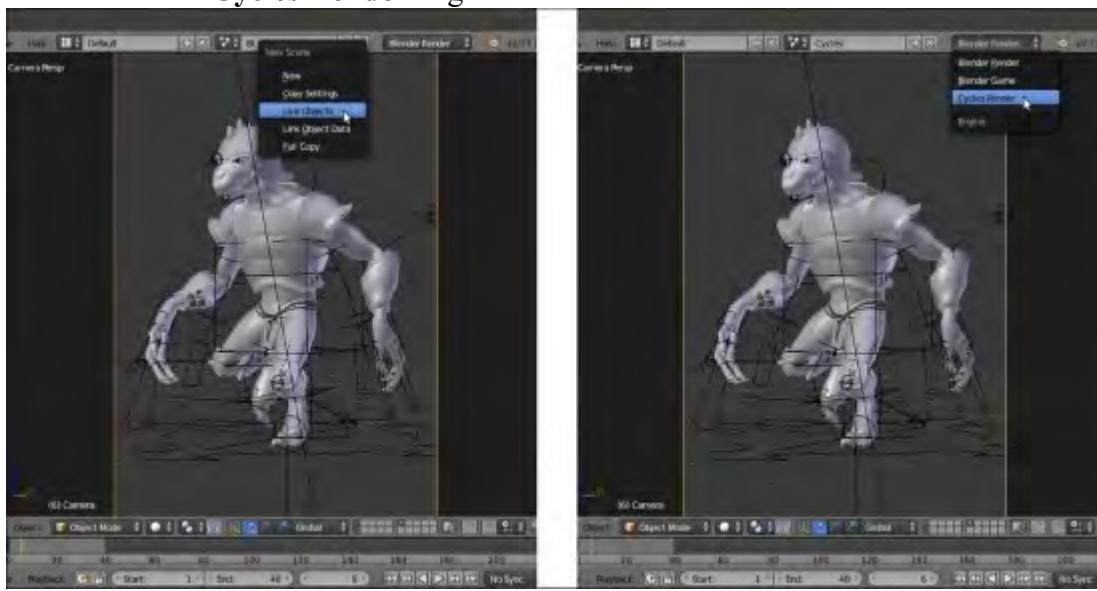
Setting the Lamp

17. Select the **Camera** and in the **Object Data** window set the **Focal Length** under **Lens** to **60.00**; in the **Transform** subpanel under the **N** side **Properties** panel input these values: **Location** as **X = 5.095385, Y = -6.777483, Z = 1.021429** and the **Rotation** as **X = 91.168°, Y = 0°, Z = 37.526°**. Put the mouse pointer in the 3D viewport and press the **0** numpad key to go in **Camera** view:



Setting the Camera

18. Now click on the **Scene** datablock button on the top main header and rename it **BI (Blender Internal)**. Click on the + icon button to the right and from the **New Scene** pop-up menu select the **Link Objects** item.
19. Change the **BI.001** name of the new scene in **Cycles** and click on the *Engine* button to the right to switch to the **Cycles Render** engine:



Adding a new scene with linked objects

20. Go to the **Outliner** and select the **Lamp**; go to the **Object Data** window and, in the **Nodes** subpanel, click on the **Use Nodes** button and then set the **Strength** to **10000.000**. Go to the **Lamp** subpanel and set the **Size** to **0.500** and enable the **Multiple Importance** item.
21. Save the file.

How it works...

A scheme of what we made in this recipe is: we prepared a blend file that *links* both the **character** and the **action**; this means that *neither* of them is *local* to the file and they cannot be directly edited in this file. Moreover, the character, in its library file, links the textures that are contained in the **textures** folder (which is at the same level as the blend files).

The **Simplify** subpanel in the **Scene** window allows us to *globally* modify some of the settings that can usually slow a workflow, such as the **subdivision** levels, the number of **particles**, the quality of **ambient occlusion** and **subsurface scattering**, and the **shadows** samples; through this panel they can be temporarily lowered or even disabled to have faster and more responsive previews of the rendering and the animation. Just remember that the **Simplify** subpanel also affects the rendering, so you have to disable it before starting the final rendering task.

See also

- http://www.blender.org/manual/data_system/introduction.html#copying-and-linking-objects-between-scenes
- http://www.blender.org/manual/data_system/scenes.html
- http://www.blender.org/manual/data_system/linked_libraries.html

Setting image based lighting (IBL)

The **image based lighting** technique is almost essential in computer graphics nowadays; as the name itself says, it's a technique to light a scene based on the pixel color information of an image, usually an **hdr** image (**High Dynamic Range** image); other image formats can also work, although not so well.

In Blender it's possible to obtain **IBL** both in **BI** and in **Cycles**, although with different modalities.

Getting ready

Start Blender and load the previously saved `Gidiosaurus_3D_layout.blend` file; save it as `Gidiosaurus_IBL.blend`.

How to do it...

We can divide this recipe into two parts: **IBL in Cycles** and in **Blender Internal**.

Image based lighting in Cycles

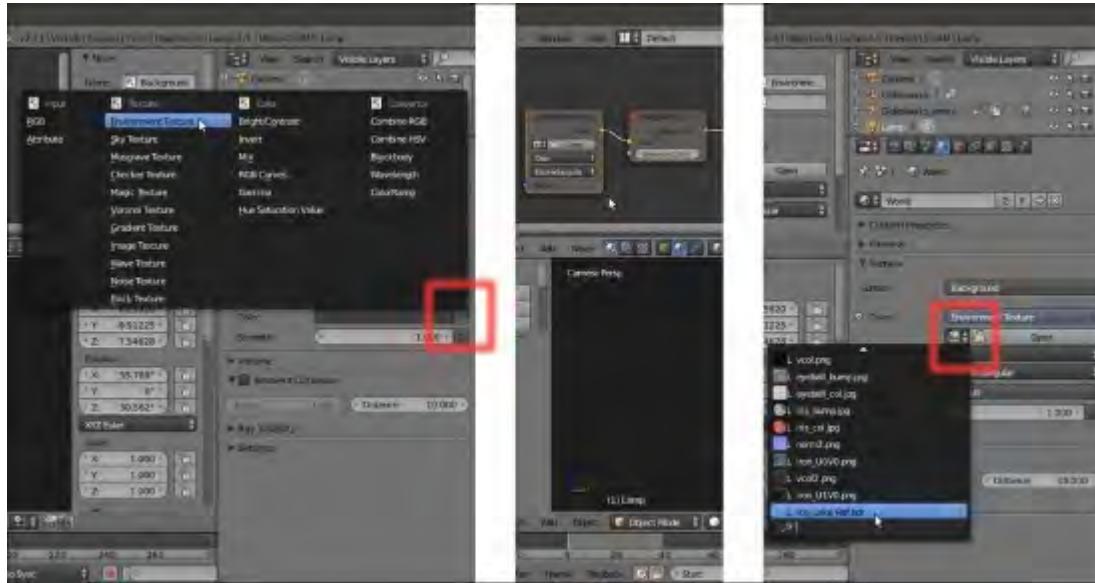
Let's start with the **Cycles Render** engine:

1. First, split the 3D window vertically into two windows, then change the upper one into a **Node Editor** window. In the toolbar, click on the **World** icon button to the right side of the **Object** icon button (selected by default; it's the one enabled for building the objects' shaders). Check the **Use Nodes** checkbox (or, click on the **Use Nodes** button inside the **Surface** subpanel in the **World** window); a **Background** node connected to a **World Output** node will appear in the **Node Editor** window.



Enabling the World nodes in the Node Editor window

- Click on the dotted button to the right side of the **Color** slot in the **Surface** subpanel under the **World** window, to call the pop-up menu and select an **Environment Texture** node, which is automatically added and correctly connected to the **Color** input socket of the **Environment** node; then, click on the double arrows to the left side of the **Open** button (both in the **Node Editor** or in the **World** window) and select the **L_Ice_Lake_Ref.hdr** item.



Adding an Environment node to the World and loading the hdr image

- In the **World** window or in the **Node Editor** window, set the **Color Space** to **Non-Color Data**.
- In order to gain some feedback, start the **Rendered** preview in the bottom **Camera** view, then go back to the **Node Editor** and add a **Texture Coordinate** node (**Shift + A | Input | Texture Coordinate**) and a **Mapping** node (**Shift + A | Vector | Mapping**).
- Connect the **Generated** output of the **Texture Coordinate** node to the **Vector** input socket of the **Mapping** node and the output of this latter node to the **Vector** input socket of the **Environment Texture** node; set the **Rotation Z** value of the **Mapping** node to **-235**.



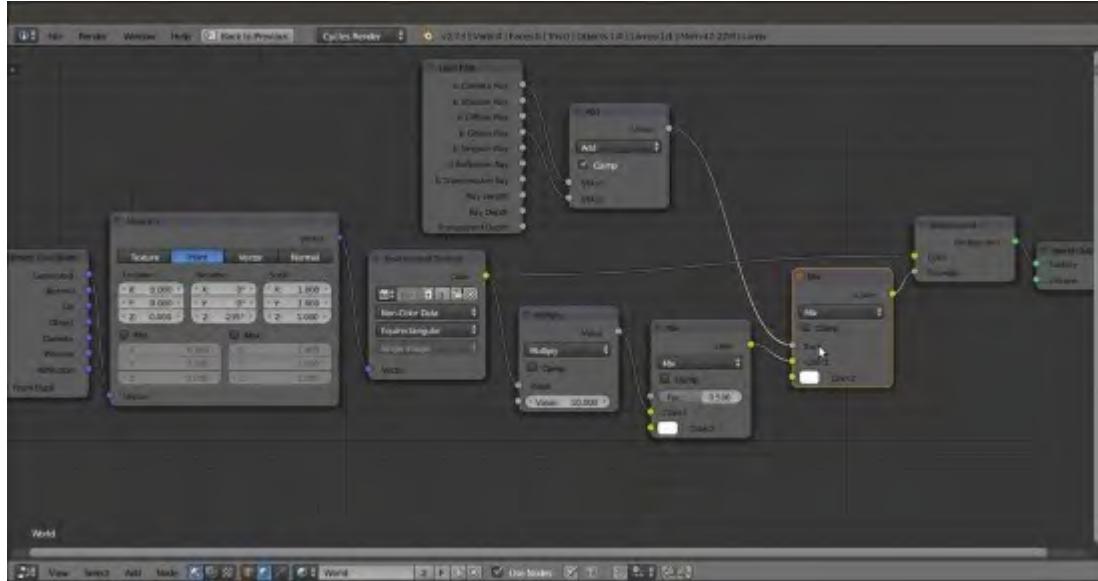
Rotating the hdr image to match the position of the Lamp

6. Now add a **Math** node (*Shift + A | Converter | Math*) and a **MixRGB** node (*Shift + A | Color | MixRGB*); connect the **Color** output of the **Environment Texture** node to the first **Value** input socket of the **Math** node, set the **Operation** of this latter node to **Multiply** and the second **Value** to **10.000**.
7. Connect the output of the **Multiply-Math** node to the **Color1** input socket of the **MixRGB** node and set the **Color2** to pure white; connect the **Color** output of the **MixRGB** node to the **Strength** input socket of the **Background** node:



Adding nodes to the World

- Press **Shift + D** to duplicate both the **Math** and the **MixRGB** nodes: paste the duplicated **MixRGB** node between the first **MixRGB** and the **Background** nodes; set the **Operation** of the duplicated **Math** node to **Add**.
- Add a **Light Path** node (**Shift + A | Input | Light Path**); connect its **Is Camera Ray** output to the first **Value** input socket of the duplicated **Add-Math** node and the **Is Glossy Ray** output to the second **Value** input socket; connect the **Value** output of the **Add-Math** node to the **Fac** input socket of the second **MixRGB** node and enable the **Clamp** item:



The completed IBL World setup for the Cycles render engine

- Go to the **Settings** subpanel and enable the **Multiple Importance** item, then click on the **World datablock** to change the name in **World_Cycles**.
- Go to the **Render** window and in the **Film** subpanel enable the **Transparent** item.



Renaming the World and some more settings in the main window

12. Save the file.

Image based lighting in Blender Internal

Now let's see the same thing in **Blender Internal**:

1. Click on the *Scene datablock* button in the top main header to switch from **Cycles** to **BI**.
2. In the **World** window to the right, click on the **2** icon button to the right side of the **World** name datablock to make it single user, then rename it **World_BI**.
3. Go directly to the **Texture** window: click on the **New** button, then click on the double arrows to the side of the image datablock to select the **L_Ice_Lake_Ref1.hdr** item from the pop-up menu:



Selecting the hdr image in the Blender Internal World

4. Rename the *ID name datablock* to **Ice_Lake_Ref1**, then go down to the **Mapping** subpanel and click on the **Coordinates** slot to select the **Equirectangular** item; set the **Offset** to **X = 0.80500** and then go further down to the **Influence** panel and enable the **Horizon** item.



First BI World settings

5. Back in the **World** window, in the **World** subpanel enable the **Real Sky** item.
6. Enable the **Environment Lighting** subpanel and click on the **Environment Color** button to select the **Sky Texture** item.
7. In the **Gather** subpanel, enable the **Approximate** method, the **Pixel Cache** item, the **Falloff** item and set the **Strength** value of this latter item to **0.900**.



More BI World settings

8. Go to the **Render** window and in the **Shading** subpanel click on the **Alpha Mode** slot to switch from the **Sky** to the **Transparent** item:



Enabling the transparent background for the rendering

9. Save the file.

How it works...

In **Cycles**: at steps 6 and 7 we added nodes to increase the source light intensity of the **hdr** image; because this also increased the contrast of the image, at steps 8 and 9 we made it less contrasted again but kept the same light intensity, thanks to the **Light Path** node. The light rays shoot from the **Camera** position and directly hit a surface (**Is Camera Ray**) or any glossy surface (**Is Glossy Ray**) and have value = **1.000**, hence corresponding to the **Color2** socket of the second **MixRGB** node, therefore giving a pure white (**1.000**) value to the **Background** node's **Strength**; any other ray (transmitted, shadows, reflected, transparent, and so on) has the high contrast **Strength** values we established at steps 6 and 7.

We used the **Mapping** node for the sole reason of matching (visually and thanks to the **World Background** item enabled in the **Display** subpanel under the **N** side **Properties** panel) the source light direction of the image with the position of the **Lamp** in the 3D scene: that's why we rotated the **hdr** image to negative **235** degrees on the **z** (vertical) axis.

In **Blender Internal**: we can't rotate the image, so instead we offset it on the **x** axis to (almost perfectly) match the position it has in **Cycles**.

The **Approximate** gathering method is the one developed during the production of the short open movie **Big Buck Bunny** (<https://peach.blender.org/>) to have faster rendering and absence of noise in **Ambient Occlusion**, inevitable with the default **Raytrace** method (that still remains the more accurate, by the way).

Note that, in both the render engines, we didn't load a brand new **Ice_Lake_Ref.hdr** image from the **textures** folder, but we instead used the linked one coming from the materials of the character, as

indicated by the L in front of the name and by the name itself and all the settings grayed in the image datablock subpanel.

See also

The free **sIBL** addon currently, only works with **Cycles** materials but it can read the .ibl file provided with the free **hdr** images at the **sIBL Archive** (link provided further) and therefore, in one click, it can create the complete nodes setup to provide image based lighting in Blender.

- The official documentation about the addon (http://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Import-Export/sIBL_GUI)
- An updated and bug-fixed version of the addon (<https://raw.github.com/varkenvarken/blenderaddons/master/sibl.py>)
- The sIBL archive (<http://www.hdrlabs.com/sibl/archive.html>)
- Official documentation about the **World** in Blender:
 - <http://www.blender.org/manual/render/cycles/world.html>
 - http://www.blender.org/manual/render/blender_render/world/index.html

Setting a three-point lighting rig in Blender Internal

Thanks to the global illumination, a path-tracer like **Cycles** doesn't necessarily need big lighting setups; in fact, in the recipes we made, we only used one single **Spot** lamp in addition to the **IBL** and the results have been quite good anyway.

In **Blender Internal**, instead, a minimum arrangement of lamps must be done to obtain satisfying results, even with the aid of the **World** settings we have previously seen.

In this recipe, we are therefore going to see a classic *movie* three-point lighting rig, an industry standard. The effect of the main **key light** is enhanced by the other two lamps: the **fill light**, to brighten (and color) the shadow areas on the subject, and the **backlight**, to create a light rim on the subject edges thus making it stand out against the background.

Getting ready

Start Blender and load the previously saved `Gidiosaurus_IBL.blend` file; if necessary, switch to the **Blender Render** engine by the *Engine to use for rendering* button in the top main header.

1. Put the mouse pointer inside the **Camera** view and press the numpad 7 key to go in **Top** view, then press numpad 5 to switch from **Perspective** to **Ortho** view.
2. Press **Shift + C** to put the **3D Cursor** at the center of the scene and then go to the **Outliner** to select the **Gidiosaurus** item: press the numpad period (.) key to center and zoom the view on the selected object:



Centering the top view on the character

3. Press **Ctrl + Spacebar** to disable the widget and scroll the mouse wheel to zoom backward and show the **Spot** lamp, then press the dot (.) key to switch the **Pivot Point** from **Median Point** (or whatever else) to **3D Cursor**.
4. Select the **Lamp**, then go to the **Object Data** window.
5. Remember to go to the **Scene** window and disable the **Simplify** subpanel!
6. Save the file as **Gidiosaurus_lighting.blend**.

Don't take into consideration the **Node Editor** window at the top showing the **Lamp** nodes under **Cycles**; the settings to look for are those inside the main **Properties** panel to the right:

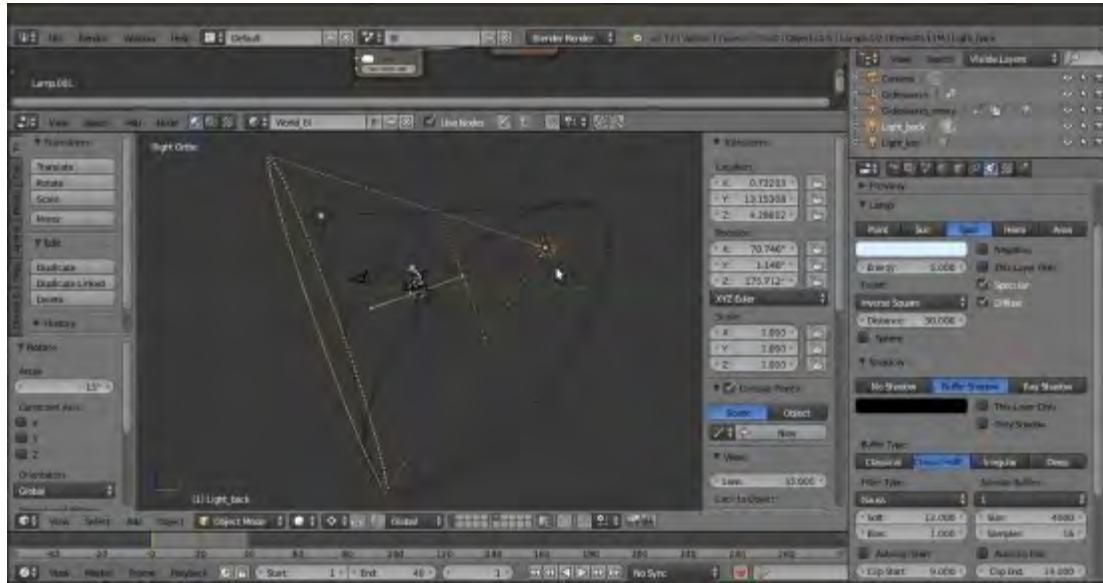


These are the settings you are looking for...

How to do it...

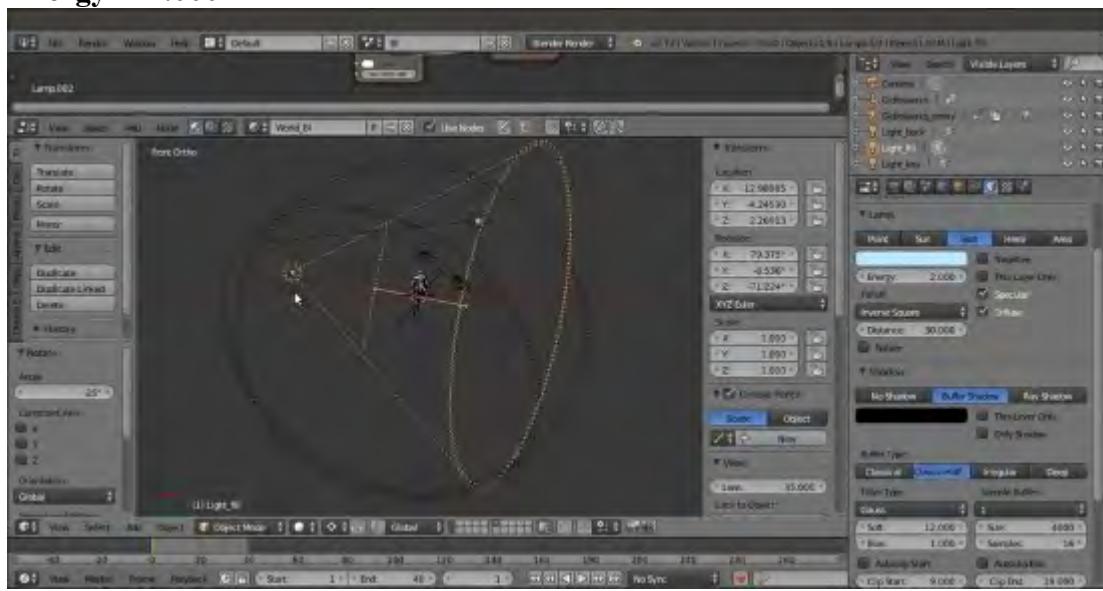
Let's go with the settings of the lights:

1. In the **Outliner**, rename the **Lamp** item as **Light_key**.
2. Press **Shift + D** to duplicate the **Key_light** lamp, press **R** and, while still in **Top** view, rotate the duplicated lamp approximately **-145** degrees, then go in **Side** view (numpad 3 key) and rotate it around **15** degrees: in the **Outliner**, rename it as **Light_back**.
3. In the **Object Data** window, set the color to a light blue = **R 0.700, G 0.900, B 1.000** and the **Energy** to **5.000**:



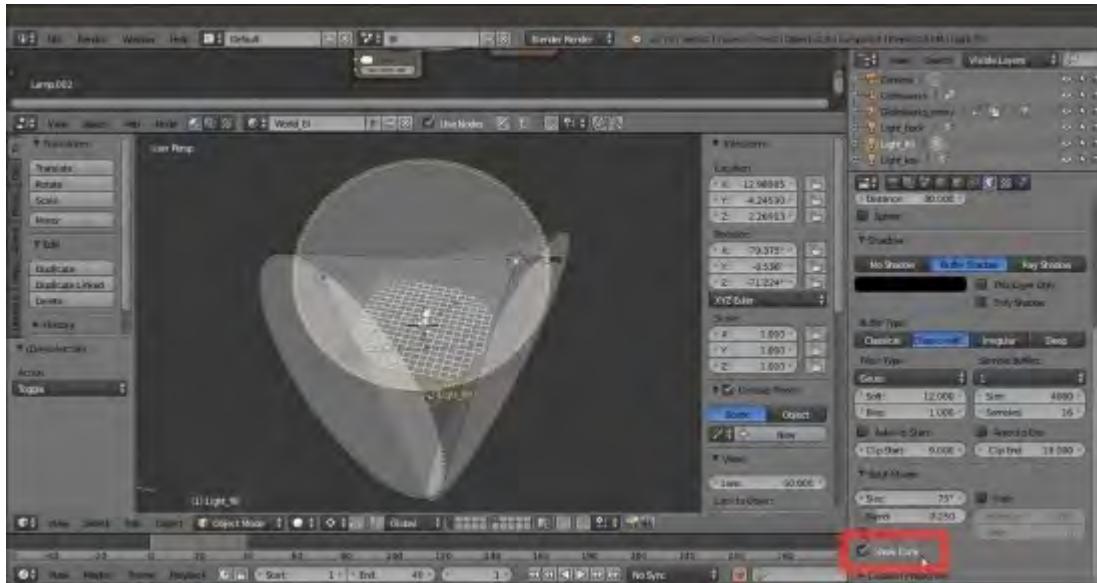
Positioning the Light_back lamp

4. Go back in **Top** view (numpad 7 key) and re-select the **Light_key** lamp, press *Shift + D* and rotate the duplicate by **100** degrees; in the **Outliner**, rename it as **Light_fill**. Go in **Front** view (numpad 1 key) and rotate it around **-25** degrees.
5. In the **Object Data** window, set the color to a lighter blue = **R 0.500, G 0.800, B 1.000** and the **Energy** to **2.000**:



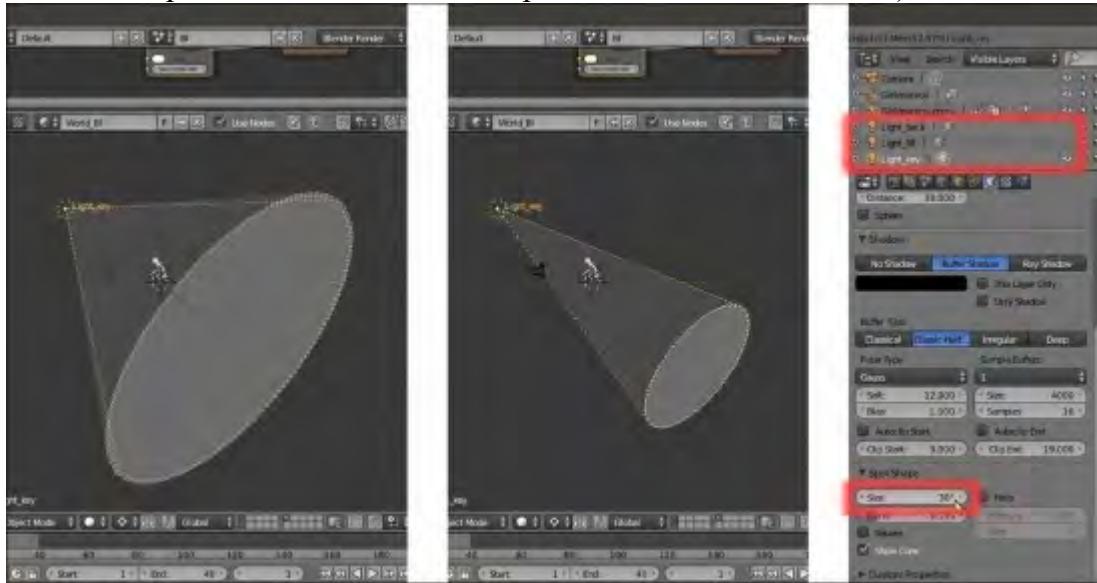
Positioning the Light_fill lamp

6. Go to the **Object** window and in the **Display** subpanel enable the **Name** item for the three lamps; then, back to the **Object Data** window and in the **Spot Shape** subpanel, enable the **Show Cone** item for each one:



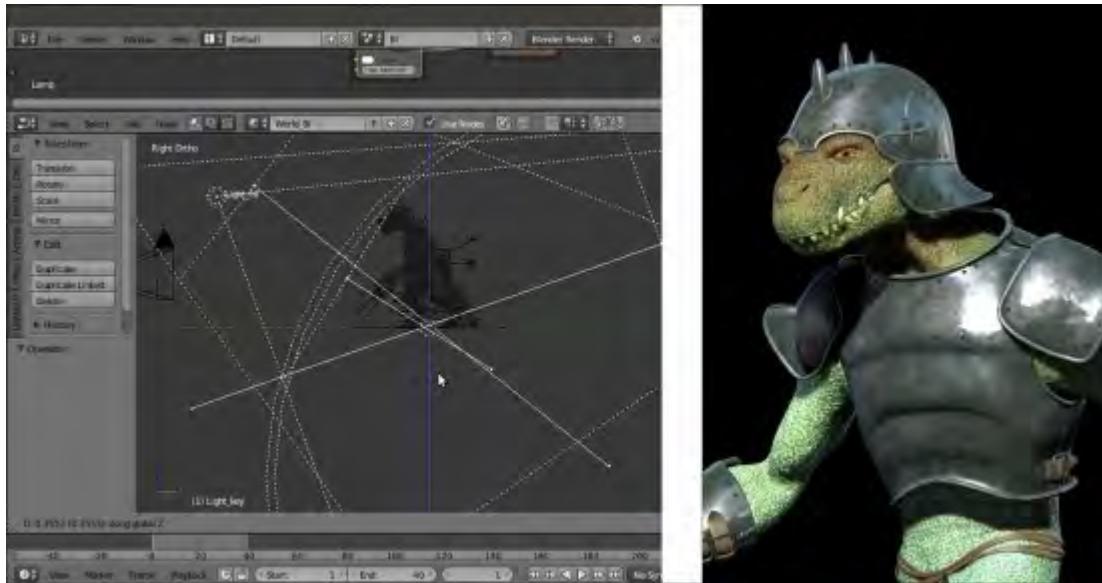
The three spot lamps showing their cones of influence

7. In the **Outliner**, disable the 3D viewport visibility of the **Light_back** and **Light_fill** lamps by clicking on the respective eye icon, then go in **Side Ortho** view and select the **Light_key** lamp.
8. Go to the **Spot Shape** subpanel again and lower the **Size** value from the default 75° to 30° (or the smallest possible value that still comprehends the whole character):



Lowering the spot lamp size value

9. Repeat steps 7 and 8 for the other two lamps as well, then *Shift-select* all the three lamps and move them upward (on the *z* axis) a bit, just to better center the light cones' centers on the position of the feet of the character.
10. Save the file.



The final result of the three-point lighting rig

How it works...

A classic three-point lighting rig can in some way compensate for the lack of real global illumination in **Blender Internal**, although to obtain really good results, three lamps are usually not enough; in any case, the lighting rig of this recipe can be used as a base for even more complex setups.

When using more than one lamp in **Blender Internal**, we should always be sure that the shadows are enabled for all of them, unless we want particular effects; in fact, a back lamp with disabled shadows can easily *shine* through the model and also illuminate parts that shouldn't be in light, giving unrealistic results.

To calculate the buffered shadows, **Spot** lamps take into consideration everything inside their cone from the **Clip Start** to the **Clip End** values; this is why we lowered the **Size** values of the cones as much as possible.

One other crucial factor that can slow the calculation and the rendering times is, obviously, the size of these buffers, which we set to **4000** for each one of the three lamps; quite big, but because we set the cones that large enough to just comprehend the shape of their target object. This means we could use big shadow buffers, to obtain more details in the shadows if needed.

We do all of this, even though the **Gidiosaurus** was the only object to be rendered in the scene.

See also

- http://www.blender.org/manual/render/blender_render/lighting/index.html

Rendering an OpenGL playblast of the animation

Playblast is a term used by a famous commercial package to indicate the preview of the animation in true speed; although I've heard only very few people using it in relation to Blender, I thought it might be a good way to indicate the fast OpenGL preview rendering obtained for checking the animated action.

Getting ready

Start Blender and load the `Gidiosaurus_lighting.blend` file.

1. In the **Outliner**, select the **Light_key** lamp item and go to the **Object Data** window, under the **Spot Shape** subpanel, to disable the **Show Cone** item.
2. Repeat the procedure for the **Light_back** and **Light_fill** lamps, then disable their visibility in the 3D viewport by clicking on the respective eye icon.
3. Disable the visibility in the viewport for the **Gidiosaurus_proxy** item (the linked and proxified rig) also and/or disable the **11th** scene layer.
4. Save the file as `Gidiosaurus_playblast.blend`.

How to do it...

Here are the steps to begin with the OpenGL rendering:

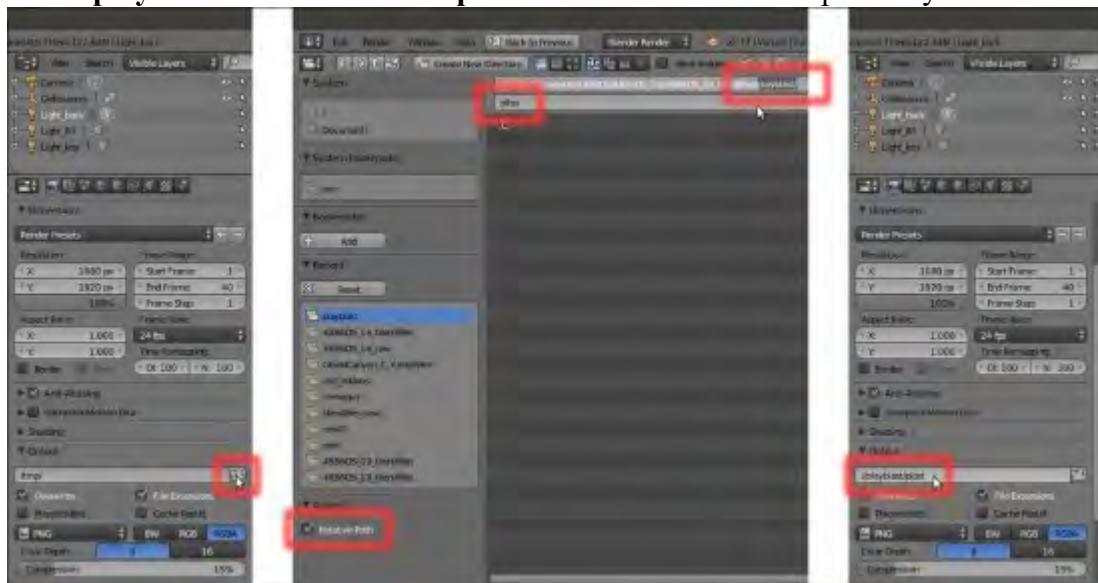
1. Put the mouse pointer inside the 3D viewport and press the numpad **0** key to go in **Camera** view; press the **Z** key to go in **Solid** viewport shading mode, then scroll the mouse wheel to zoom the **Camera** view inside the window:



The Camera view in Solidviewport shading mode

2. Go to the **Render** window and to the **Dimensions** subpanel; check for the **X** and **Y** sizes of the rendering under **Resolution**, specified in pixels, and move the *Percentage scale for render resolution* slider, usually set to **50%**, to **100%**.
3. Go down to the **Output** subpanel and click on the folder icon button to the end of the path slot; browse to the location you want to save your rendering, then type in the first line of the path to the folder you want to create at that location, followed by the slash (/) and press *Enter*.
4. A pop-up will ask you to confirm the creation of the new directory; confirm and then type a generic frame name in the second line, go to the left side vertical bar to be sure that the bottom **Relative Path** item is enabled and finally click on the **Accept** button at the top left of the screen.

I used **playblast** for the folder and **plbst** for the frame name respectively.



The new directory and the rendered frames name

5. Save the file, then go to the **Camera** view toolbar and click on the last *ciak* icon button to the left to start the OpenGL playblast:



The two buttons to start the OpenGL rendering (for a still to the left, for the animation to the right)

How it works...

In our example, the OpenGL playblast rendered single .png images with an alpha background because, as you can see in the **Render** window visible in the previous screenshot, these are the settings of the **Output** subpanel. Be aware that the resolution, the format and the path where the playblast frames are saved, always depend on the settings in the **Render** window, the same settings that will be used for the final real rendering (but of course the resolution of the playblast can be easily and temporarily be made smaller with the slider of the percentage scale).

There's more...

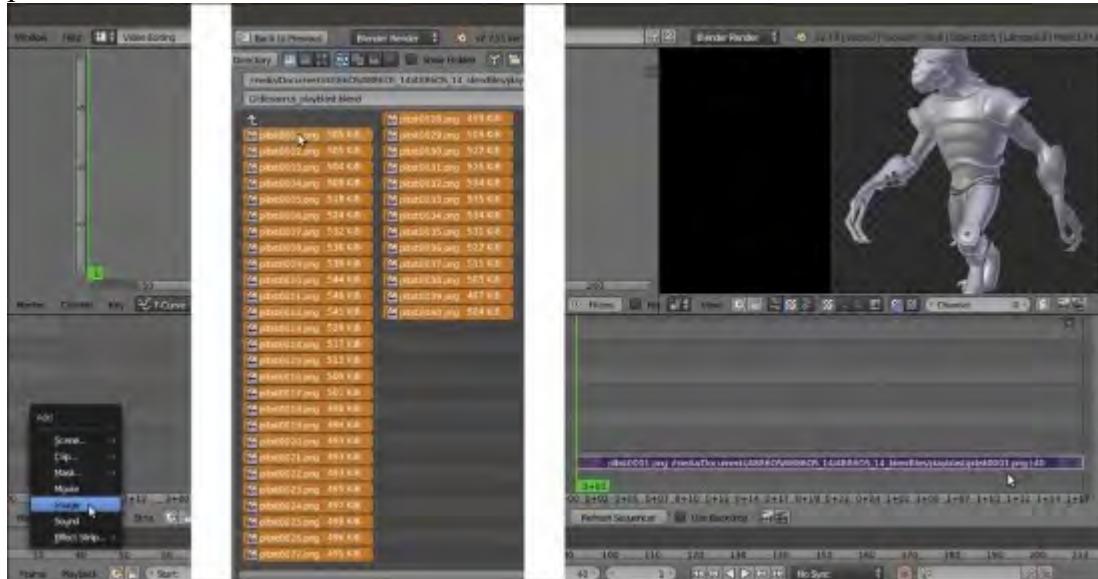
Once we have rendered all the frames, we can use an external player to see them in sequence (in **Ubuntu**, I use the free player **DJV Imaging**, <http://djh.sourceforge.net>) or, just quickly build a movie through the **Blender Sequencer**:

1. Go to the *Screen datablock* button on the top main header and click it to switch to the **Video Editing** screen:



Switching to the Video Editing screen layout

- Put the mouse pointer in the **Video Sequence Editor** window at the bottom and press *Shift + A*; from the pop-up menu select the **Image** item (*Add an image or image sequence to the sequencer*), then browse to the playblast folder location, click on it and once inside, press the *A* key to select all the contained frames, then press *Enter* to confirm. The frames are added to the **Video Sequence Editor** window as a single strip and the current frame appears in the preview window:



Loading the rendered frames in the Video Sequence Editor

3. Go back to the **Default** screen and to the **Render** window under the main **Properties** panel. In the **Output** subpanel, where you can change the path to save the movie in a different location (or also leave it as it is), click on the **File Format** button to select a **Movie** format, for example, **AVI JPEG**. Choose **BW** or **RGB** and the **Quality** compression ratio (but the default **90%** is usually OK); then go to the **Post Processing** subpanel and ensure that the **Sequencer** item is enabled:



The Output and the Post Processing subpanels inside the Render window

4. Go to the top of the **Render** window and click on the **Animation** button; remember that Blender uses two different buttons to start the rendering of a still image or of an animation, both for the final rendering and for the 3D viewport toolbar OpenGL preview we have seen in the *How to do it...* section.

The rendering starts and the **Sequencer** processes all the .png images outputted by the playblast, transforming them into a single compressed .avi movie then saved in the same directory as the frames.

The process is visible in the **UV/Image Editor** window that replaced the **Camera** view, indicated in the toolbar by the **Render Result** label on the image datablock to the left (because the **Image Editor** item is the one selected in the **Display** slot under the **Render** subpanel) and by the **Sequence** label visible in the **Layer** slot to the right:



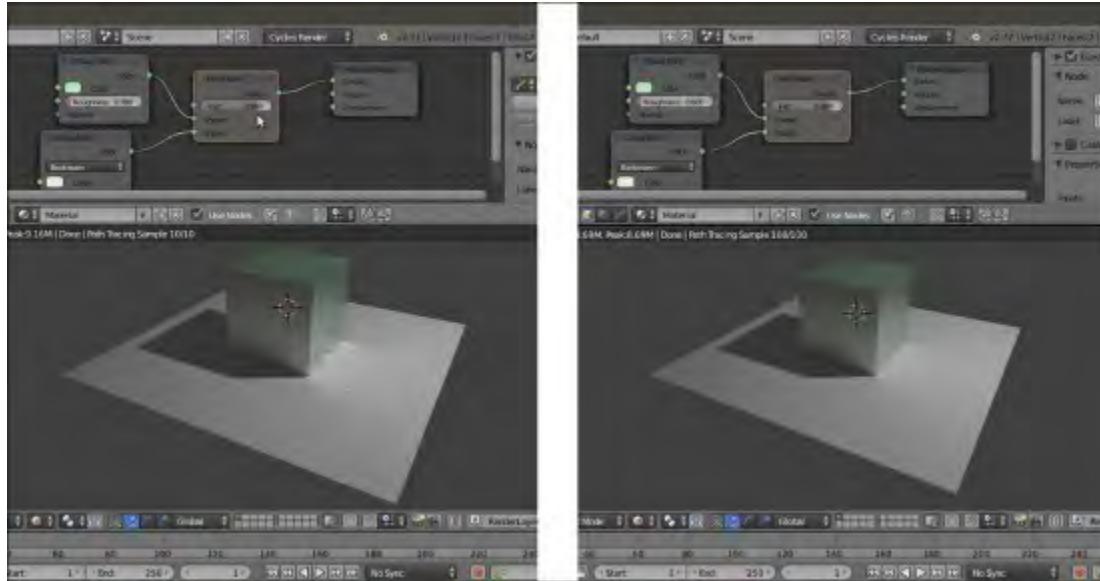
The Render Result window

See also

- <http://www.blender.org/manual/render/display.html>
- <http://www.blender.org/manual/render/output/video.html>
- <http://www.blender.org/manual/render/output.html>

Obtaining a noise-free and faster rendering in Cycles

The **Cycles Render** engine can be very slow compared to **Blender Internal**; by the way, some of the rendering settings can be tweaked to make it work faster; the goal here is to avoid fireflies and noise, usually due to low samples and a light source that is too bright.



Rendered previews of an example scene, showing a cube on a plane with and without noise and fireflies

Getting ready

Start Blender and load the `Gidiosaurus_playblast.blend` file.

1. Click on the *Scene datablock* button in the top main header to switch from **BI** to **Cycles**.
2. Go to the **Outliner** and enable the visibility of the **Light_key** lamp in the viewport by clicking on the grayed eye icon.
3. Put the mouse pointer inside the **Camera** view and press *Shift + B* to draw a box around the character's head, then zoom to it.
4. Save the file as `Gidiosaurus_render.blend` and press *Shift + Z* to start the **Rendered** preview.

How to do it...

If you have a capable graphic card supporting **GPU** (go to the last *See also* section for the link to a list of supported **GPU** graphic cards for **Cycles**), the next thing to do is:

1. Call the **User Preferences** panel (*Ctrl + Alt +U*) and go to the **System** tab; on the bottom left there is the **Compute Device** item and the slot you can click on to select the device for the rendering: if you have a graphic card that supports this feature, set the GPU instead of the default CPU, and to make this permanent, click on the **Save User Settings** button, or press *Ctrl + U*, and close the panel.
2. Go to the **Render** window under the main **Properties** panel and, in the top **Render** subpanel it is now possible to select the **GPU** item as a rendering device, but only if your graphic card supports **CUDA**.

This will boost your rendering speed several times, making it possible to significantly increase the rendering samples in the **Sampling** subpanel to reduce or even avoid the noise and keep good rendering times. Using the **GPU**, it's also possible to increase the size of the **X** and **Y Tiles** in the **Performance** subpanel (two or three times the default size is **64**).

But, not everyone has a **GPU** graphic card yet, and there are also cases where you have to mandatorily use the **CPU** instead (for example, for very big scenes with a lot of geometry that doesn't fit inside the somewhat limited **RAM** of a graphic card).

In such cases, there are things you can do to try to obtain faster and better quality render results:

3. Select the **Light_key** lamp and in the **Node Editor** window add a **Light Falloff** node (*Shift + A | Color | Light Falloff*); connect its **Linear** output to the **Strength** input socket of the **Emission** node, set the **Strength** to **1000.000** and the **Smooth** value to **1.000**.
4. Click on the color box of the **Emission** node and change the color to **R 0.800, G 0.800, B 0.650**.
5. Go to the **Render** window and in the **Sampling** subpanel set the **Samples** to **200** or a higher value both for **Render** and **Preview**, to reduce the noise as much as possible.
6. Set the **Clamp Direct** and the **Clamp Indirect** values to **3.00** or **4.00** or even higher (they are set to **0.00** by default); when possible, it is better to leave the **Clamp Direct** item at **0.00** or use values higher than **2.00**, otherwise you could get weird effects in the texturing.
7. In the **Light Path** subpanel, disable both the **Reflective Caustics** and the **Refractive Caustics** items (unless you really need to have caustics in your render) and set the **Filter Glossy** value to **4.00 – 6.00**.
8. In some cases, it won't be possible to totally eliminate the noise or the fireflies; but, because we are going to render an animation, that is several frames in sequence, at least we can make the noise less noticeable and more *natural* looking: go back to the **Sampling** subpanel and click on the **Seed** slot to type `#frame`. This creates an automated driver that takes the seed value from the current frame number, in order to have different noise at every frame.



The Light Falloff node for the Lamps, the Seed driver for the noise, the Caustics items and the Filter Glossy value

See also

- http://www.blender.org/manual/render/cycles/reducing_noise.html
- <http://www.blender.org/manual/render/cycles/settings/index.html>
- http://www.blender.org/manual/render/cycles/gpu_rendering.html
- <http://www.blender.org/manual/render/workflows/animations.html>
- http://www.blender.org/manual/render/blender_render/performance.html
- A list of supported GPU graphic cards for Cycles can be found at <https://developer.nvidia.com/cuda-gpus>

Compositing the render layers

We have seen that the rendering in the **Cycles Render** engine is quite slow but of very good quality, while the scanline **Blender Render** engine is faster but with a lower quality.

Thanks to the Blender integrated **Compositor** and to the **render layers**, it is possible to mix separated and different passes of both the renderers, obtaining a compromise between quality and speed, for example, by over-imposing the **glossy pass** obtained in **Cycles** on the **diffuse pass** obtained in **BI**, and so on.

Getting ready

To mix different passes obtained from the two render engines, we must first apply some modification to the materials of the library file:

1. Start Blender and load the `Gidiosaurus_shaders_Blender_Internal.blend` file, which is the file we used as library for the proxified character and the walkcycle action.
2. In the **Outliner** select the **Gidiosaurus_lowres** item, be sure to be in the **BI** scene and go to the **Material** window; select the first material slot, that is the **Material_skin_U0V0** slot, and in the **Node Editor** window select the **SPEC** material node:



The **SPEC** node material for **Blender Internal**

3. Go to the **Texture** window and click on the *Enable/Disable each texture* checkbox at the right side of the **env_refl_skin** texture slot to disable it:



The disabled "env_refl_skin" texture slot

4. Go back to the **Node Editor** window and select the **COL** node (this seems to be important, because of a bug in the **Blender Internal** working method with linked material nodes and the **desgraph** that doesn't update the materials and, therefore, doesn't render the diffuse color correctly).
5. Select the second **Material_skin_U1V0** slot, go to the **Node Editor**, select the **SPEC** material node, disable the **env_refl_skin** texture slot, select the **COL** material node, then go to the third **Material_skin_U2V0** slot, and so on: repeat for all the materials of the **Gidiosaurus** and of the **Armor** objects (for the **Armor** disable the **env_refl_armor** slots on both the **SPEC1** and **SPEC2** material nodes; **Eyes** and **Corneas** have the real ray-tracing mirror enabled and don't need any textures disabled).

Remember to leave the **COL** material nodes as the selected ones in the **Node Editor** window, or the diffuse color won't show in the rendering.

6. Go to the **Object** window and, in the **Relations** subpanel, assign a different **Pass Index** to the objects: assign **1** to the **Gidiosaurus_lowres** object, **2** to the **Armor** object, **3** to the **Eyes**, and **4** to the **Corneas**.

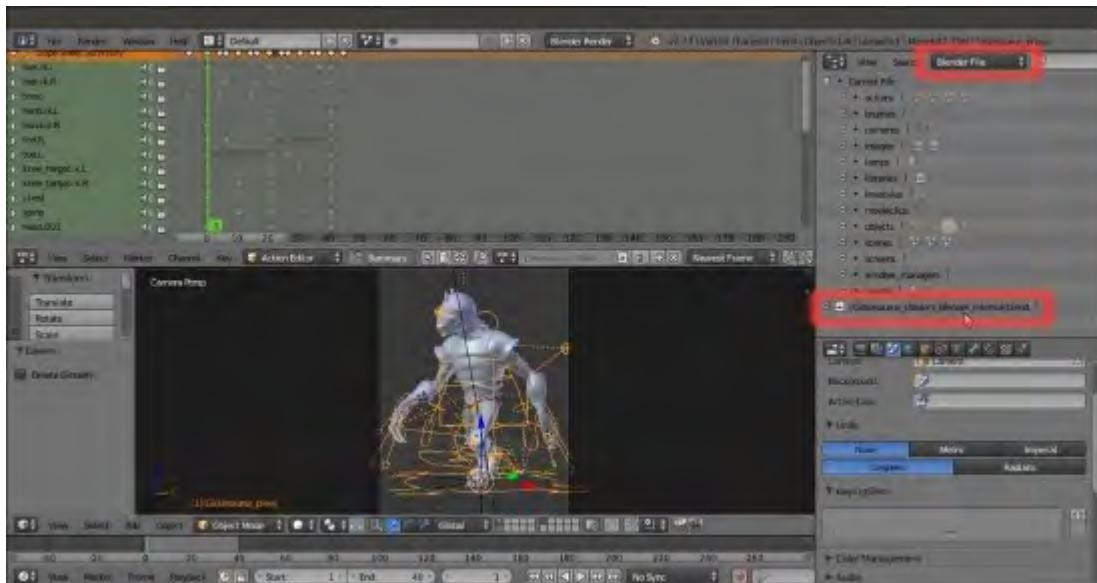


The Pass Index slot

- Save the file as `Gidiosaurus_shaders_library.blend`.

Now, because up to this point we have used the `Gidiosaurus_shaders_Blender_Internal.blend` file as library source, we must now substitute the file to be rendered the path to the new library source:

- Open the `Gidiosaurus_render.blend` file and go to the **Outliner** window: click on the *Type of information to display* button at the top to switch from **Visible Layers** to **Blender File**; expand the panel to find the `//Gidiosaurus_shaders_Blender_Internal.blend` item at the bottom:



The library path in the Outliner

- Double left-click on the item and rename it as **//Gidiosaurus_shaders_library.blend**, then press *Enter* to confirm and save the file:



The modified library path

- Press *Ctrl + O* | *Enter* to re-load the file: now all the assets should be linked from the new library file.
- Save the file as **Gidiosaurus_compositing.blend**.

How to do it...

At this point we must prepare the passes for the two scenes that will be used later, as elements to be mixed through a third new *compositing* scene:

- Click on the + icon button to the right side of the *Scene datablock* button in the main top header and, in the **New Scene** pop-up menu, select the **New** item: this creates a new **empty** scene; rename it **comp**.
- Click on the *Screen datablock* button to the left and switch to the **Compositing** screen, then click again on the *Scene datablock* button and re-select the newly created **comp** scene.
- Click again on the *Screen datablock* button and go back to the **Default** screen; go to the *Scene datablock* button and select either the **BI** or the **Cycles** scene.
- From now on, it's enough to select the **Compositing** layout in the *Screen datablock* button to switch automatically to the **comp** scene, and the **Default** layout to go to the **BI** or the **Cycles** scene (depending on the last one selected).
- Go to the **Default** screen and, if not already loaded, load the **BI** scene; in the main **Properties** panel, go to the **Render Layers** window (the second icon button from the left in the *Type of active data to display and edit* windows row).
- Double click on the **RenderLayer** name in the first slot at the top of the subpanel to rename it as **BI**. Go down to the **Passes** subpanel, disable the **Z** pass item and enable **Object Index**; then

go to the second column and enable the **Shadow** pass but then also click on the *Exclude shadow pass from combined* button to its extreme right side: do the same also for the **Emit**, the **AO** and the **Indirect** passes.



The Passes setting for the Blender Render scene

7. Click again on the *Scene datablock* button in the top main header and switch to the **Cycles** scene.
8. Double click on the **RenderLayer** name in the first slot at the top of the subpanel to rename it as **Cycles**, then disable the **Combined** and the **Z** passes, leave the already enabled **Shadow** pass as it is and enable also the **Glossy Direct**, **Indirect** and **Color** passes:



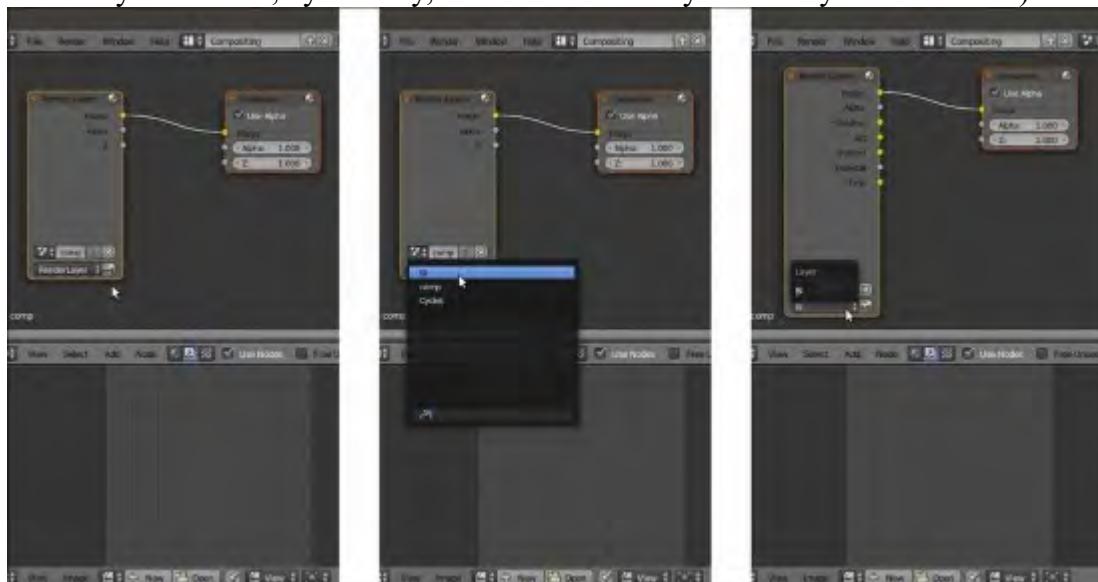
The Passes setting for the Cycles Render scene

- Now click on the *Screen datablock* button and switch to the **Compositing** screen.
- In the **Node Editor** window toolbar, go to the *Node tree type to display and edit* row, click on the *Compositing nodes* button (the middle one) and then enable the **Use Nodes** checkbox:



The Compositing nodes button

- Two compositing nodes are automatically added in the **Node Editor** window: a **RenderLayers** node connected to a **Composite** node.
- Click on the double arrows to the side of the *Scene datablock* on the **RenderLayers** node to switch from the **comp** scene to the **BI** scene, and, if necessary, in the bottom **Layer** button, select the name of the respective render layer (that we labeled as **BI** again; it shouldn't be necessary to select it, by the way, because it's the only render layer in the scene).



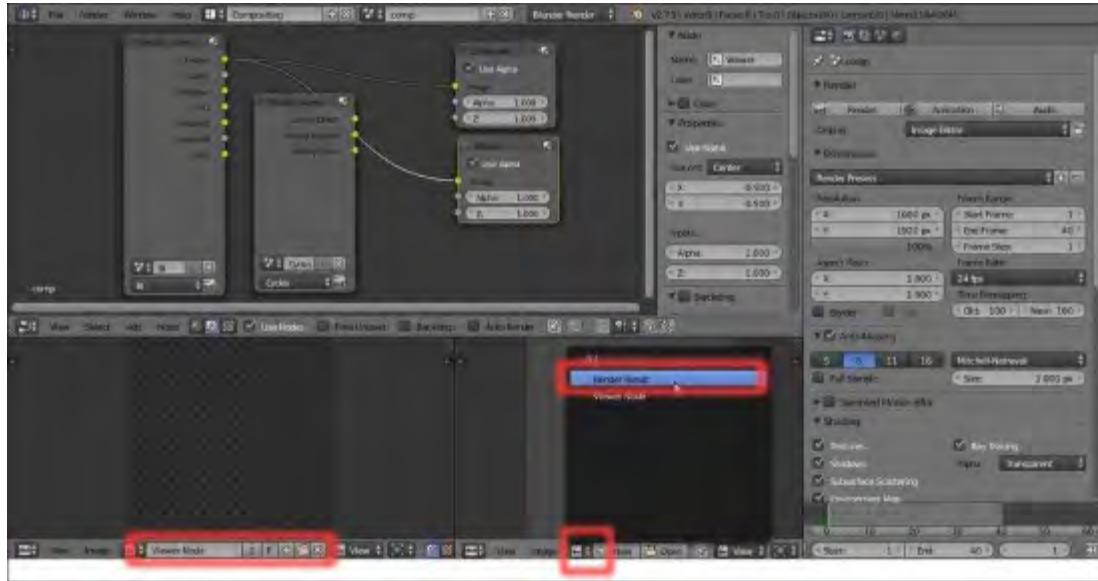
The render layer selector in the RenderLayers node and the output sockets to the Composite node

12. Press **Shift + D** to duplicate the **RenderLayers** node and repeat the procedure in the duplicated one, this time selecting the **Cycles** scene datablock and render layer:



The duplicated RenderLayers node

13. Put the mouse pointer inside the **Node Editor** window and press **Shift + A** to add a **Viewer** node (**Shift + A | Output | Viewer**); connect the **Image** output of the **BI RenderLayers** node also to the **Image** input socket of the **Viewer** node.
14. Move down and switch the **3D View** window with another **UV/Image Editor** window.
15. In the left image editor window, click on the double arrows to the left of the image datablock (*Browse Image to be linked*) and from the pop-up menu select the **Viewer Node** item.
16. In the right image editor window, instead, select the **Render Result** item.

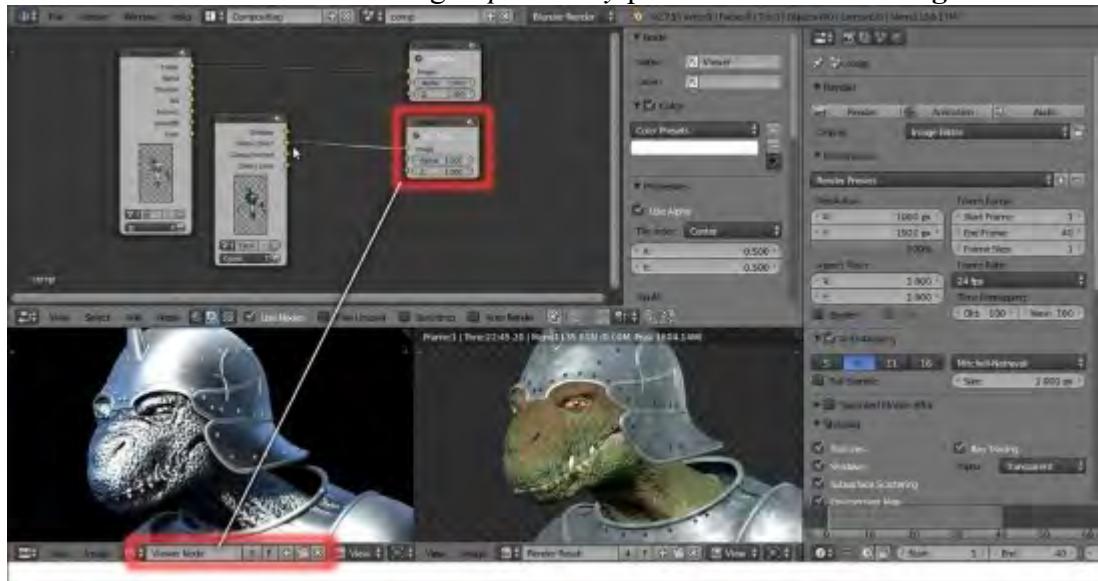


The Viewer node and the Viewer Node and Render Result windows

- At this point, press **F12** or click on the **Render** button inside the main **Properties** panel to start the rendering.

When the rendering is done, only the **BI** render is visible in the two bottom editor windows, because at the moment it's the only **RenderLayers** node connected to the **Viewer** and to the **Composite** nodes.

- Connect the **Glossy Direct** output of the **Cycles RenderLayers** node to the **Image** input socket of the **Viewer** node to see the single *specularity* pass in the left UV/Image Editor.



The Glossy Direct pass visualized in the Viewer Node window

In fact, we could also add more than one **Viewer** node to the **Node Editor** window and use them to visualize the different passes of **RenderLayers**, by connecting each pass output to each **Viewer** node; the last selected **Viewer** node will be the one visualized in the **Viewer Node** bottom window.

19. Enable the **Auto Render** item at the extreme right side on the **Node Editor** window toolbar, add a **Mix** node (*Shift + A | Color | Mix*) and paste it between the **Image** output of the **BI RenderLayers** node and the **Image** input socket of the **Composite** node; change the **Blend Type** to **Add** and then connect the **Glossy Direct** output of the **Cycles RenderLayers** node to its second **Image** input socket. Set the **Fac** value to **0.050** and label it as **Add_GLOSSY_01**.
20. If you want, also connect the output of the **Add_GLOSSY_01** node to the **Viewer** node.



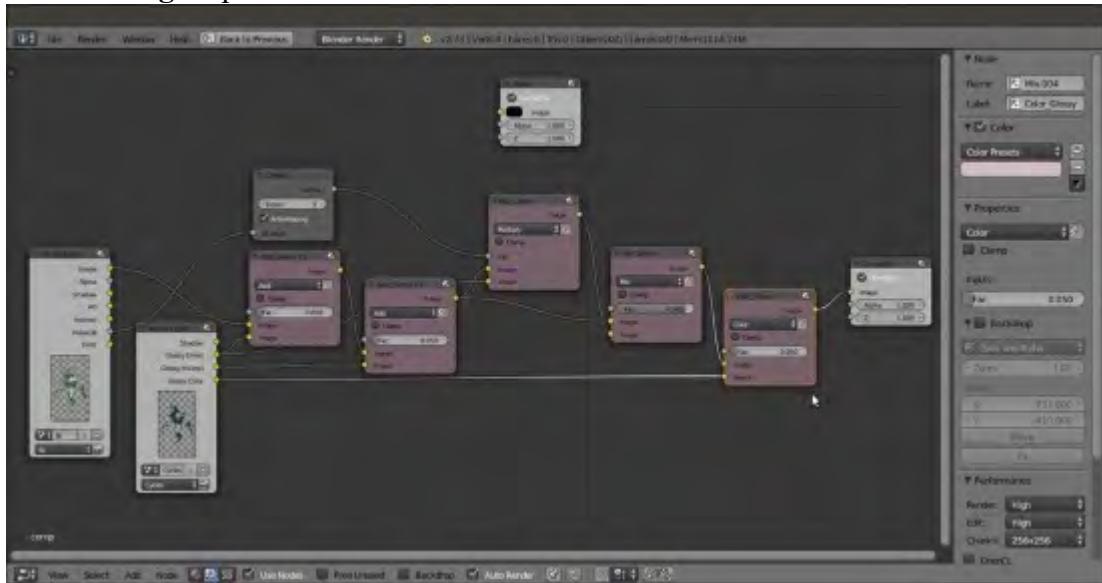
The Glossy pass, rendered in Cycles, added to the Image pass rendered in Blender Internal; note that now the armor looks too lightened

21. Press *Shift + D* to duplicate the **Add_GLOSSY_01** node, label it as **Add_GLOSSY_02** and paste it between the **Add_GLOSSY_01** and the **Composite** nodes; connect the **Glossy_Indirect** output of the **Cycles RenderLayers** node to the second **Image** input socket of the **Add_GLOSSY_02** node.
22. Press *Shift + D* to duplicate the **Add_GLOSSY_01** node again, label it as **Mul_GLOSSY**, change the **Blend Type** to **Multiply** and set the **Fac** value back to **1.000**; connect the output of the **Add_GLOSSY_02** node to its first **Image** input socket and the **Glossy_Direct** output of the **Cycles RenderLayers** node to its second **Image** input socket. Connect the output of the **Mul_GLOSSY** node to the **Composite** node.
23. Add an **ID Mask** node (*Shift + A | Converter | ID Mask*), set the **Index** value to **2** and connect the **IndexOB** output of the **BI RenderLayers** node to the **ID value** input socket, then connect the **Alpha** output to the **Fac** input socket of the **Mul_GLOSSY** node; enable the **Anti-Aliasing** item.
24. For the moment, disconnect the **Viewer** node.



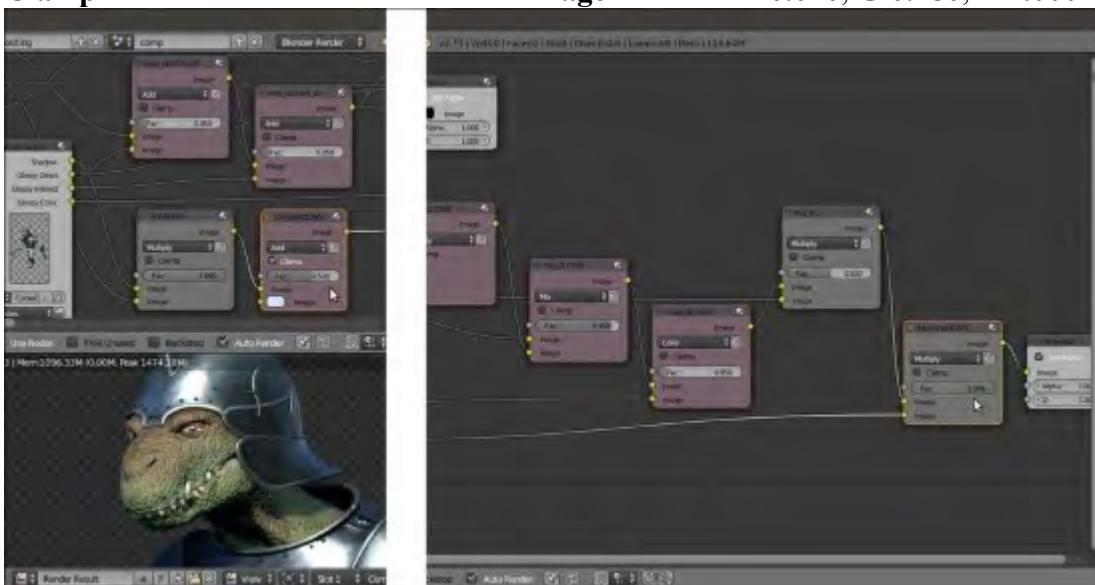
The IndexOB output used as a mask for the addition of the Glossy pass only on the character's skin

25. Press **Shift + D** to duplicate a new **Mix** node, label it as **Mix_GLOSSY** and set the **Blend Type** to **Mix** and the **Fac** value to **0.900**; connect the output of the **Add_GLOSSY_02** node to the first **Image** input socket and the output of the **Mul_GLOSSY** node to the second **Image** input socket.
26. Press **Shift + D** to duplicate another **Mix** node, label it as **Color_GLOSSY** and set the **Blend Type** to **Color** and the **Fac** to **0.050**; connect the output of the **Mix_GLOSSY** node to the first **Image** input socket and the **Glossy_color** output of the **Cycles RenderLayers** node to the second **Image** input socket:



Adding more compositing nodes to re-build the separately rendered Glossy passes

27. Add a new **Mix** node (*Shift + A | Color | Mix*) right after the **Color_GLOSSY** one, label it as **Mul_AO**, set the **Blend Type** to **Multiply** and connect the **AO** output of the **BI RenderLayers** node to the second **Image** input socket; set the **Fac** to **0.500**.
28. Add a new **Mix** node (*Shift + A | Color | Mix*), label it as **SHADOWS**, change the **Blend Type** to **Multiply** and set the **Fac** value to **1.000**; connect the **Shadow** output of the **BI RenderLayers** node to its first **Image** input socket and the **Shadow** output of the **Cycles RenderLayers** node to the second **Image** input socket.
29. Press *Shift + D* to duplicate the **Mul_AO** node and label it as **Mul_SHADOWS**; paste it right behind the **Mul_AO** node and set the **Fac** value slider to **1.000**.
30. Connect the output of the **SHADOWS** node to the second **Image** input socket of the **Mul_SHADOWS** node.
31. Add a new **Mix** node (*Shift + A | Color | Mix*), label it as **Color_SHADOWS** and paste it right behind the **SHADOWS** node; set the **Blend Type** to **Add**, the **Fac** to **0.500** and enable the **Clamp** item: set the color of the second **Image** socket to **R 0.640, G 0.780, B 1.000**.



Multiplying and coloring the shadow pass

32. Add a new **Mix** node (*Shift + A | Color | Mix*) right behind the **Mul_SHADOWS** one, label it as **Add_INDIRECT**; set the **Blend Type** to **Add**, the **Fac** to **0.300** and connect the **Indirect** output of the **BI RenderLayers** to the second **Image** input socket.
33. Repeat the previous step but label the node as **Add_EMIT**, set the **Fac** value to **1.000** and connect the **Emit** output of the **BI RenderLayers** node to the second **Image** input socket.
34. Add one more **Mix** node (*Shift + A | Color | Mix*), label it as **Col_EMIT**, set the **Blend Type** to **Multiply** and paste it behind the **Add_EMIT** node; set the color of the second **Image** socket to **R 1.000, G 0.542, B 0.073**.
35. Add a new **ID Mask** node (*Shift + A | Converter | ID Mask*), connect the **IndexOB** output of the **BI RenderLayers** node to the **ID value** socket, set the **Index** value to **3** and connect its **Alpha** output to the **Fac** input socket of the **Col_EMIT** node.



Coloring and adding the eyes

36. Connect the output of the **Col_EMIT** also to the **Image** input socket of the **Viewer** node:



The completed compositing network

37. Save the file.

How it works...

In the *Getting ready* section:

- From step 2 to step 5 we disabled the **env_refl_skin** and the **env_refl_armor** texture slots in all the material nodes of the **BI** shaders; in fact, because we are going to add the **Cycles** reflection on the **BI** diffuse, we don't need to fake the environment reflection on the character and on the armor surfaces anymore.
- In step 6 we assigned a different **Index Pass** number to each one of the objects; this is useful to later *separate* the objects in the **Compositor** for particular effects (in our case we only needed the armor **Index Pass** number, but it's a good habit to give index passes to all the objects for any eventuality).
- At step 7 we saved the file with a new name, and at steps 8 and 9 we changed, in the file to be rendered, the path to the new library file.

In the *How to do it...* section:

- From step 1 to step 3 we added a new empty scene, **comp**, to the blend file, linked to the **Compositing** screen layout and to be used for the compositing.
- In fact, in the **comp** scene, all the **compositing** nodes are connected together so as to recreate the best possible rendering look of the **Gidiosaurus**, using the different passes from the different **BI** and **Cycles** scenes through the **render layers**.
- From step 4 to step 7 we enabled the required passes in the **Render layers** windows of both the **BI** and **Cycles** scenes; note that basically we set an almost complete *only-diffuse* render in **Blender Internal** (besides the **Indirect**, **Ambient Occlusion** and **Emit** passes, subtracted from the total render and separately outputted), while we set the **Glossy** passes in the **Cycles** engine.
- From step 8 to step 15 we set up the **RenderLayers** nodes, the **Viewer** and the **Composite** nodes.

The **RenderLayers** node outputs the rendering of a particular scene also delivering the enabled passes for that scene, through the **Render Layer** setup.

The **Compositing** node is the mandatory final output node and must always be connected as the last step of the compositing chain.

The **Viewer** node, instead, is optional but always used anyway to visualize the different steps of the compositing itself.

- At step 16 we started the rendering; Blender starts to render the **BI** and the **Cycles** scenes using the settings setup for each scene (and not the settings of the **comp** scene, which are true only for the compositing).
- From step 18 to step 34 we mixed the different passes together. Basically, we used the same approach that we have seen in the creation of the shaders, that is, by decomposing the final result into the different components; then, we obtained the diffuse from the **Blender Internal** engine because it's quite fast for that, and the glossy component from the **Cycles** render engine for the same reasons, and then we re-mixed them. The **Subsurface Scattering** pass is actually rendered together at the diffuse component in **BI** and is delivered through the **Combined (Image)** pass.
- The **Mix_GLOSSY** node added at step 24 is to tweak the strength of the multiplied **Glossy Direct** pass; we couldn't use the **Fac** value, in this case, because it was already used by the **ID Mask** output to *isolate*, thanks to the **Object Index**, the **Armor** from the rest of the render.
- With the **Col_SHADOWS** node at step 30 we obtained two goals: first, we set the dark intensity of the shadows to **0.500** and, second, we gave them a bluish coloration.

- The emission pass for the eyes has been added to the rendering through the same technique used to multiply the glossy only on the **Armor**, that is, by an **ID Mask** node and the **Object Index**.

See also

- http://www.blender.org/manual/render/blender_render/layers.html
- http://www.blender.org/manual/render/blender_render/passes.html
- http://www.blender.org/manual/render/post_process/layers.html
- http://www.blender.org/manual/composite_nodes/index.html

Part 2. Module 2

Blender 3D Incredible

Design, model, and texture complex mechanical objects in Blender

Chapter 1. Sci-Fi Pistol - Creating the Basic Shapes

In this section, we'll cover a few introductory topics, discuss our goals for the project, and then create the basic shape of a pistol. As we do this, we'll use a number of different tools and modifiers. At the end of this section, we'll be ready to move on and detail our pistol:

- A project overview
- Creating the barrel
- Modeling a handgrip and other pieces

Welcome to Blender 3D Incredible Machines.

In this book, we'll be working through a series of Blender projects aimed at increasing your modeling, texturing, and rendering skills.

Before we jump right into it, there are a few quick things that we need to cover.

First of all, I should mention that this book isn't meant for absolute beginners.

It would be great if a single volume could take you from knowing nothing about the software to cranking out sophisticated 3D models; unfortunately, this is not realistic. Blender's an incredible piece of software, but it's also complex. The sheer number of features means that learning to use it (or at least, learning to use it well) is a long-term endeavor. The best Blender modelers in the world will tell you that they always discover something new.

So, who exactly is the target audience?

In terms of skill level, this book will be most useful to intermediate users. Among other things, an intermediate user can do the following:

- Navigate comfortably in 3D space
- Switch between different views
- Add objects to a scene
- Move, rotate, and scale objects
- Open, close, and save files
- Switch between the Object and Edit modes
- Switch between face, edge, and vertex selections
- Add modifiers
- Drink large quantities of coffee

This book is called Blender 3D Incredible Machines for a reason. We're going to focus on building sophisticated mechanical models here, which means that invariably, there are certain topics that we won't cover.

There's nothing in here about creating realistic fur or setting up materials to simulate human skin. While these are fascinating topics, they're beyond the scope of this book (and frankly, I don't feel qualified to teach you about them).

As you get started with our first project, you'll find that the instructions are detailed and specific. However, as we move further along, this will become less true. This avoids repetition and also allows us to pull back and see the big picture. We can start focusing on workflow and project management, which is critical when building complex models.

There's one last thing I'd like to mention before we jump into it. In a lot of Blender tutorials, you build models based on reference or background images. This is incredibly useful when you model a real object, but everything in this book is fictional. We want to focus on technique and workflow here and leave ourselves some flexibility on the designs. Therefore, we're going to build everything in a freehand manner. However, if you'd prefer to work from a reference image, blueprints for all the models are provided at the back of this book.

This pretty much wraps up our introductory topics...so, if you're ready, let's get to work!

A project overview

For our first project, we'll be modeling a high poly, sci-fi weapon. It's approximately the same shape as a modern day handgun, but it gives us the flexibility to be creative and explore different modeling techniques.

Here's an example of what the final model will look like (without materials and textures):



In this first chapter, we'll create the basic beveled shapes of our gun. We'll start by extruding some basic shapes. Next, we'll add the main body of the gun, then move on to the additional pieces. As we do this, we'll look at a number of modifiers and mesh tools. Specifically, we'll be discussing the following topics:

- Extrusion
- Face Normals
- Tris, Quads, and N-Gons
- Smooth versus flat shading
- The Mirror Modifier
- The Edge Split Modifier
- The knife tool
- The bevel tool

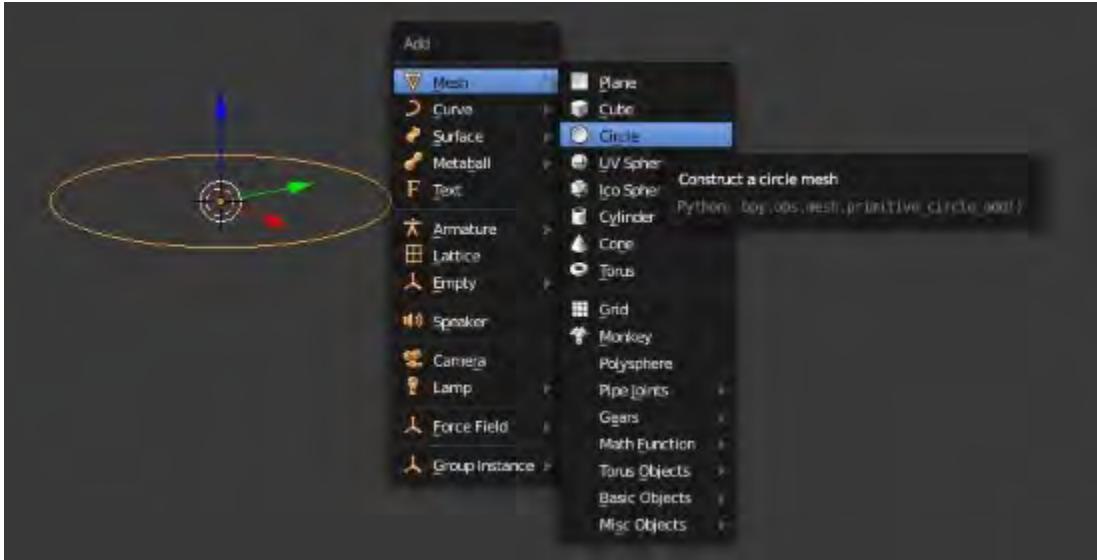
When we're finished with this section, our project will look like this:



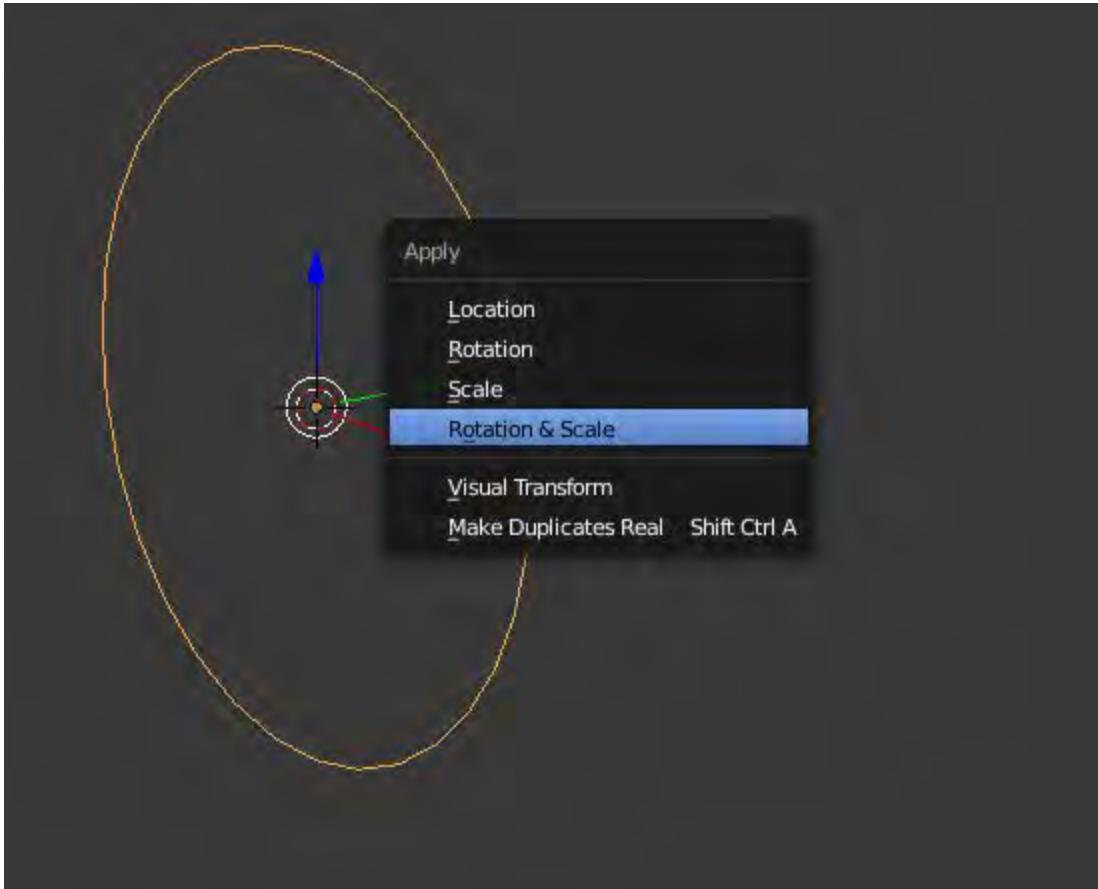
If you're ready, let's open a new scene in Blender and get started!

Creating the barrel

We'll start by adding a circle to our scene (press Shift + A to access the **Add** menu). By default, circles are added with 32 sides (and 32 vertices). This screenshot will work just fine for us:

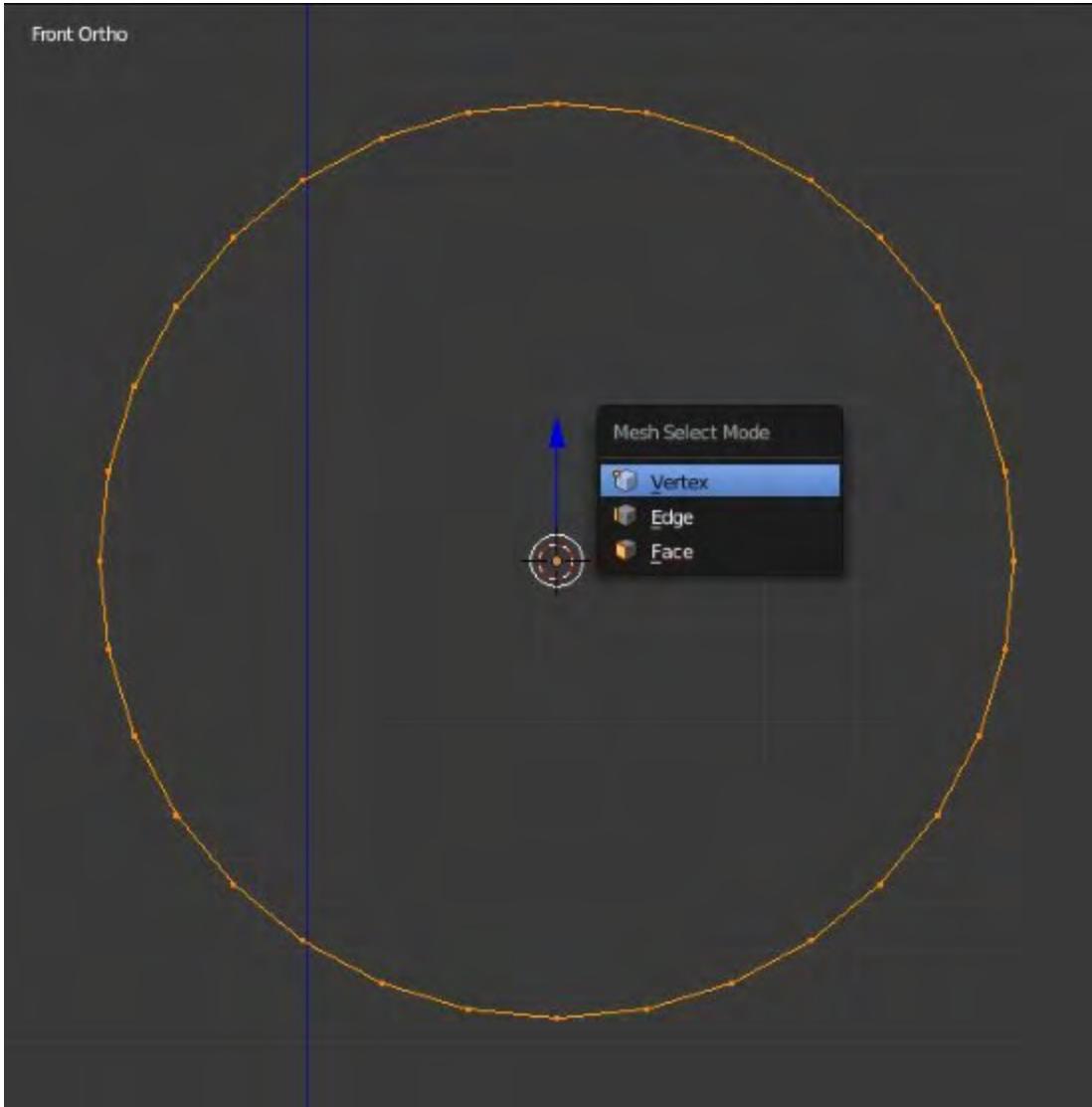


Next, we'll rotate the circle by 90° on the X axis (**R, 90, X**, and Enter). Then, we'll apply our rotation and scale by pressing **Ctrl + A** and selecting **Rotation & Scale**. We want to apply the rotation here for Blender to be aware of the object's default orientation. This is relevant to several tools and modifiers that we'll be using. We need to apply the scale for the same reason. For instance, the Bevel tool often doesn't work properly when objects do not have their scale values applied. We'll discuss this in detail in the later sections. For now, go ahead and apply your rotation and scale to the circle as shown here:

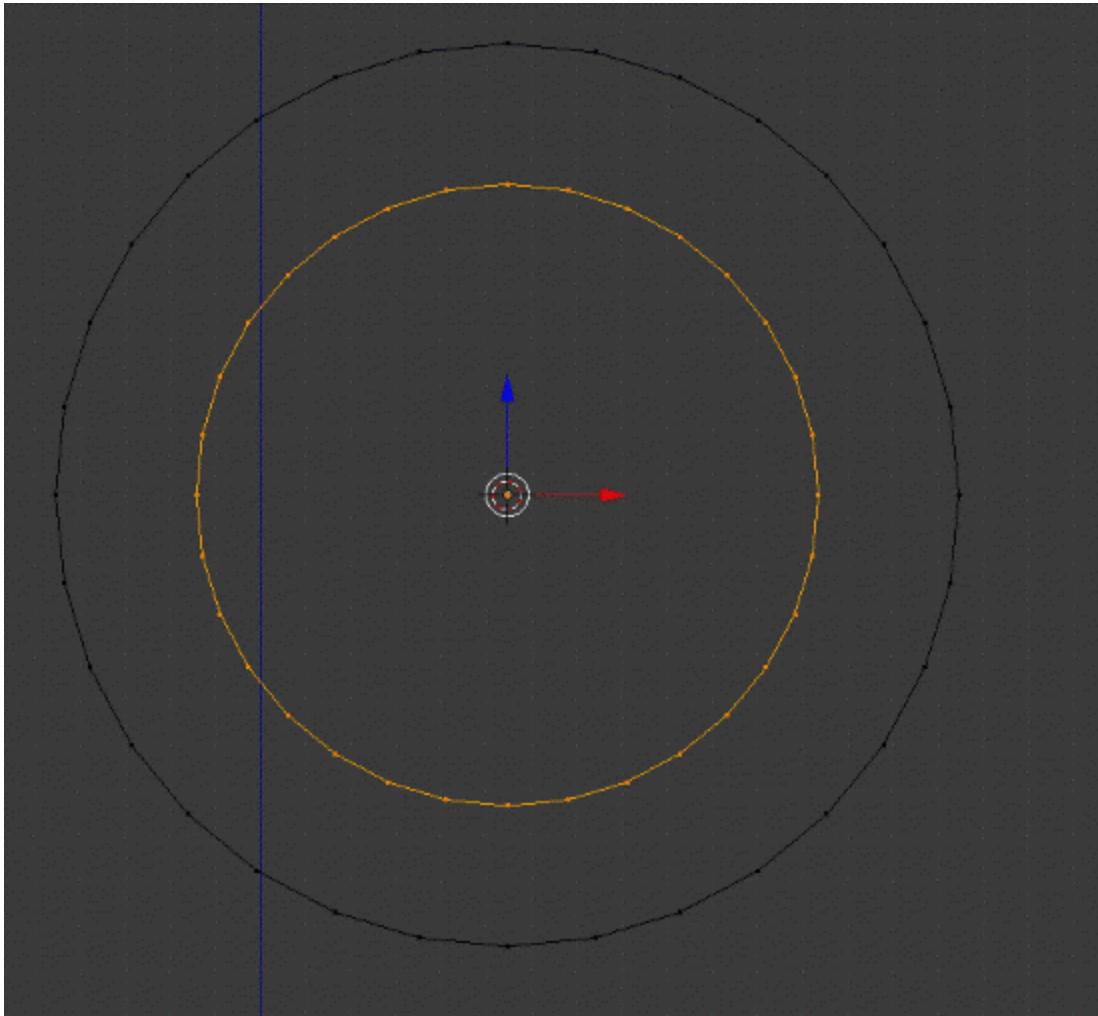


<pagebreak></pagebreak>

Now, we'll switch to the **Front** view with 1 on the numpad. We'll then go into **Edit mode** by hitting the Tab key. Switch to **Vertex** selection mode by using Ctrl + Tab:



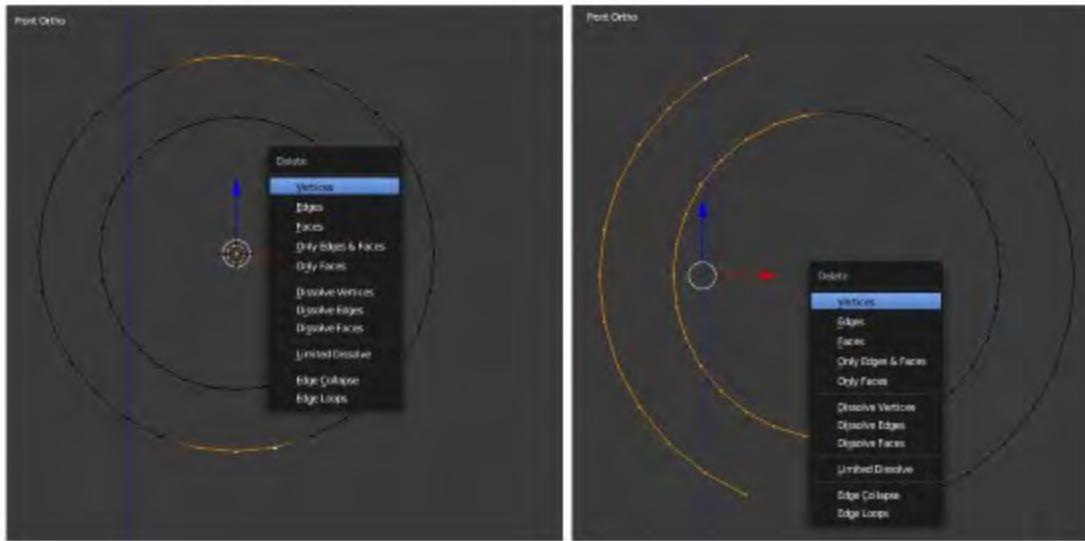
Next, we'll use Shift + D to duplicate them. We'll scale the new circle down using the S key and drag the cursor to the middle of the circle. This smaller circle will form the inside of the barrel and allow us to make changes to the outer circle without losing track of the center (and vertices) of the original one.



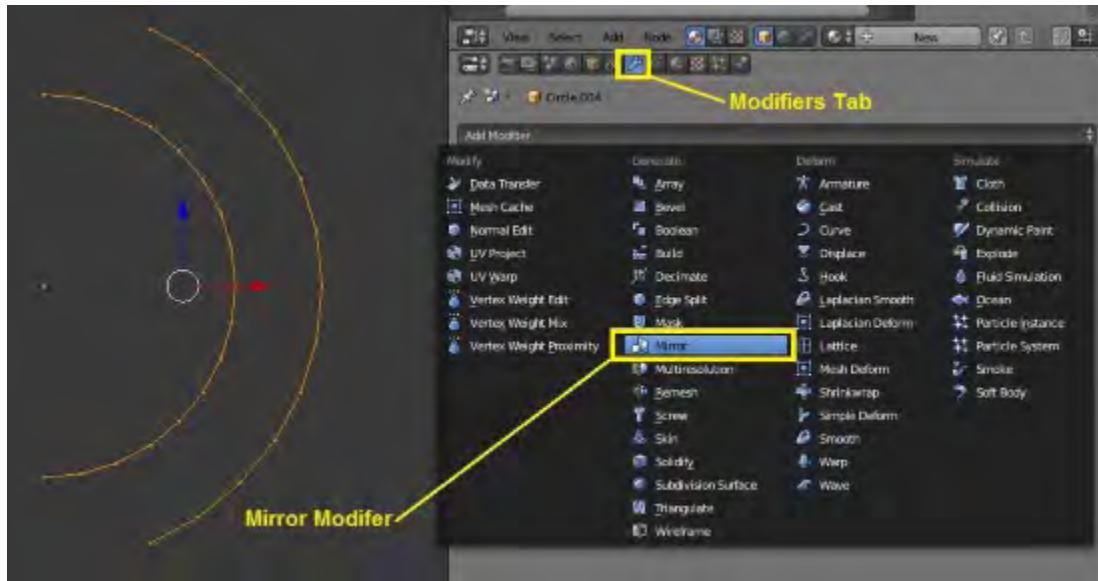
Next, we'll start deleting some vertices that we don't want. The outside of the barrel will not be a perfect circle, so we'll need to change it. First, we'll delete three vertices at both the top and bottom of the circle. To do this, we'll select the ones that we don't want any longer, and then hit X. From the pop-up menu, select **Vertices**. This will give you a good idea of how much of the original circle is remaining. You can select more or fewer vertices depending on the desired look.

<pagebreak></pagebreak>

Next, we'll delete half of both the circles:



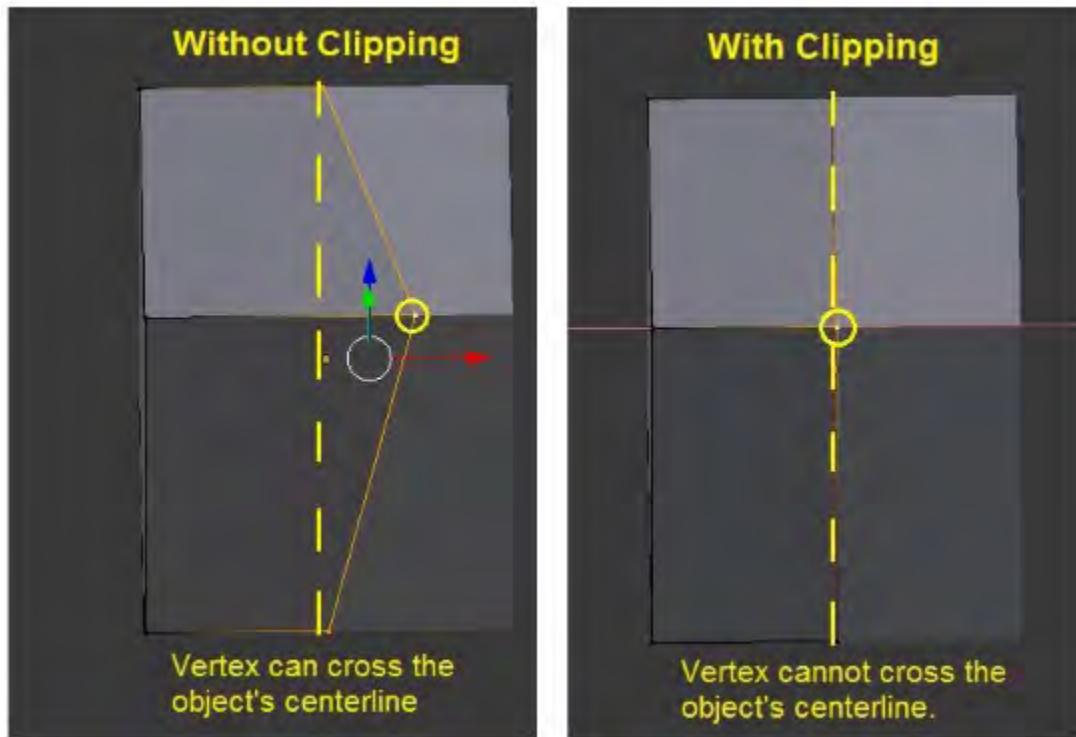
This will allow us to add the Mirror modifier. This modifier will duplicate half of an object across its axis. Using it, we can model one side of the gun, and Blender will automatically fill in the other side to match it. Obviously, this will save us a lot of time. In order to do this, go to the **Modifiers** tab of your properties panel and select **Mirror**.



<pagebreak></pagebreak>

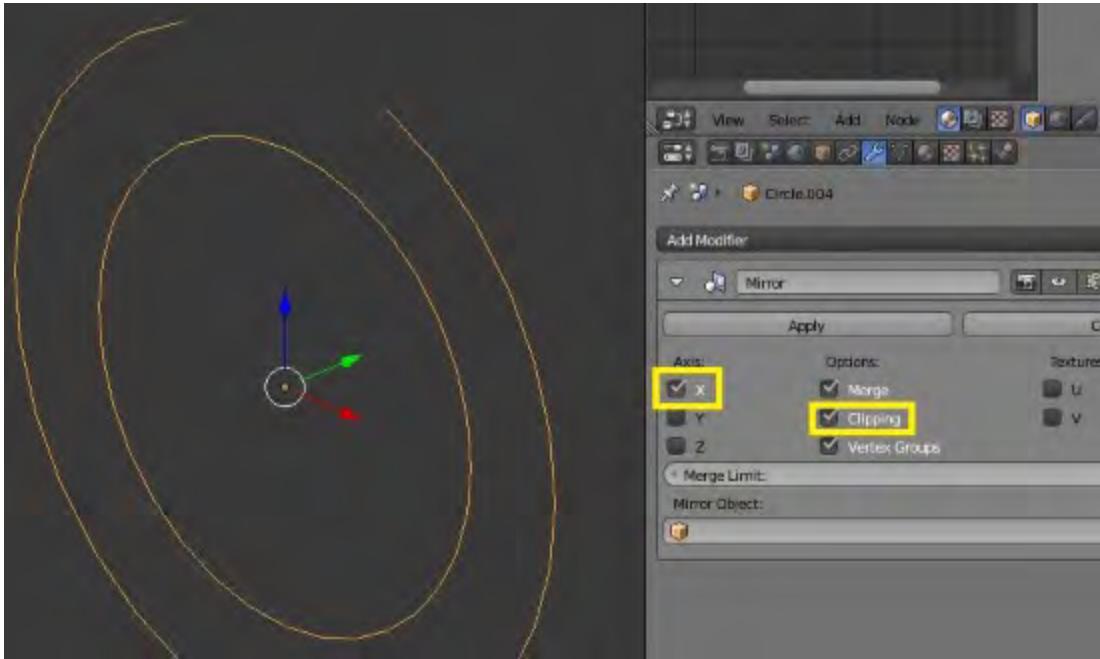
At this point, it's important to select the proper axis. A few steps back, we applied rotation and scale to the circle. This is an example of why this is important—we want the object coordinates (in Blender's 3D space) to match the axis that's listed on the mirror modifier. In this particular case, it may not have

mattered (since we're mirroring across the same axis that we rotated on). In general, however, this is a good habit to get into. So, let's go ahead and select the X axis on the mirror modifier. At this point, we'll also want to enable the Clipping feature. This will ensure that our vertices do not cross the centerline of an object:



<pagebreak></pagebreak>

So, let's go ahead and select these options:

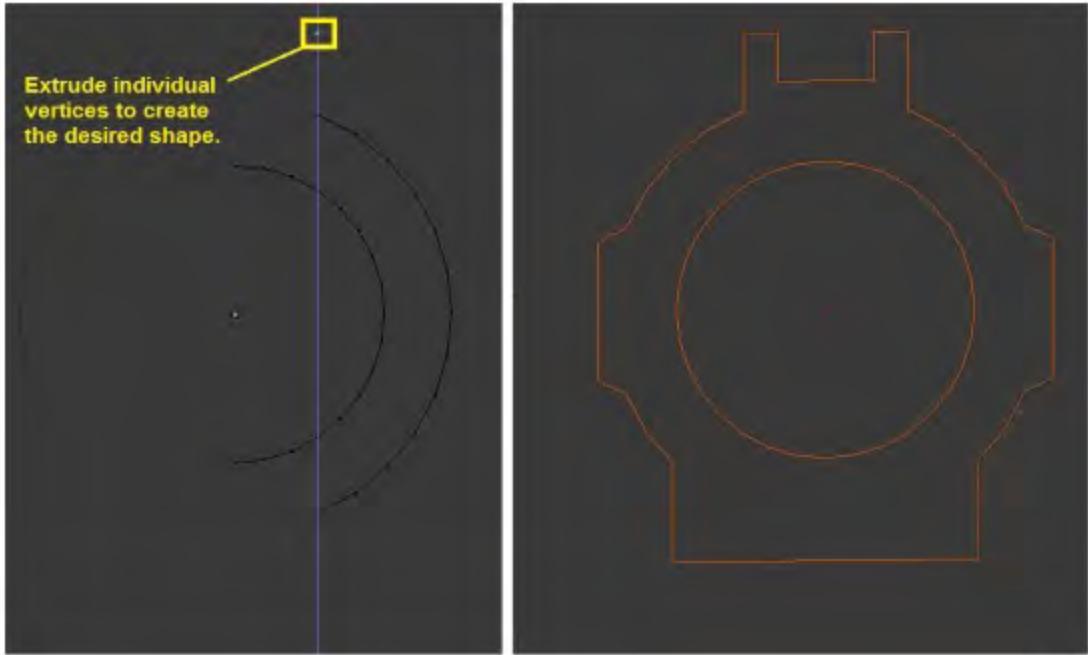


Now that we've done this, we'll be able to work on just one side of the object. So, let's start making changes to the mesh. I'm going to grab the innermost vertex at the top of our circle and extrude it upon the Z axis by pressing E and Z and hitting Enter when we've finished extruding.

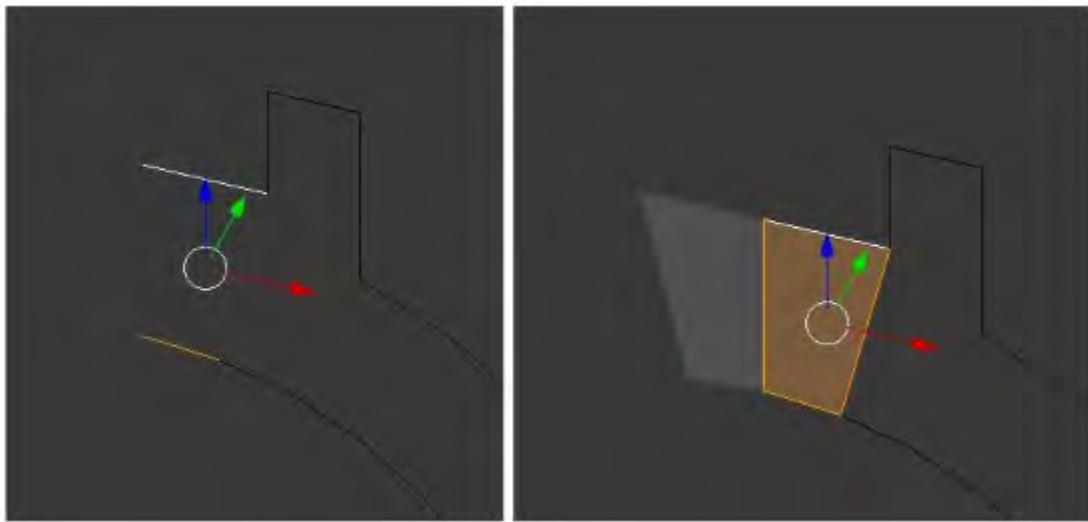
If you don't specify an axis, you'll be able to extrude (and move) this vertex in all three dimensions, which we definitely don't want to do here. So, we'll just continue to extrude vertices and move them until we have a rough shape for the cross-section of our barrel.

<pagebreak></pagebreak>

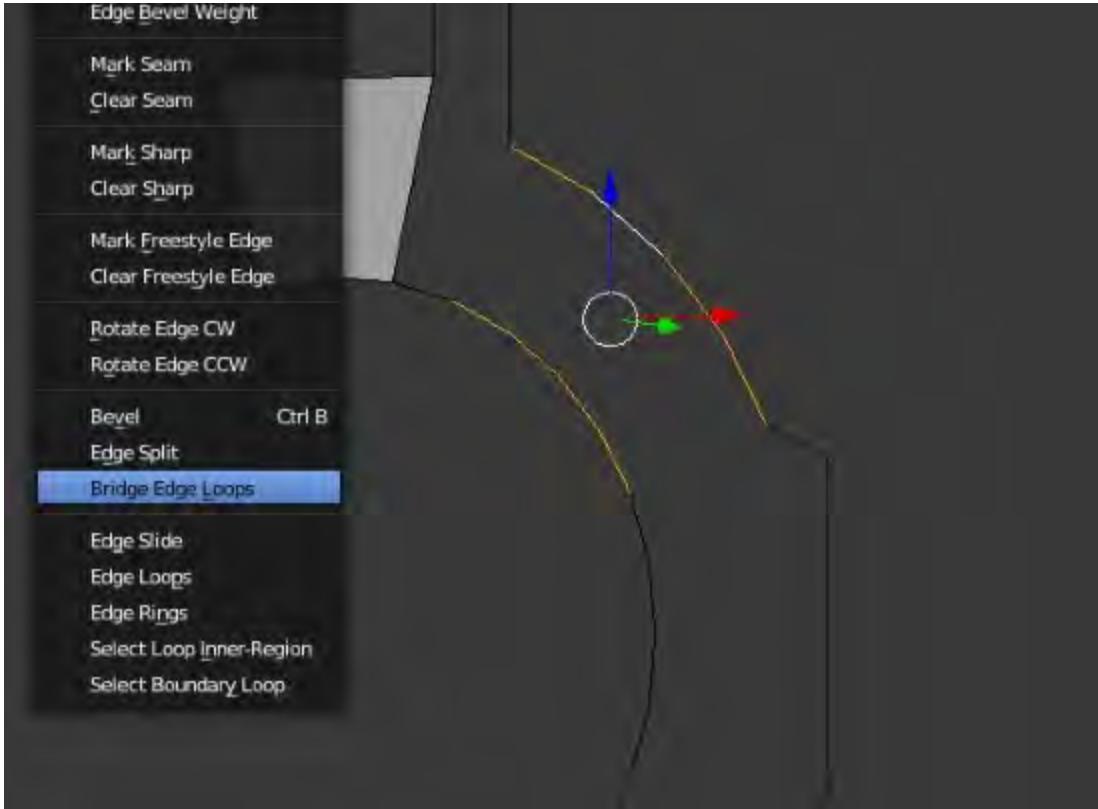
Obviously, there's a lot of room to personalize the model at this point—you don't need to follow these pictures exactly:



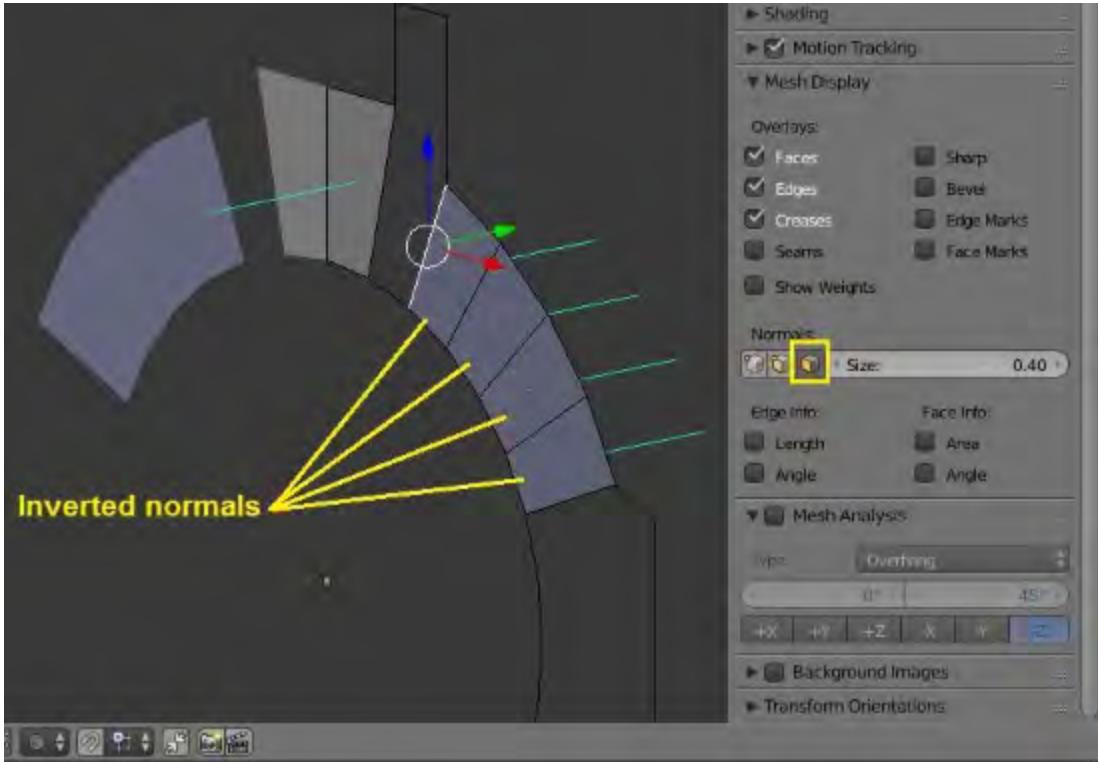
Once you have the cross-section that you're looking for, it's time to fill in some faces. It's not strictly necessary to do this right now (we'll actually end up deleting these), but it will enable us to see how Blender handles pre-existing faces during mesh editing. So, let's pick two edges at the top of our circle and create a face with the F key.



This is one way to create faces (individually). We can create a series of faces from two sets of edges. In this next example, we'll select two sets of four edges and automatically fill them in. To do this, use C **trl** + and **Bridge Edge Loops**.



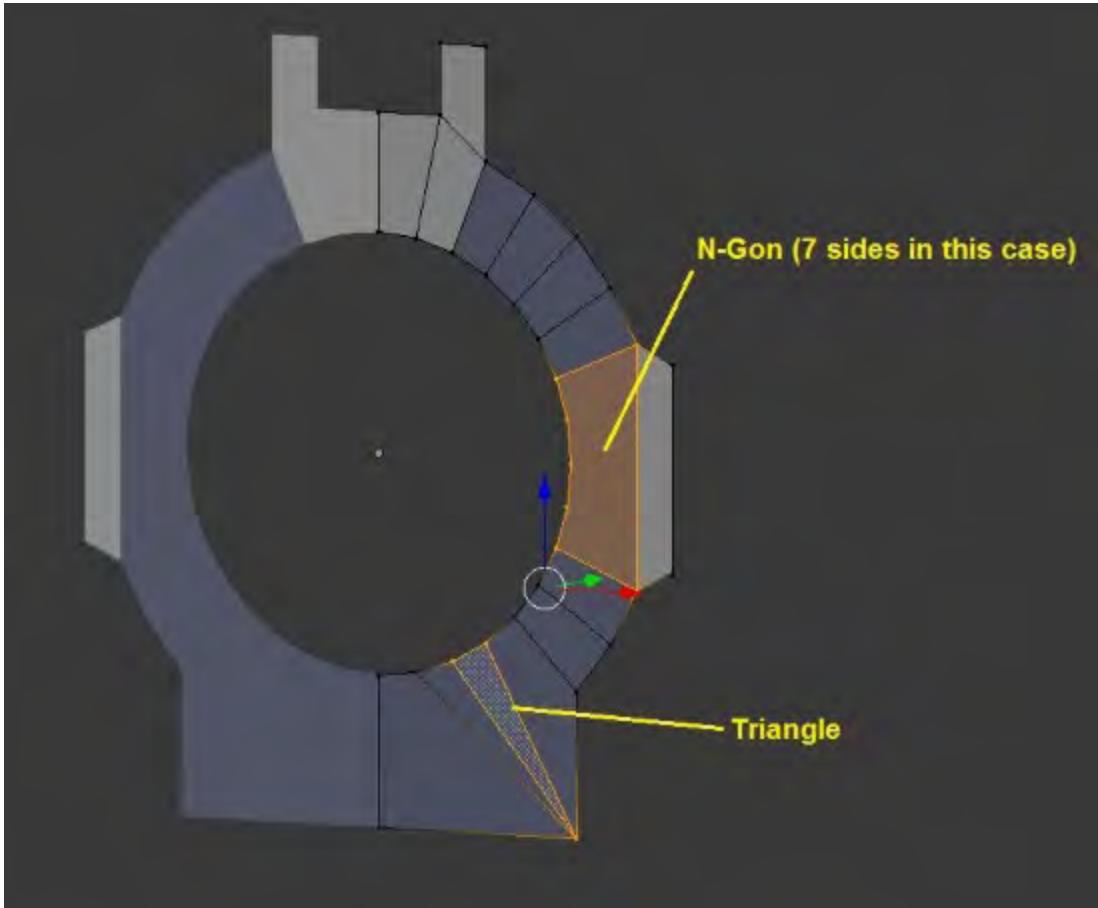
As soon as we do this, there's a good chance that you'll have a problem. Some of the faces will appear darker than others, which indicates that the **normals** are inverted. In Blender (and other 3D applications), a normal is the vector perpendicular to your face or the direction that your face is "pointing" at. When normals are inverted, they can cause all types of problems with your model. Lines that should be smooth may end up looking sharp, and materials and textures may not work properly. In older versions of Blender, there were no visual indications (by default), and your faces were inverted. However, in newer versions, faces appear darker when you look at the "bottom" or "inside" area, and they appear lighter when you look at the "top" or "outside" area. You can also pull up your shelf with the N key, and select **Normals** under **Mesh Display**. This will produce a series of lines, which show you the direction your faces are "pointing" at. In this example, the series of faces that I just created point in the wrong direction:



<pagebreak></pagebreak>

Blender does have a tool for automatically correcting normals. When you select all the faces with the A key and press Ctrl + N, Blender attempts to calculate the proper direction of the faces and flips the incorrect ones. Unfortunately, this doesn't work very well with 2D shapes. Since we just have a 2D cross section of our gun barrel right now, we'll leave this step for later.

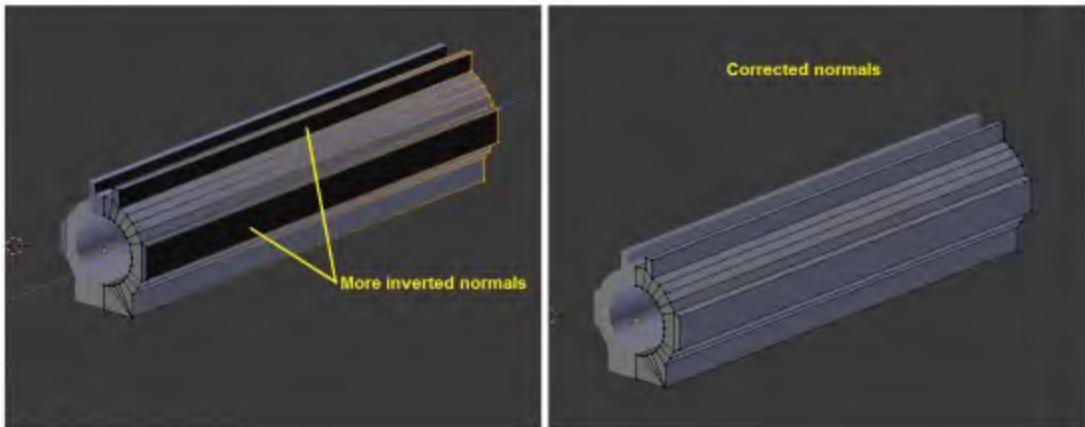
For now, let's go through and fill in the rest of our faces. All of the faces that we've created this far are known as **quads**, meaning that they're four-sided faces. However, you can also have **tris** (or triangles) and N-Gons, which are faces with more than four edges. There are advantages and disadvantages to using N-Gons, and we'll look at this later on in the modeling process. For now, you can just create an N-Gon by selecting a ring of edges and hitting the F key.



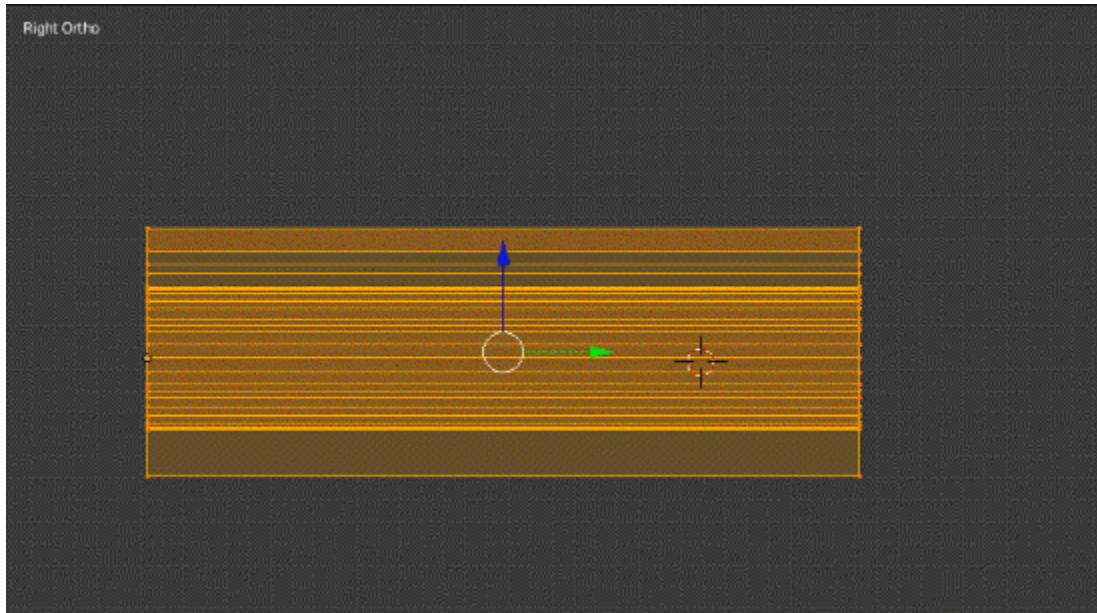
Once all your faces are filled in, you can select all of them with A key and extrude them with the E key. By default, the faces will extrude along the Y axis, which is exactly where we want them to go. In some cases, however, the faces will extrude along an axis that you don't want them to. To fix this, you can specify the axis (as we did previously with the vertices). In this case, press E and Y to extrude the faces back on the Y axis. At this point, you can drag the faces back as far as you'd like to create the gun barrel.

<pagebreak></pagebreak>

As you do so, you'll see more faces with incorrect normals. At this point, however, we have a real 3D object, so Blender will be able to recalculate our normals properly. Select the entire mesh again and hit Ctrl + N. All of your faces will instantly point in the correct direction.



Now, let's move to our side view (numpad 3).

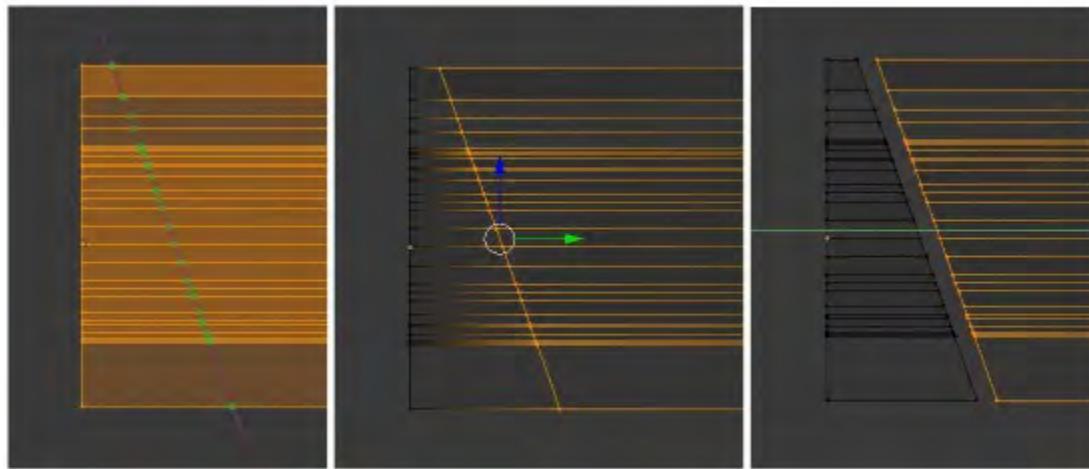


<pagebreak></pagebreak>

The front of our barrel will be angled, so we'll use the knife tool to cut across our existing mesh to create this slant. To activate the knife tool, press Shift + K (or K then Z to cut through the entire mesh). Move the knife at the top-left corner of the barrel, and anchor it with the left mouse button. Then, drag the knife down across your faces until you have the angle that you want. Press the left mouse button again and hit Enter. At this point, you have cut a new set of edges into your mesh.

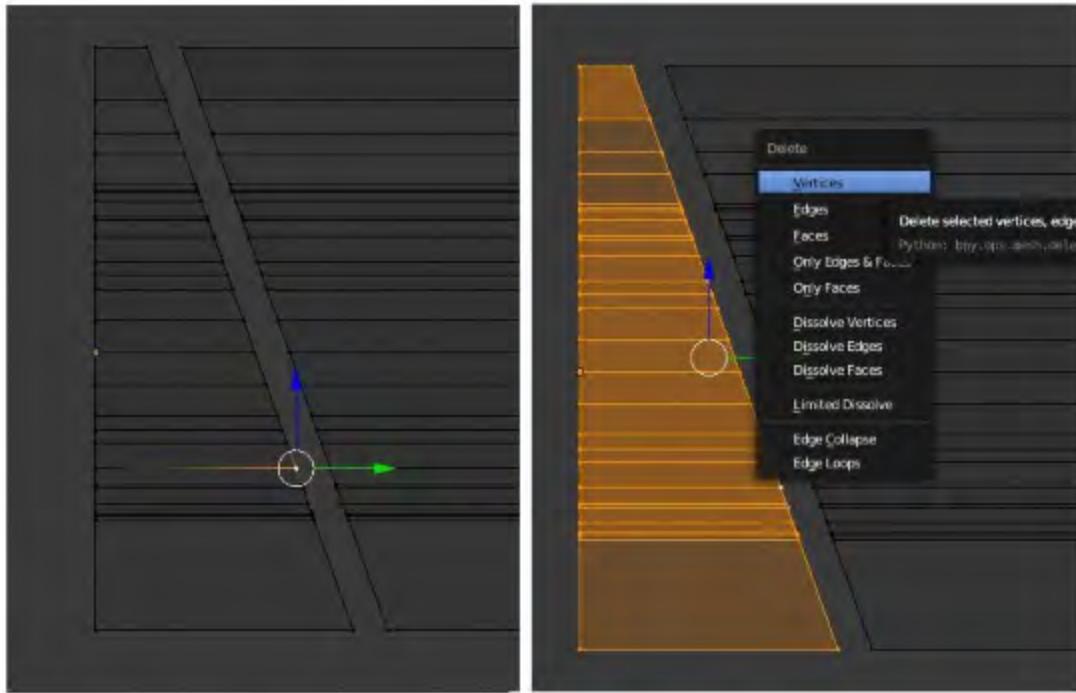
However, we're not done yet. With our new edges still selected, press the V key to rip them apart. This creates duplicate vertices that sit at the top of each other, "ripping" the mesh apart. You can move one set

of vertices away from the other on the Y axis if you'd like; it makes things easier to see. You can also hit Esc immediately after ripping it apart.



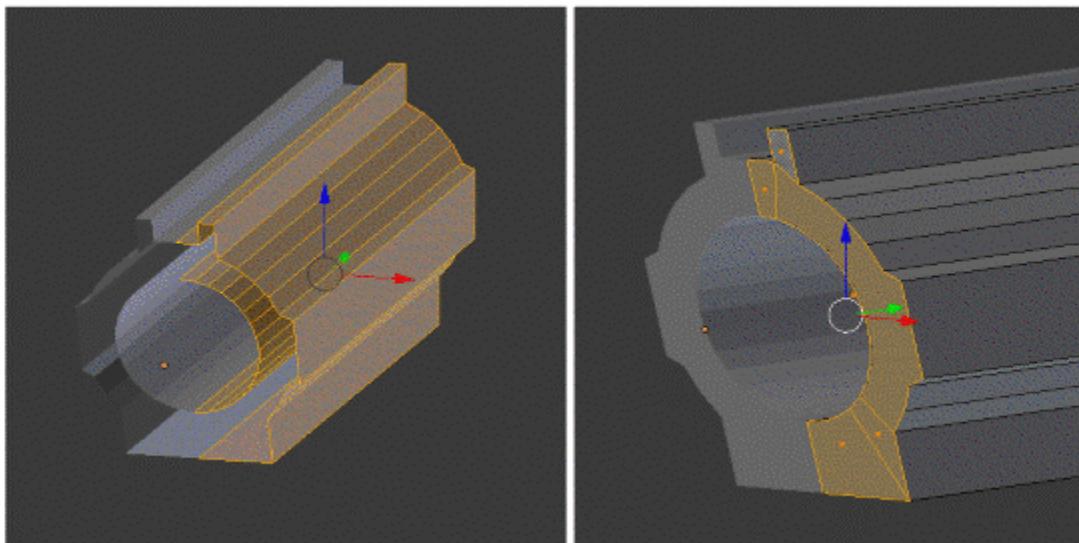
<pagebreak></pagebreak>

Next, we'll select one vertex in our front section of the barrel (the piece we don't want). By pressing Ctrl + L, Blender will automatically select all the linked (connected) vertices. You can then delete them, leaving just the section that you want.

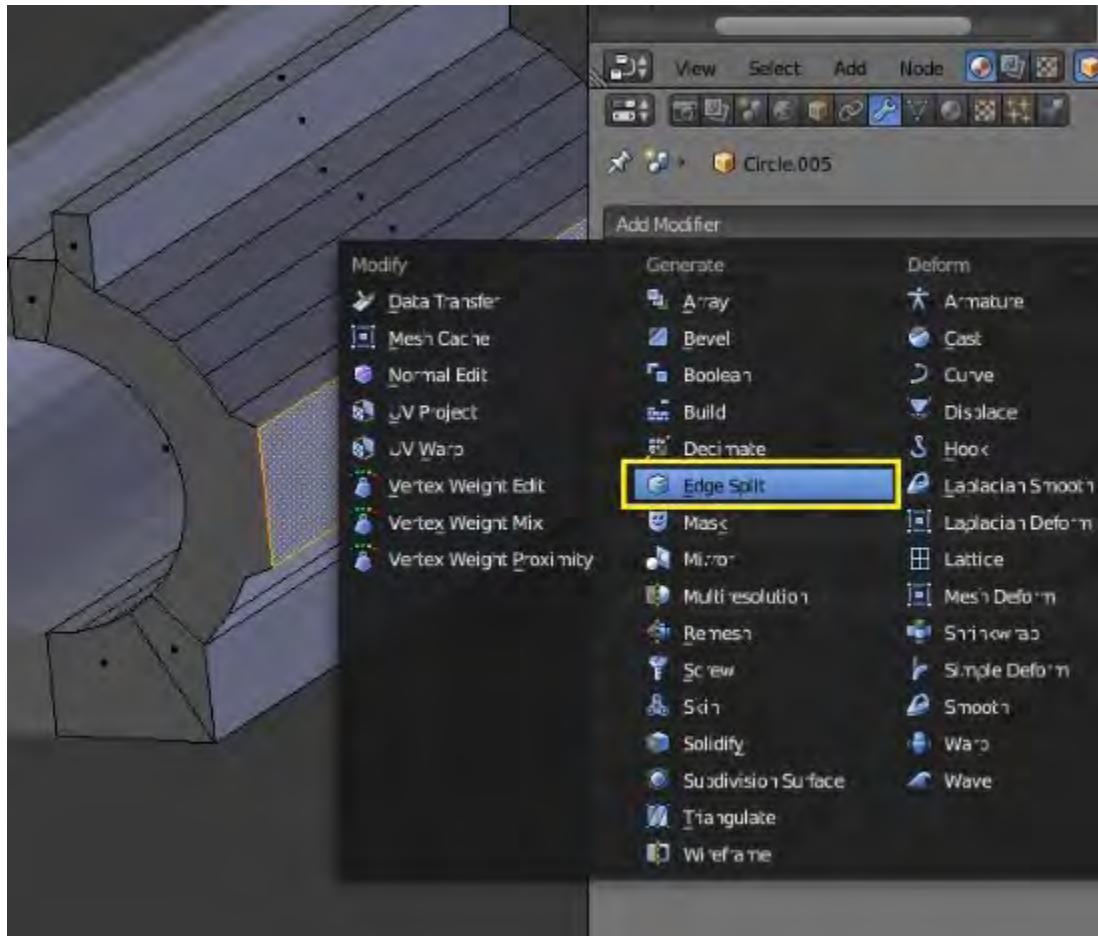


<pagebreak></pagebreak>

Next, we'll fill in faces at the front of the gun barrel again. Use a combination of quads and N-Gons to do this, so we can take a look at how each one is affected by the next step in our modeling process.

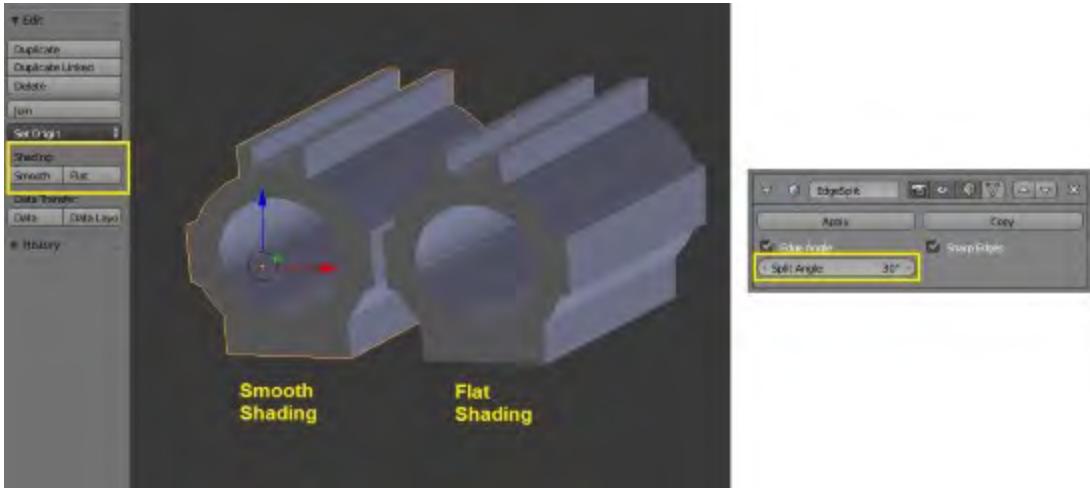


First, we'll add an Edge Split modifier to the gun. Most mechanical objects contain both smooth and sharp edges. Using the **Edge Split** modifier, we can tell Blender which angles should be sharp and which should be smooth. It's an incredibly useful tool for creating machines in Blender. To add it, just go back to your **Modifier** and select **Edge Split**.



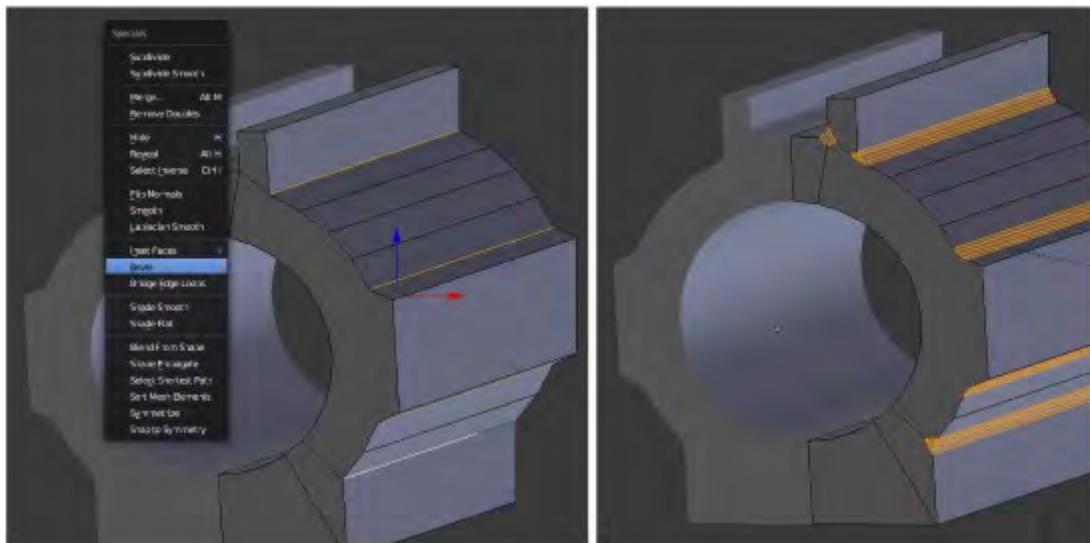
Once you add the modifier, go ahead and hit **Smooth** on the toolbar at the left-hand side of the screen. This will tell Blender that the entire object should use smooth shading except where the Edge Split modifier tells it not to. In this case, any angle that is greater than (or equal to) 30° will show up as **Flat** (sharp). Any angle less than this will show up as smooth. You can see the difference between smooth and flat shading in the upcoming picture.

By changing the **Split Angle** within the modifier, you're telling Blender which angles should be smooth and which should be sharp. There are times you'll want to change this number, but the default angle of 30° is actually very good most of the time. So, we'll leave this in place.

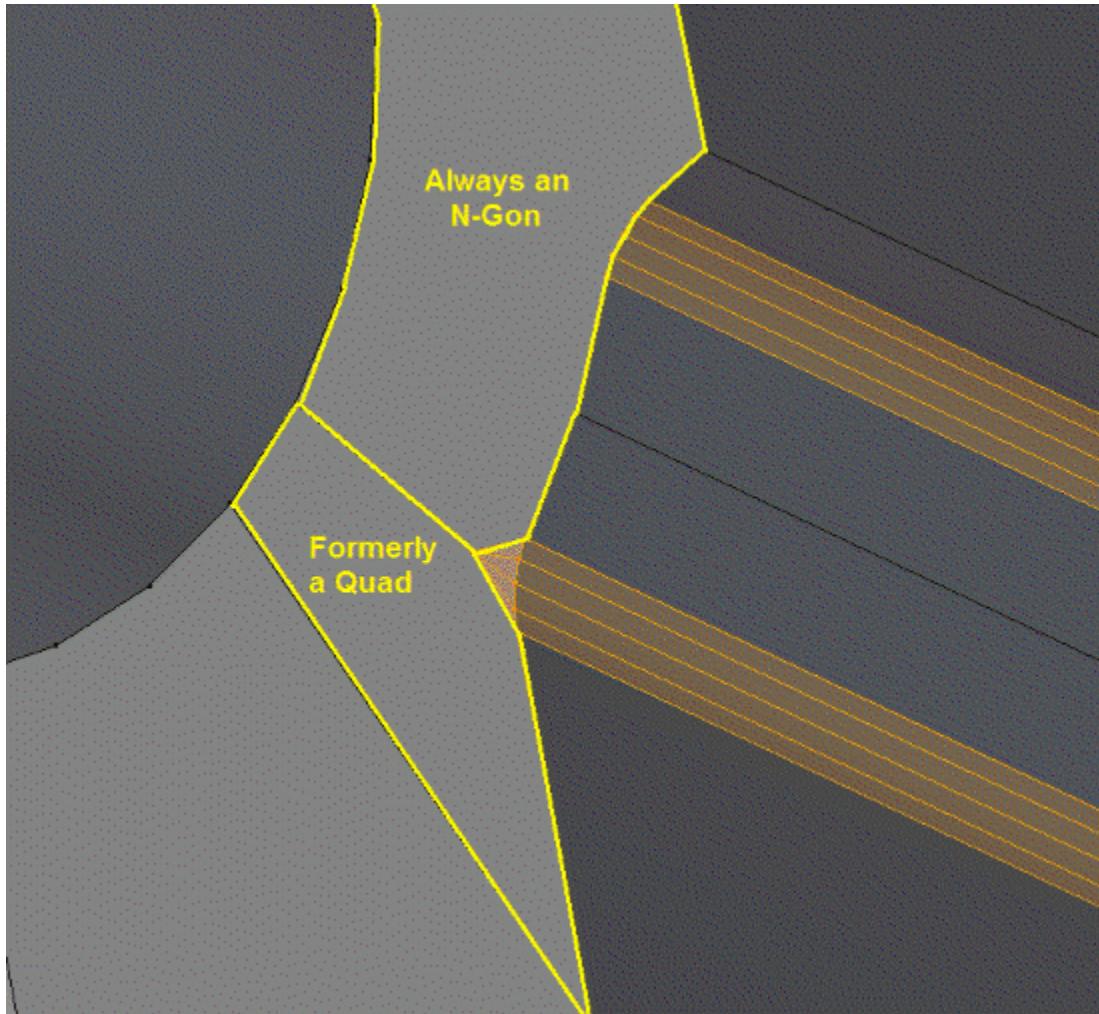


Now that we've got this taken care of, we'll smooth out our gun barrel a little bit. To do this, we'll use the **Bevel** tool. First, select the edges that you'd like to smooth out, then press W and select **Bevel** to activate the tool. Alternatively, you can activate the tool by pressing Ctrl + B.

With the tool activated, you can drag the mouse across the screen to change the amount of beveling on an edge. By **rolling the mouse wheel**, you will increase the number of **segments** to the bevel. If you want a sharp edge (like the corner of a piece of metal) it's best to just use one segment. If you're trying to create smooth curves, you can **roll the mouse wheel** up until the curve is sufficiently smooth. By selecting different edges on the gun barrel, you can smooth out the overall shape.

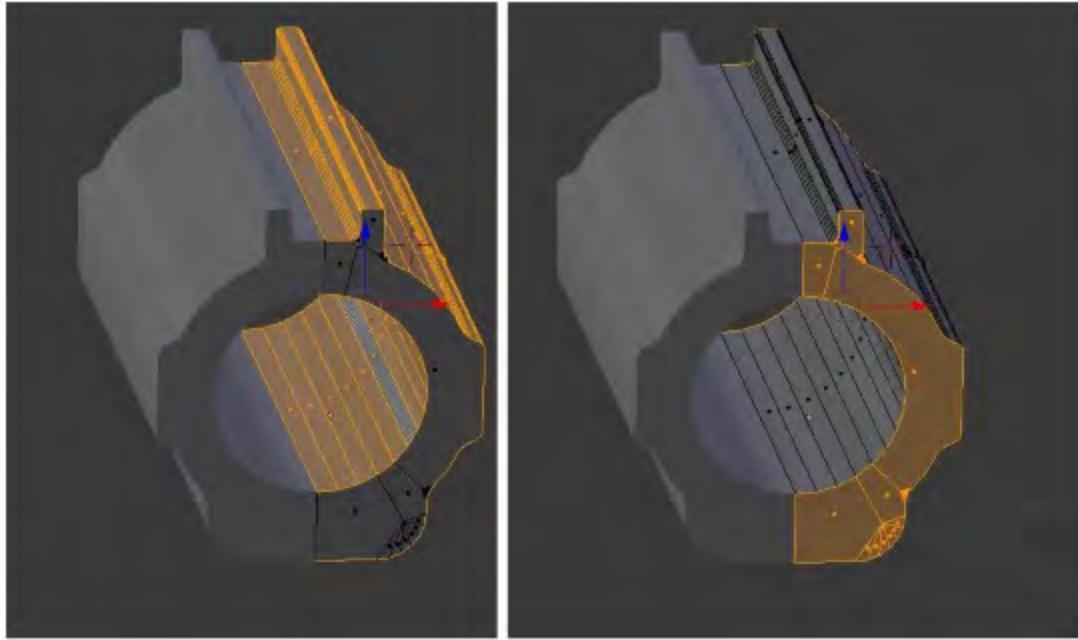


As we look at the front of the gun, we notice an interesting problem. Actually, it's not a problem; it's just the way that Blender does beveling. This is why we've filled in these faces at the front of the barrel, so we could observe the effects of the bevel tool on different types of faces. Here, you can see one of the key advantages of an N-Gon. It will not become distorted by the beveling process. This is not true for QUADS and TRIs:

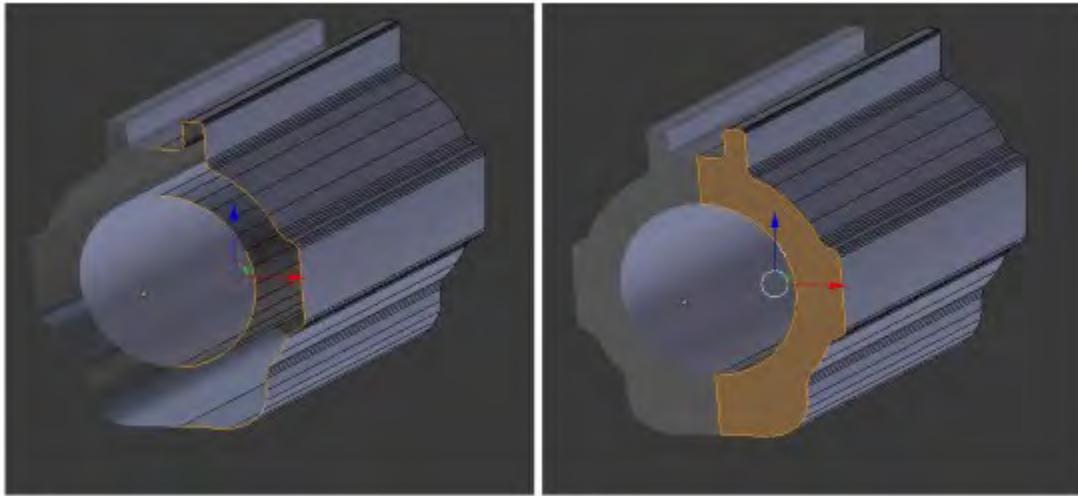


<pagebreak></pagebreak>

These faces on the front of the barrel will technically work, but they look a bit messy. If you want to clean things up, you can quickly select all the faces you want to keep, then press **Ctrl + I**. This is known as the select inverse function, and it does exactly what the name implies.



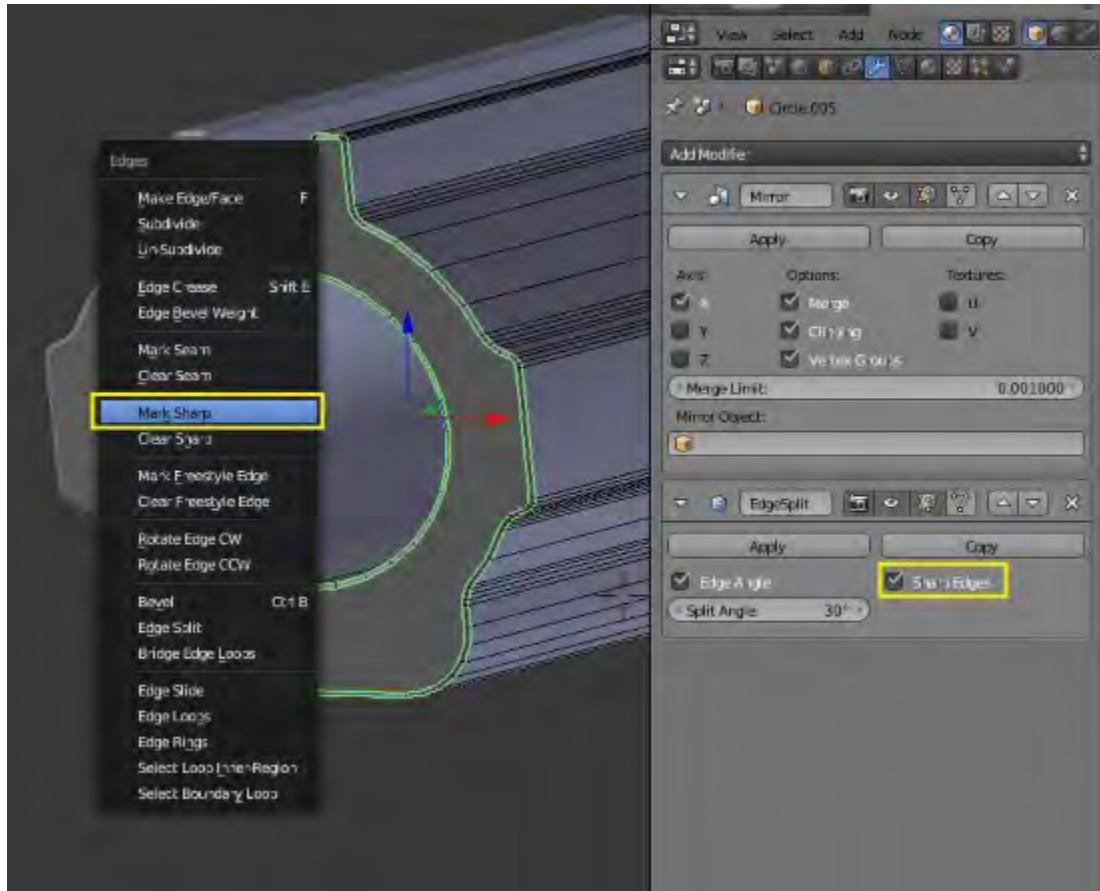
Next, you can just delete the faces with the **X** key. Then, select the edge loops that form the front of the barrel and press F. This will automatically fill in a single N-Gon on the front of the barrel.



Next, we'll use the Bevel tool again to just slightly bevel the front edges of the barrel. Since there are no perfect angles in nature, this is a very common way to establish a bit of realism for mechanical models.

<pagebreak></pagebreak>

When you've done this, just select the two edge loops that make up the barrel and press **Ctrl + E** and **Mark Sharp**. The angles will show up as sharp anyway (since they probably exceed 30°), but if you know that you want an edge to be sharp, it never hurts to mark it that way. You may make changes down the line that affect an edge's angle. This way, you never have to worry about it.

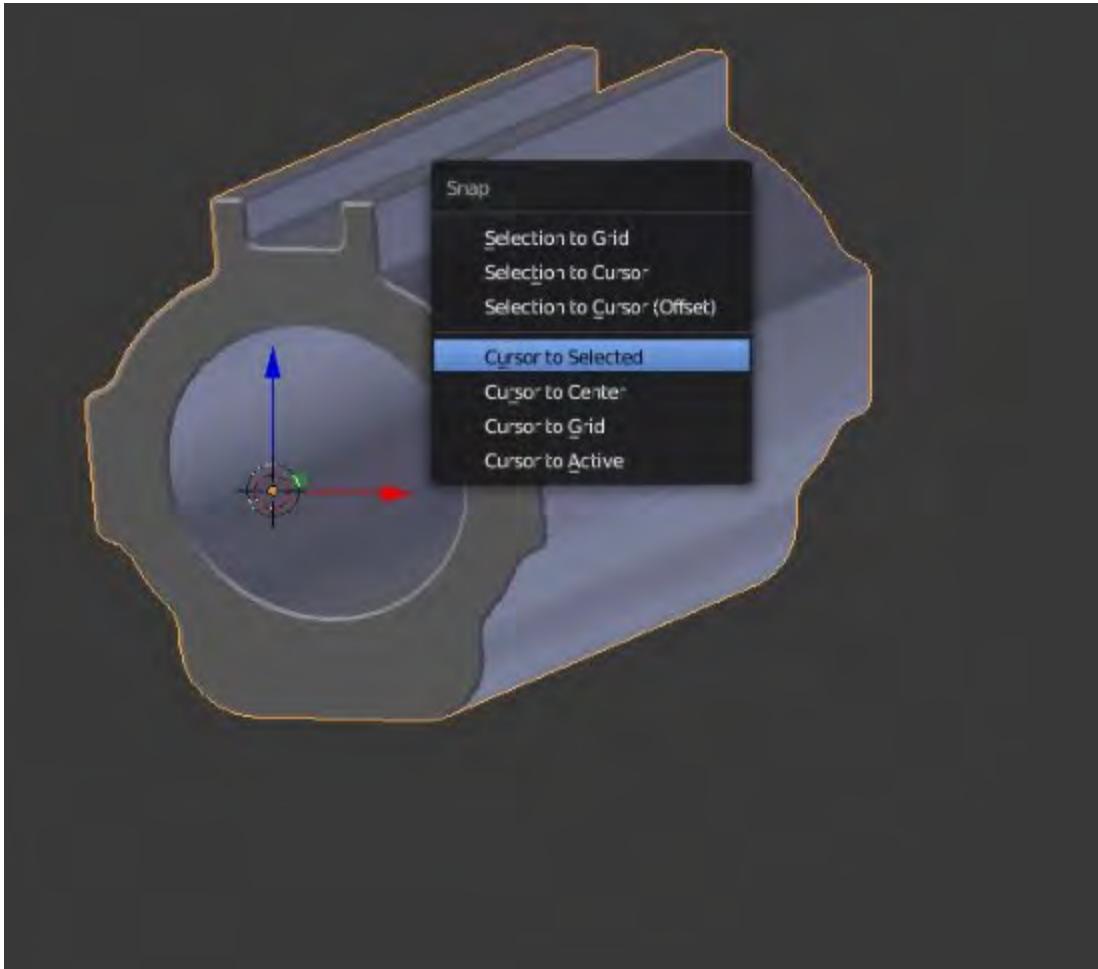


At this point, the basic shape of our barrel is complete. Make any final corrections that you'd like, and then we'll move on to the rest of the gun.

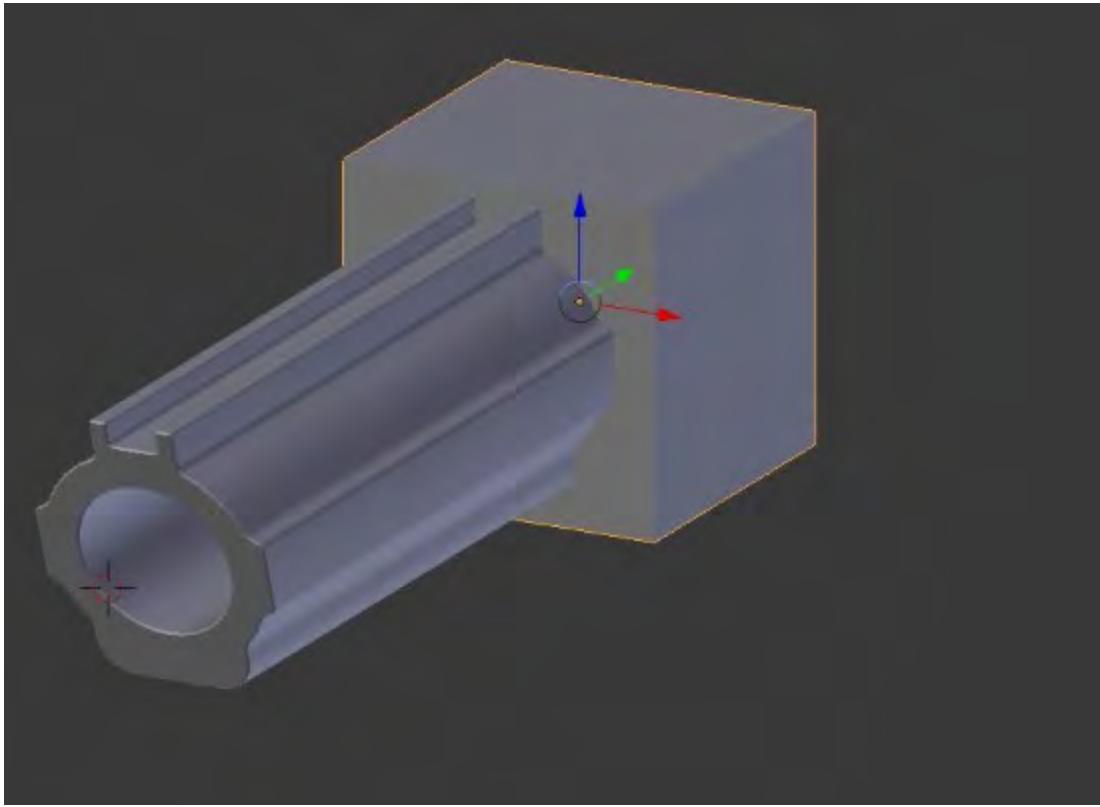
<pagebreak></pagebreak>

Modeling a handgrip and other pieces

The first thing we want to do is select the barrel again, and then press **Shift + S** and **Cursor to Selected**. This will place the 3D cursor at the origin point of our gun object. Any objects that we add after this will be automatically placed there:

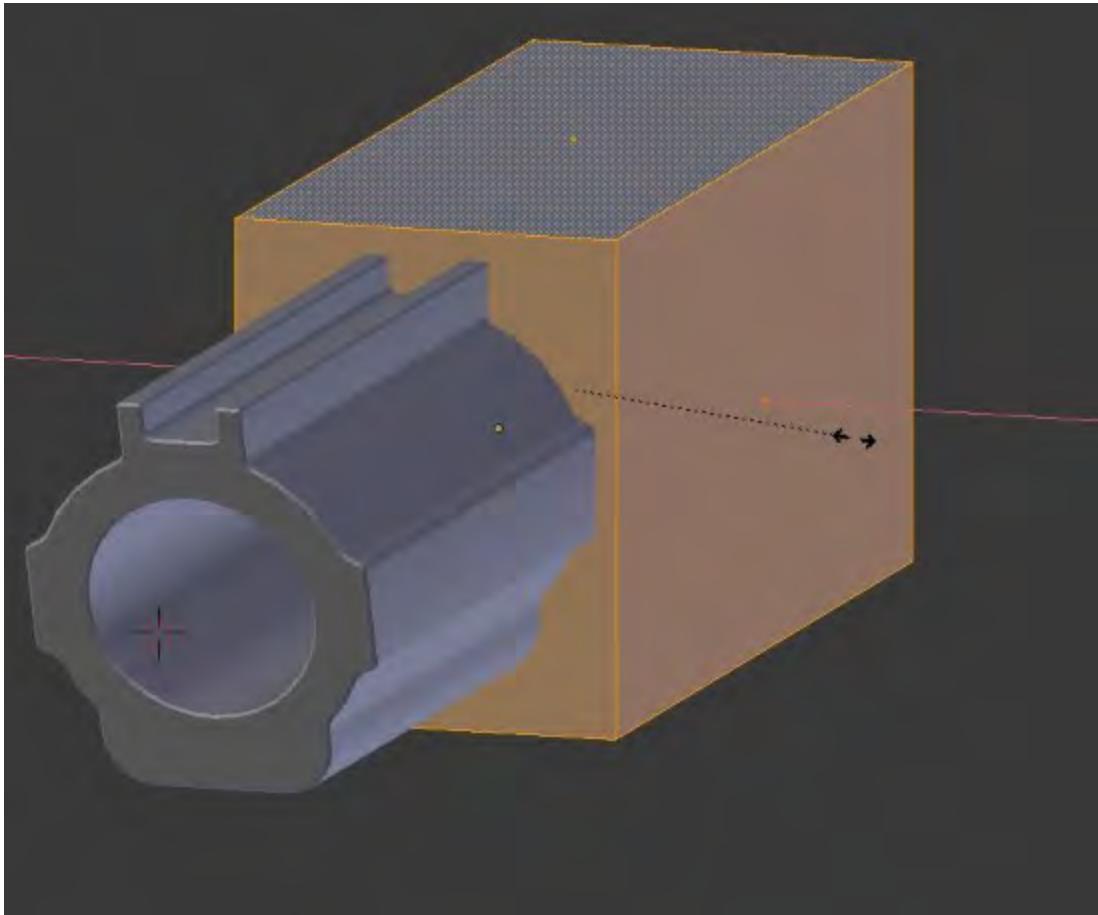


Next, we'll bring up the **Add** menu again and add a cube to our project. It will appear directly in line with the gun barrel. Depending on the scale of your barrel, the cube may appear slightly bigger or smaller than the image shown here:

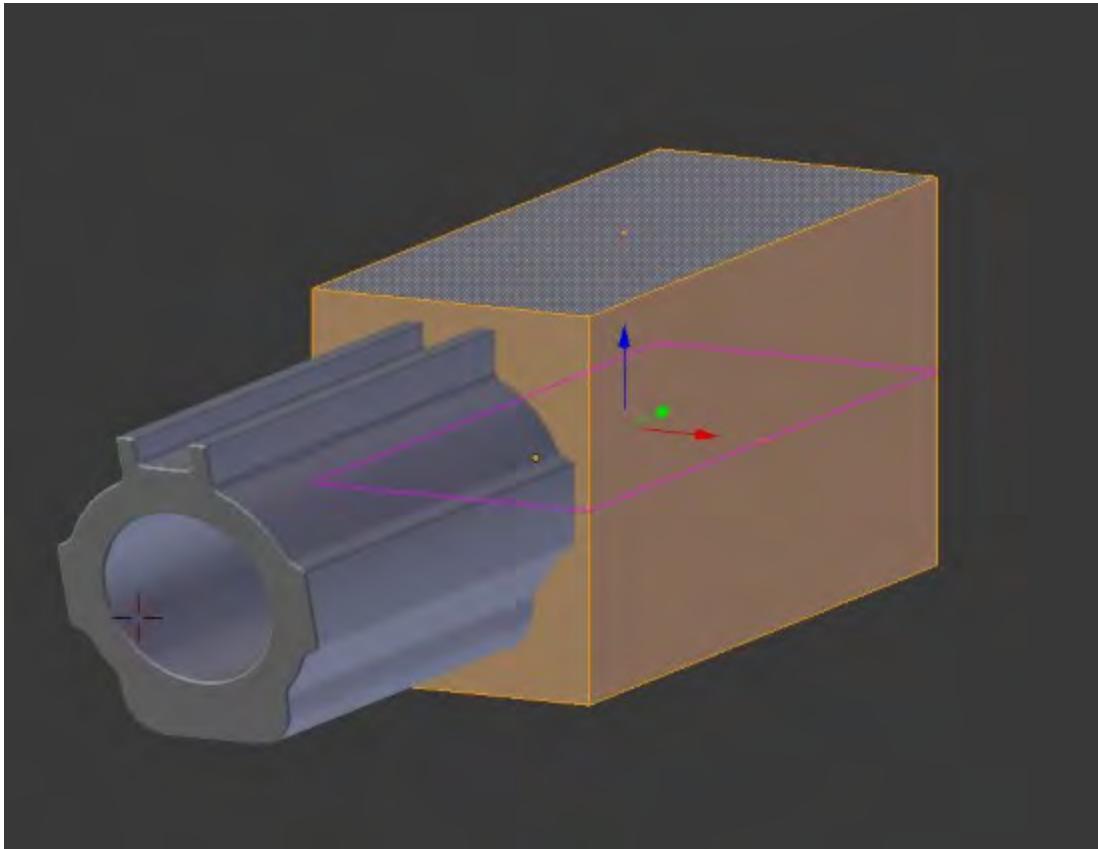


<pagebreak></pagebreak>

Tab into the **Edit** mode and adjust the scale until it's to your liking.

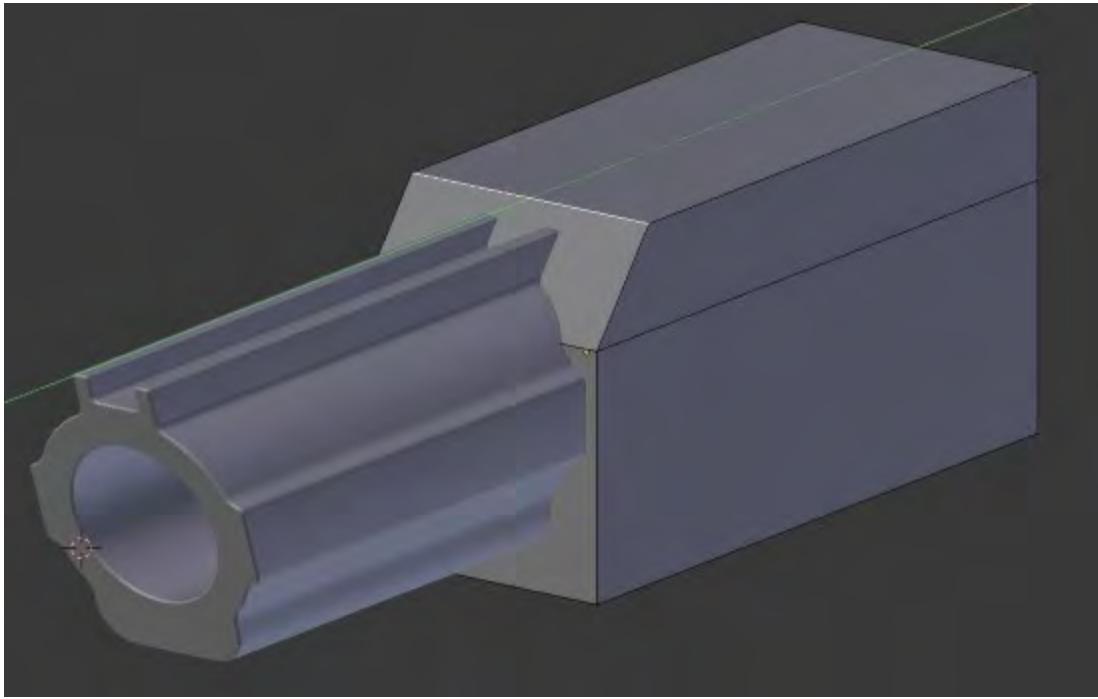


Next, use **Ctrl + R** to run a **loop cut** around the outside of the cube. This will enable us to make a few changes to the shape.

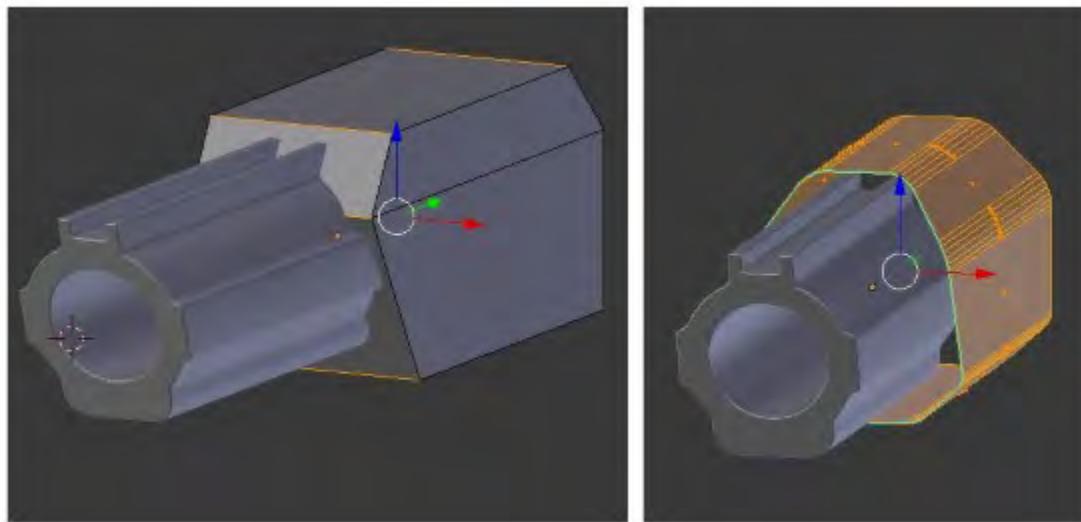


<pagebreak></pagebreak>

You can move the loop cut up or down as you'd like. When you've finished this, you can grab various edges and move them back or forward to create the desired shape for the gun's body.



Once you've formed the basic shape that you'd like, we'll delete the front and rear faces of the gun's body. Afterwards, you can go in and bevel the edges as much (or as little) as you'd like. Again, you can use the provided image at the beginning of the chapter if you'd like, or you can modify it to make it your own.



Once you have a shape that you like, we'll go ahead and join this object to our barrel. In order to do this, we'll need to make a few changes.

First, we'll delete the very top and bottom faces of the body. Then, we'll delete one half of it (make sure to delete the same half that you did with the barrel—we want all of the mesh to be on the same side).

<pagebreak></pagebreak>

Then, **Tab** out to **Object** mode. Select the gun body first and then the barrel. Press **Ctrl + J** to join the body to the barrel.

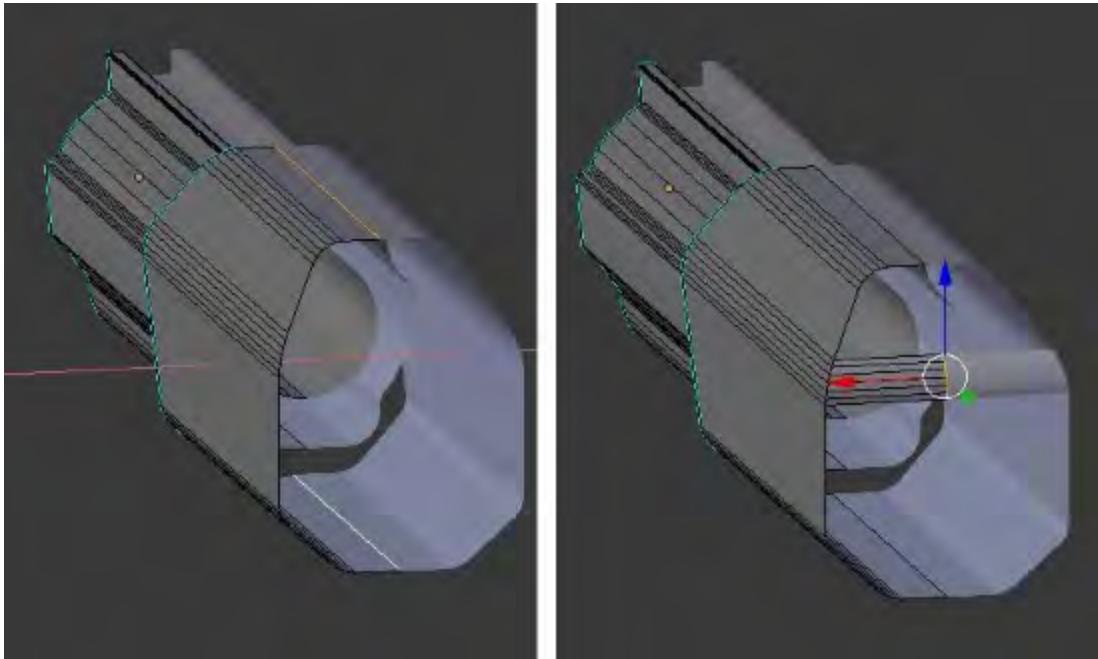
Note

The first object that you pick will always be joined to the second object.



Next, we'll grab the very top and bottom edges of our gun block and extrude them in toward the center (on the X axis). Since **Clipping** was selected on our **Mirror** modifier, the edges will not go past the object's centerline.

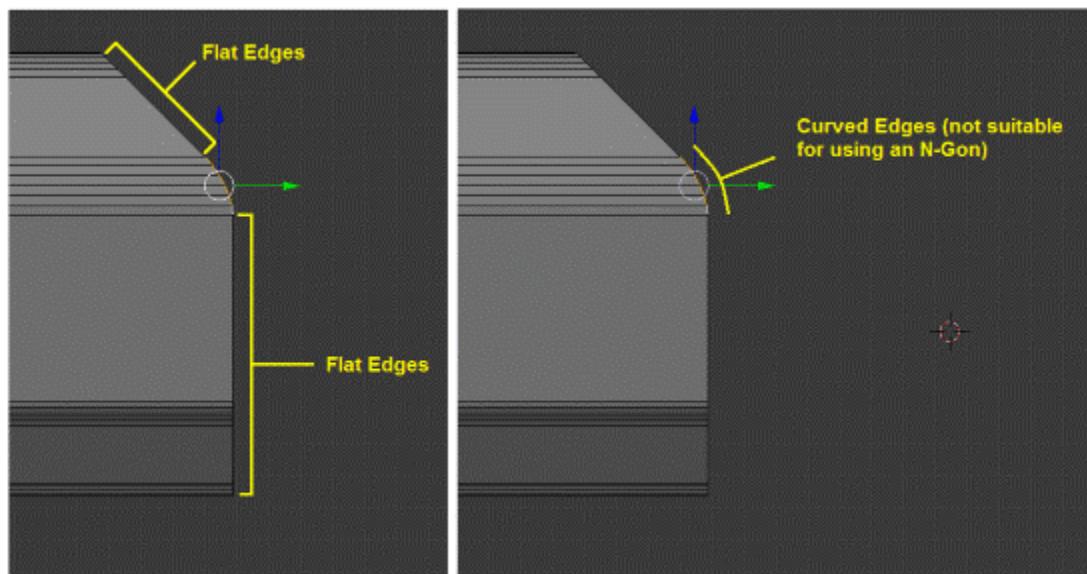
After you've done this, select the series of small edges (from the bevel) in the middle of the gun block and extrude these into the centerline as well.



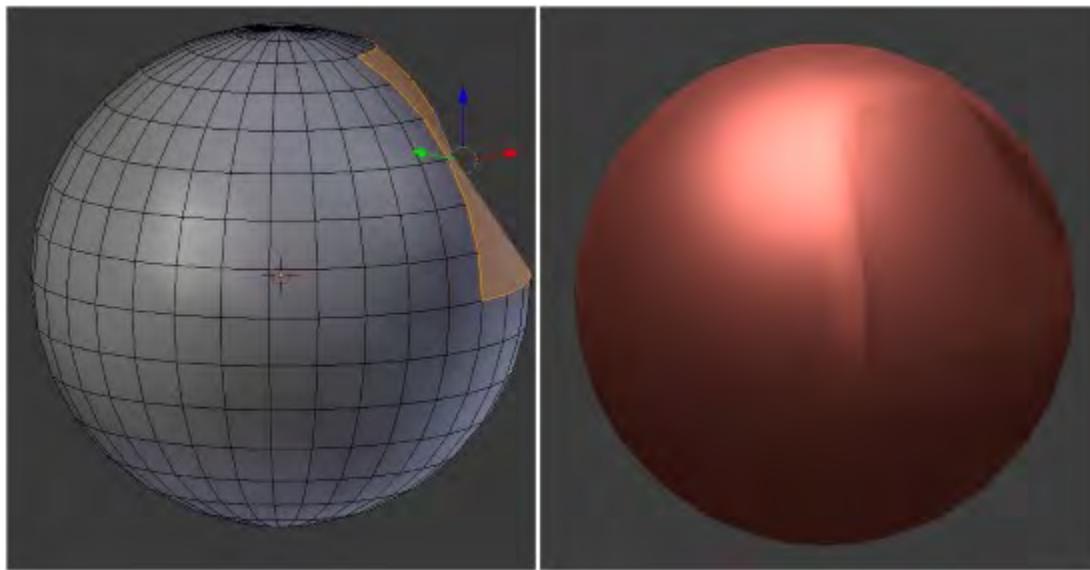
<pagebreak></pagebreak>

Next, we'll be filling in some faces on the back of the gun's body. We'll use a combination of N-Gons and quads to do this.

It's important to know where you can use an N-Gon and where you can't. The most important rule about N-Gons is that they must always be flat. As we look at the back of our gun's body, we can see a series of small curved edges in the middle:

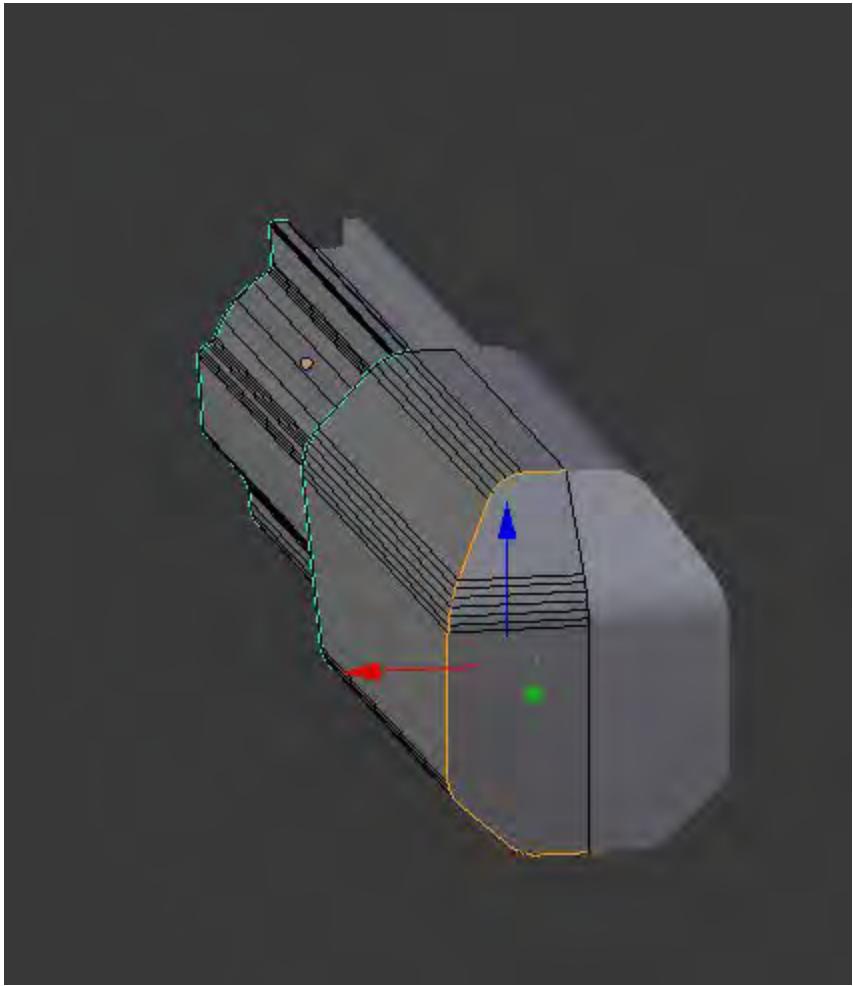


To better understand this rule, look at the following image. You can see that the portions of a sphere have been cut out and replaced with a single, curved N-Gon. The N-Gon is bending to connect various edges together. This dramatically affects both the shape and shading of your object, and it will create unwanted artifacts at render time:

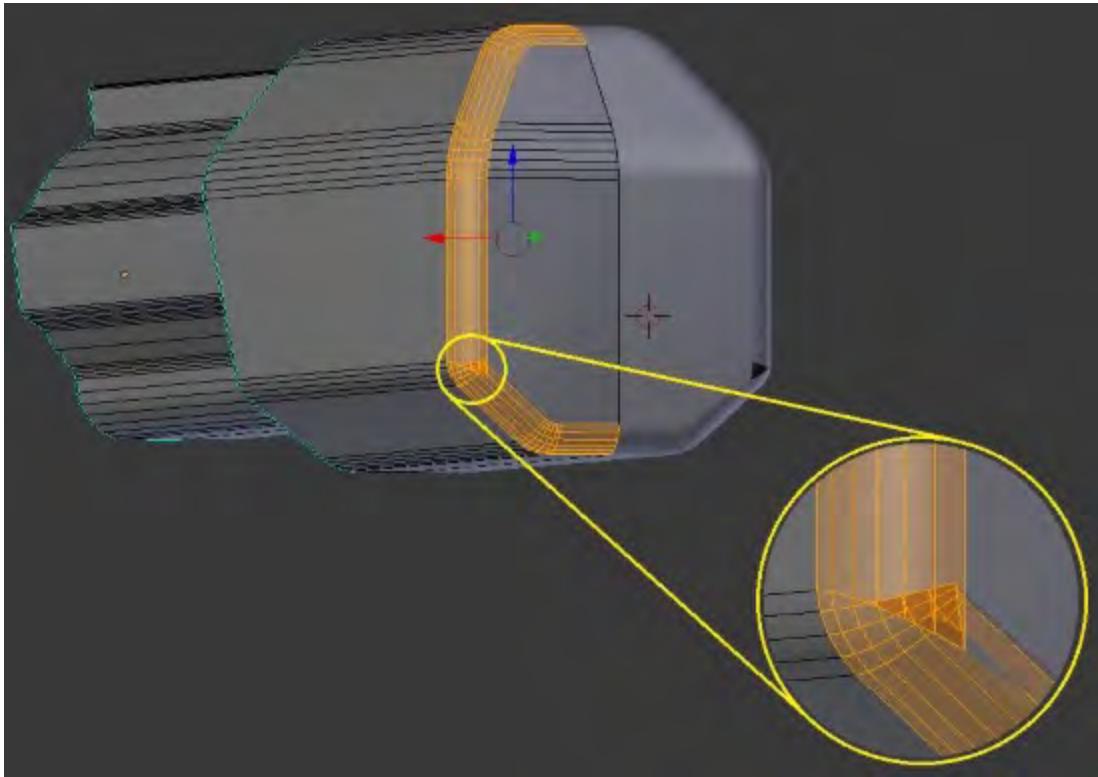


<pagebreak></pagebreak>

Going back to our gun's body, we'll fill flat edges in with N-Gons. When you're finished with this, select the outer ring of edges at the back.

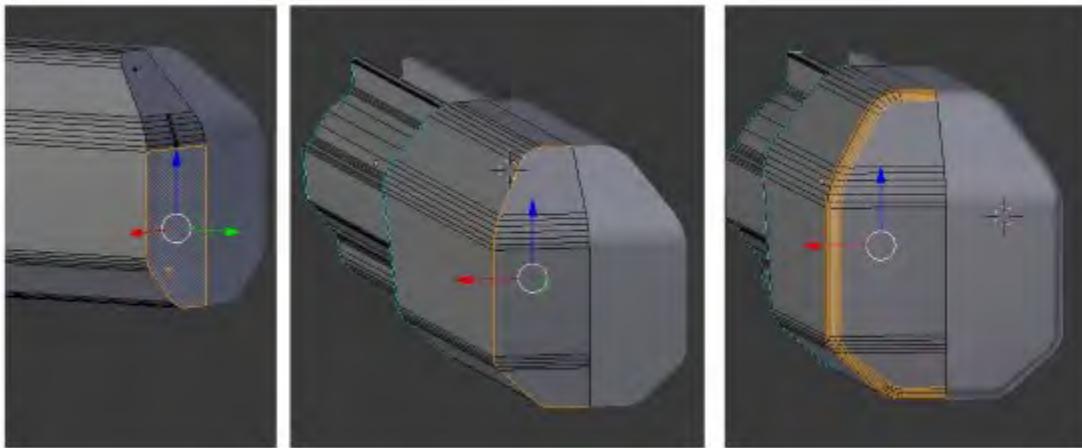


We'll attempt to bevel this in the same way we did before. However, depending on the exact shape of your gun's body, you may end up with something like this:

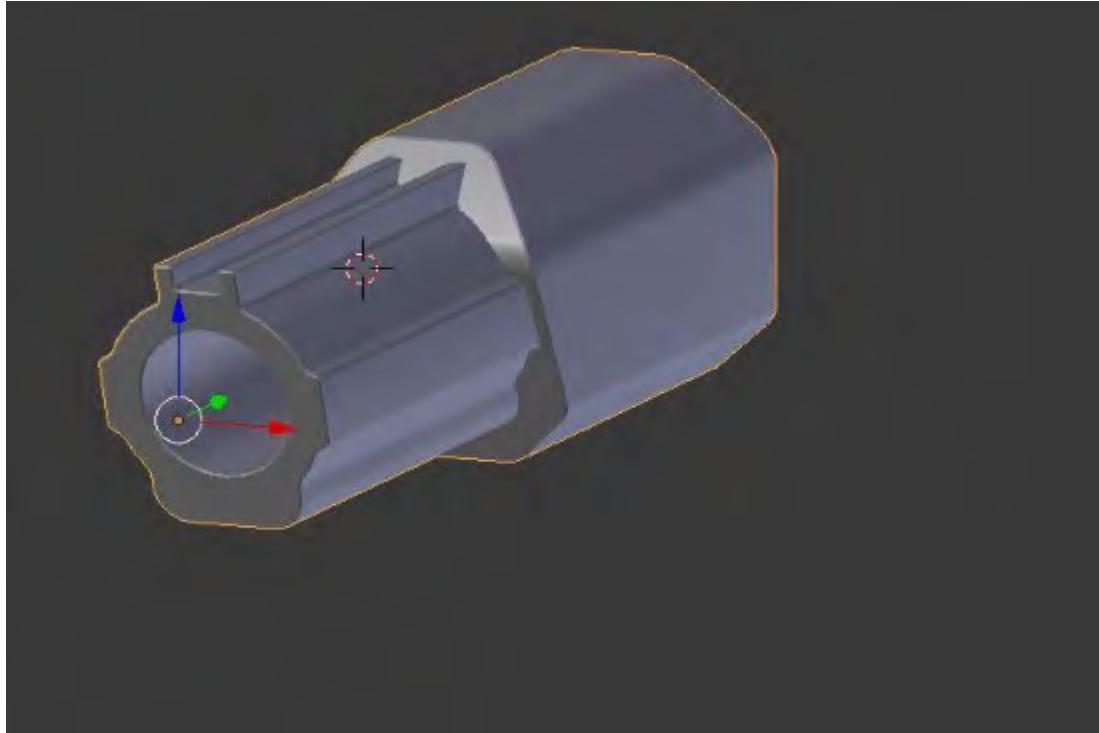


<pagebreak></pagebreak>

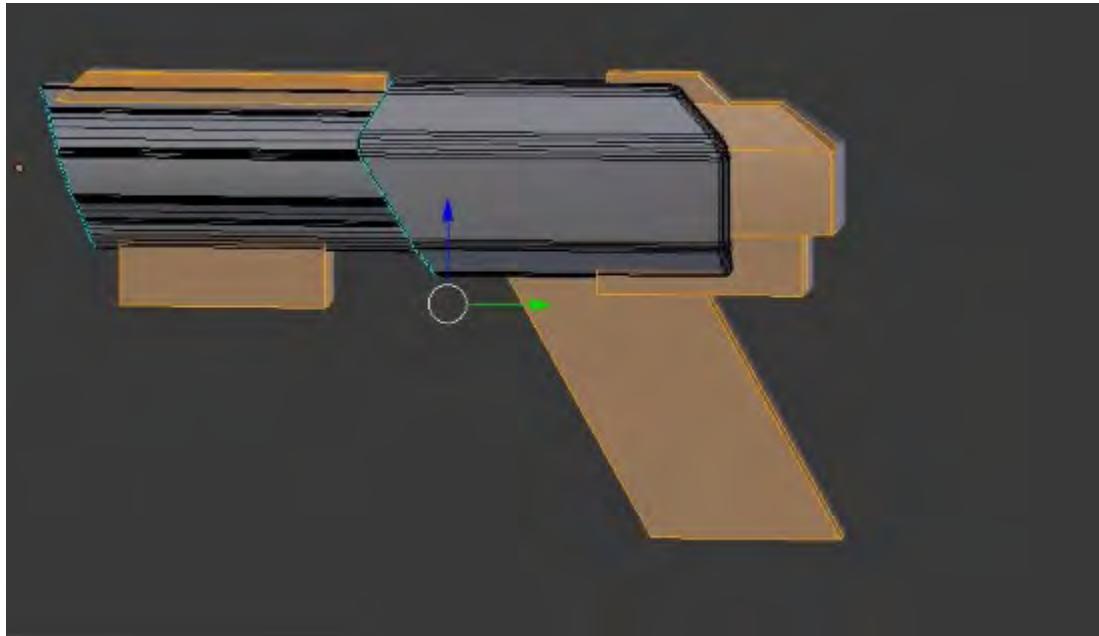
What you're seeing here is beveled edges that are overlapping each other. This is a fairly common occurrence with the bevel tool. When it happens, the best fix is to change the geometry a little bit. In this case, we'll remove the bottom N-Gon first. Then, we can extrude a series of small edges across, towards the middle. Once you've done this, you can fill the holes back in and re-bevel them:



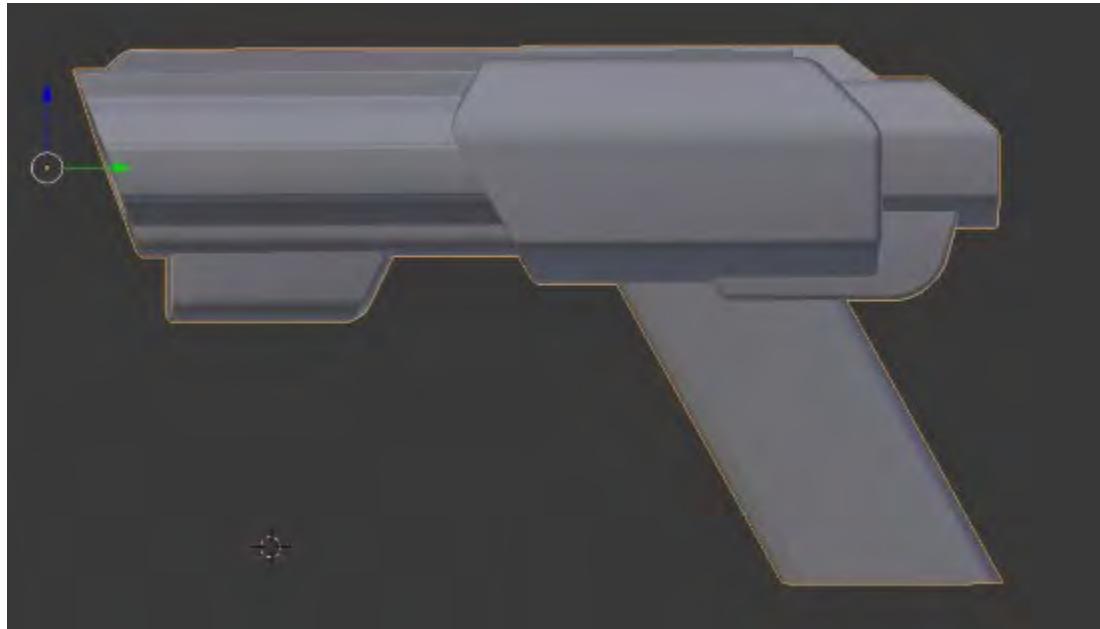
Repeat these steps with the front of the body as well.



At this point, we've covered all of the key topics for this chapter. Using the same methods that we've just looked at, you can now go through and create basic shapes for the rest of body. The following image is just a sample; feel free to use your imagination here!



When you've got basic shapes added and beveled, the result will look something like this:



Make sure that all of your different meshes are a part of the same object (the original barrel). When you're finished with this, we'll be ready to move on to the detailing phase.

Summary

In this section, we blocked out the basic parts of our gun. We used the **Mirror** and **Edge Split** modifiers as well as a number of mesh tools. In the next section, we'll use a number of different tools, modifiers, and techniques to bring out a lot of detail. These steps will not only allow us to finish our gun, but they will be useful for the rest of our projects as well. So, let's get started!

Chapter 2. Sci-Fi Pistol - Adding Details

In this chapter, we'll cover some additional topics as we work on our pistol. These include the following:

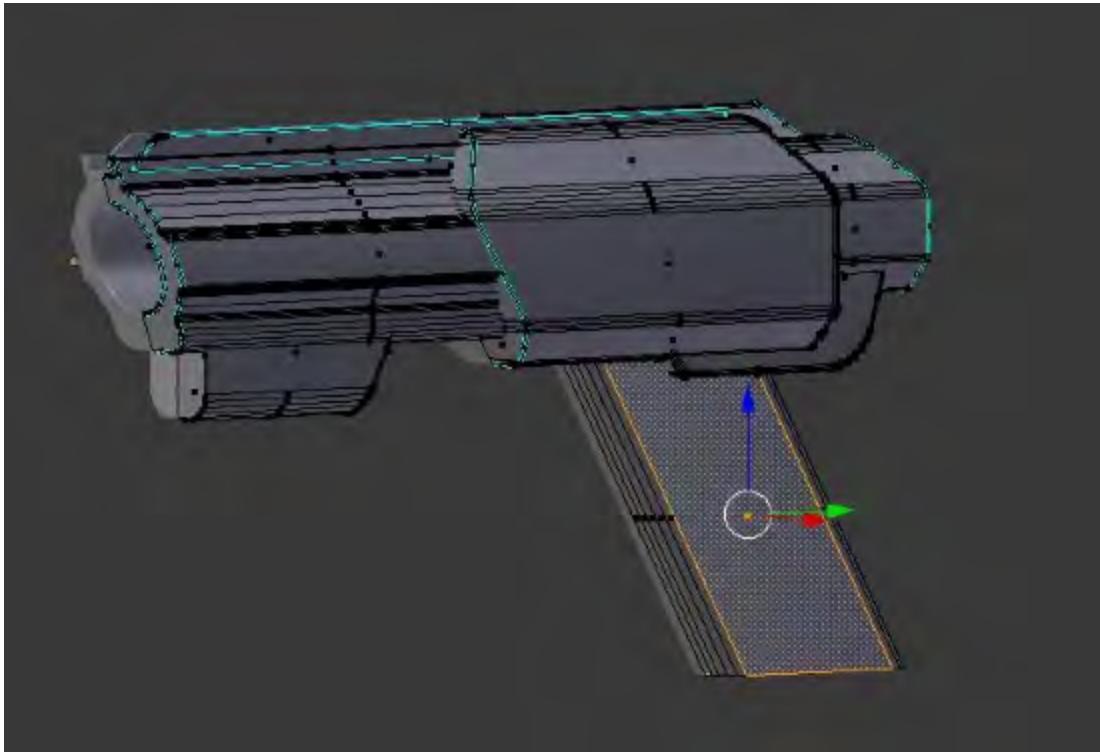
- Scaling along face normals
- The Shrinkwrap modifier
- Cutting shapes into curved surfaces
- The knife project tool
- The inset tool
- Proportional editing
- Pivot points for rotation
- Scaling along individual origins
- Creating pipes with torus objects
- The Array modifier

Finishing the handgrip

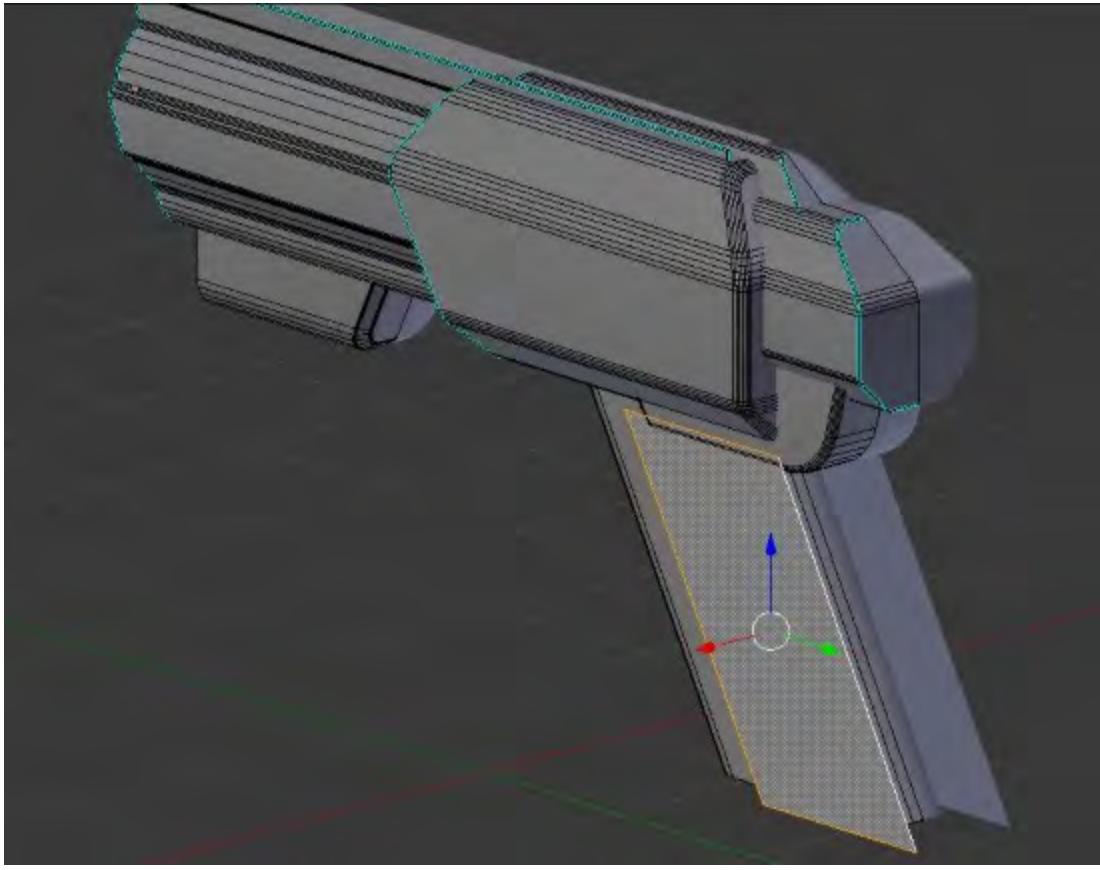
Right now, our handgrip (or handle) looks fairly plain. Let's work on this for a while, and see if we can get a better look at it.

<pagebreak></pagebreak>

The first thing we'll do is select the large face at the side of the handgrip:

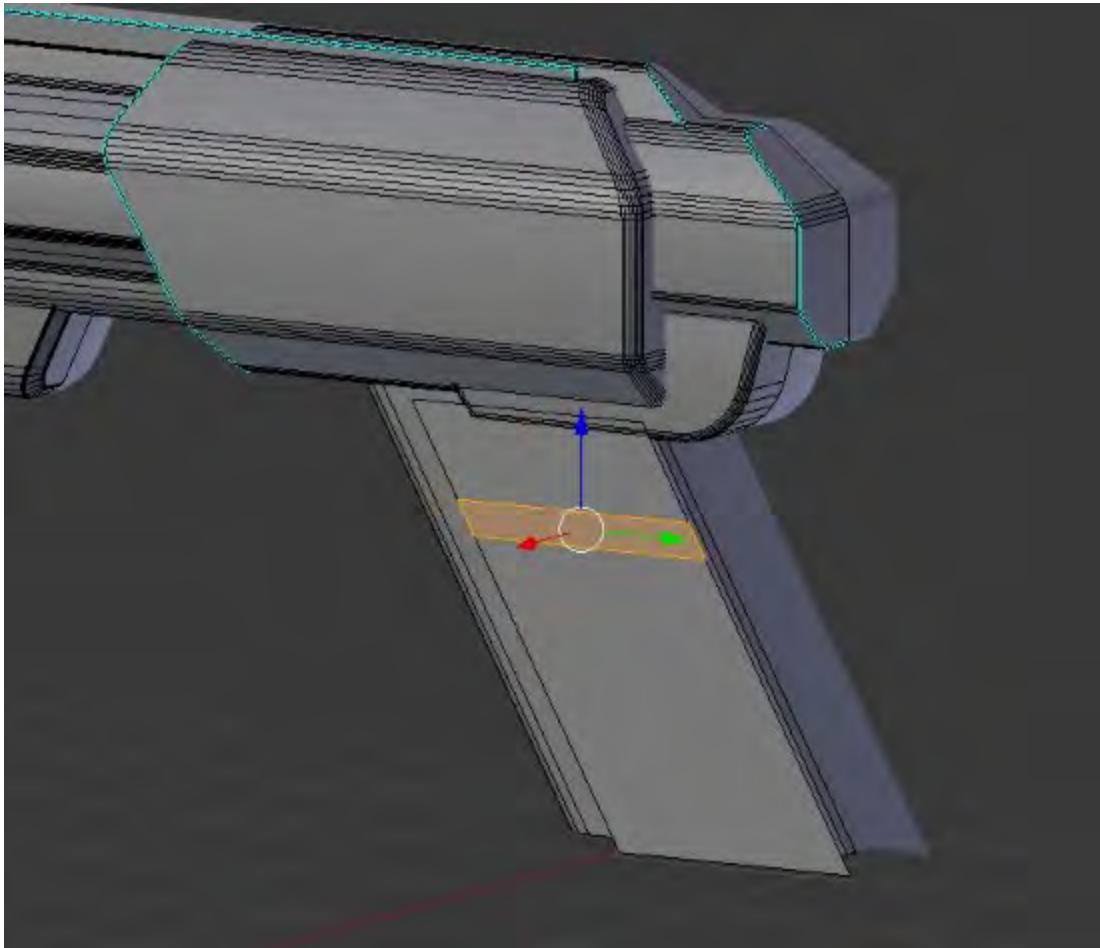


We'll duplicate this by pressing Shift + D and move it back a bit:

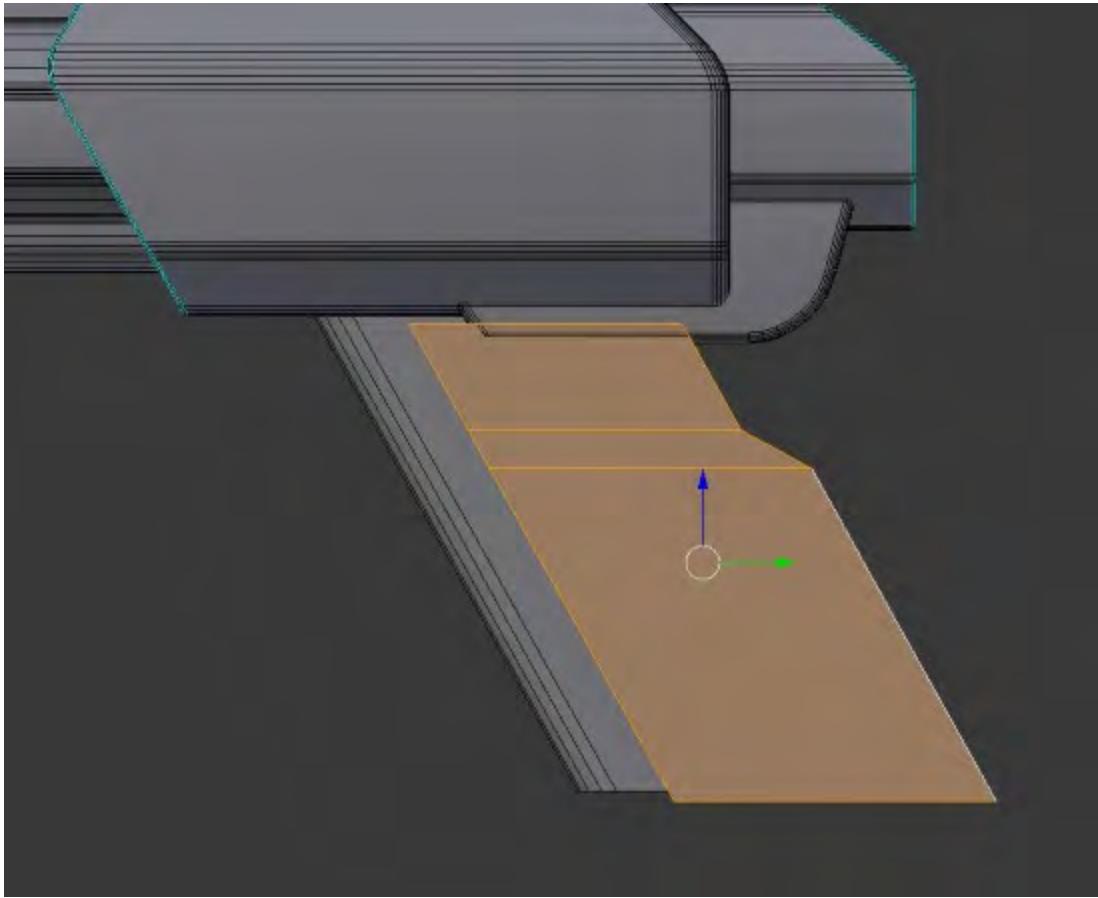


<pagebreak></pagebreak>

To add a little more detail, let's run a couple of loop cuts across this face by pressing **Ctrl + R + wheel up** (rolling the mouse wheel changes the number of edge loops):

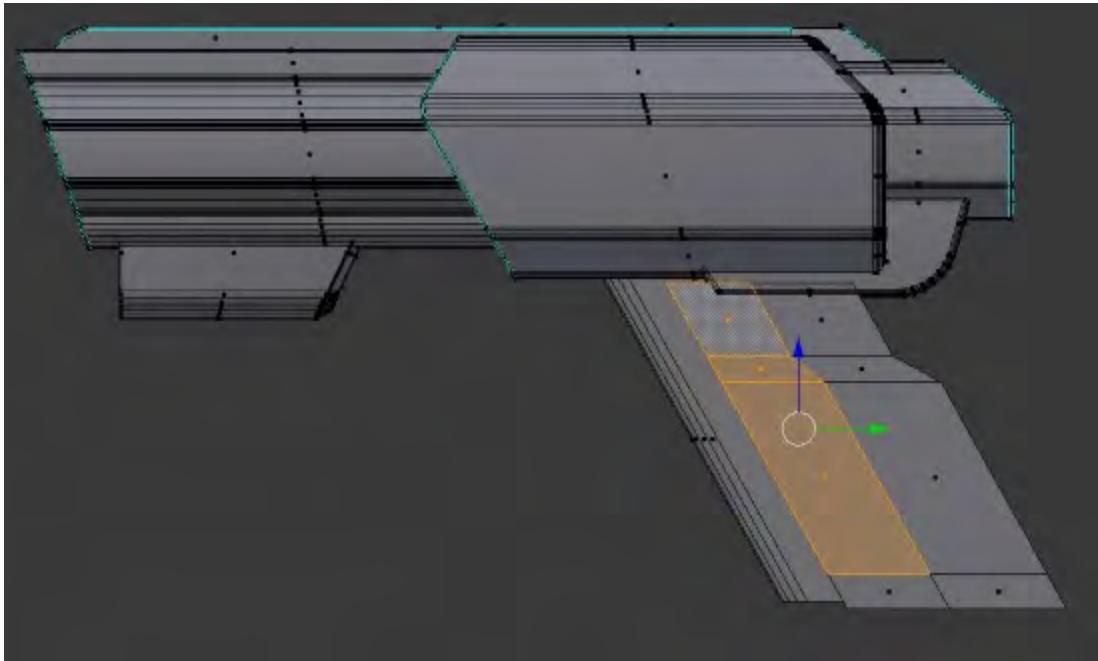


Then, we can drag the lower section back a bit by selecting the bottom-right edge and moving it to the Y axis:

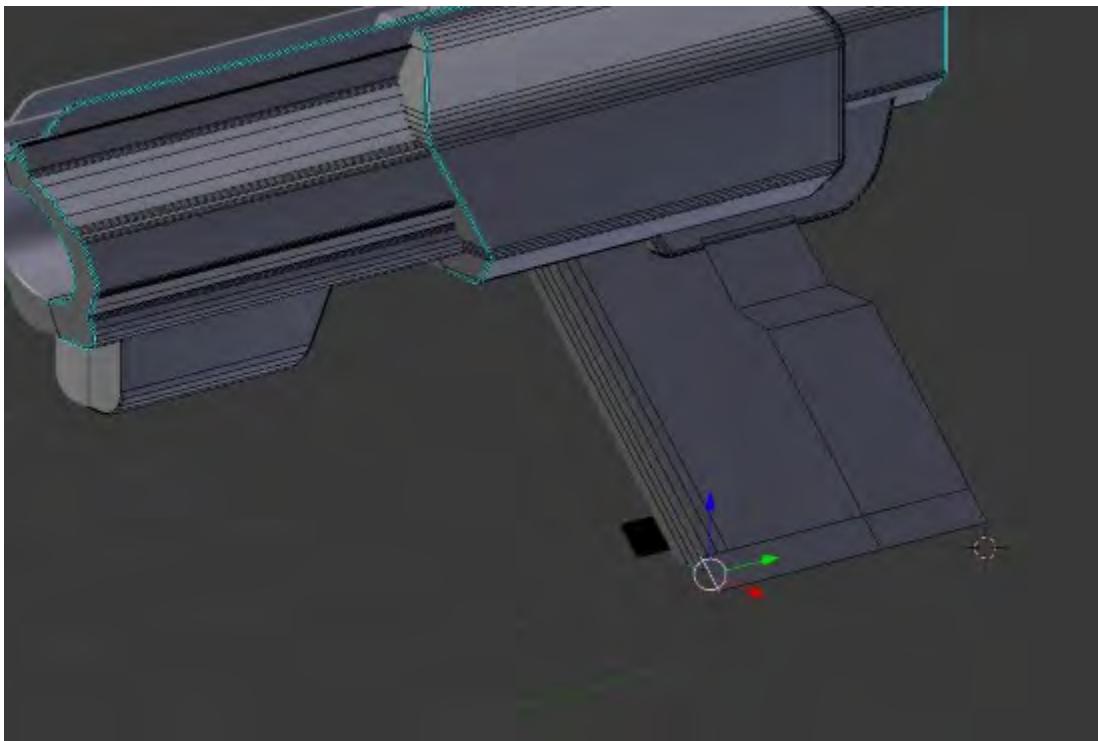


<pagebreak></pagebreak>

Next, let's run another edge loop vertically so that we can start forming the shape of the cut-out (for the grip):



We'll delete these unneeded faces. Next, let's grab this front edge and pull it forward:

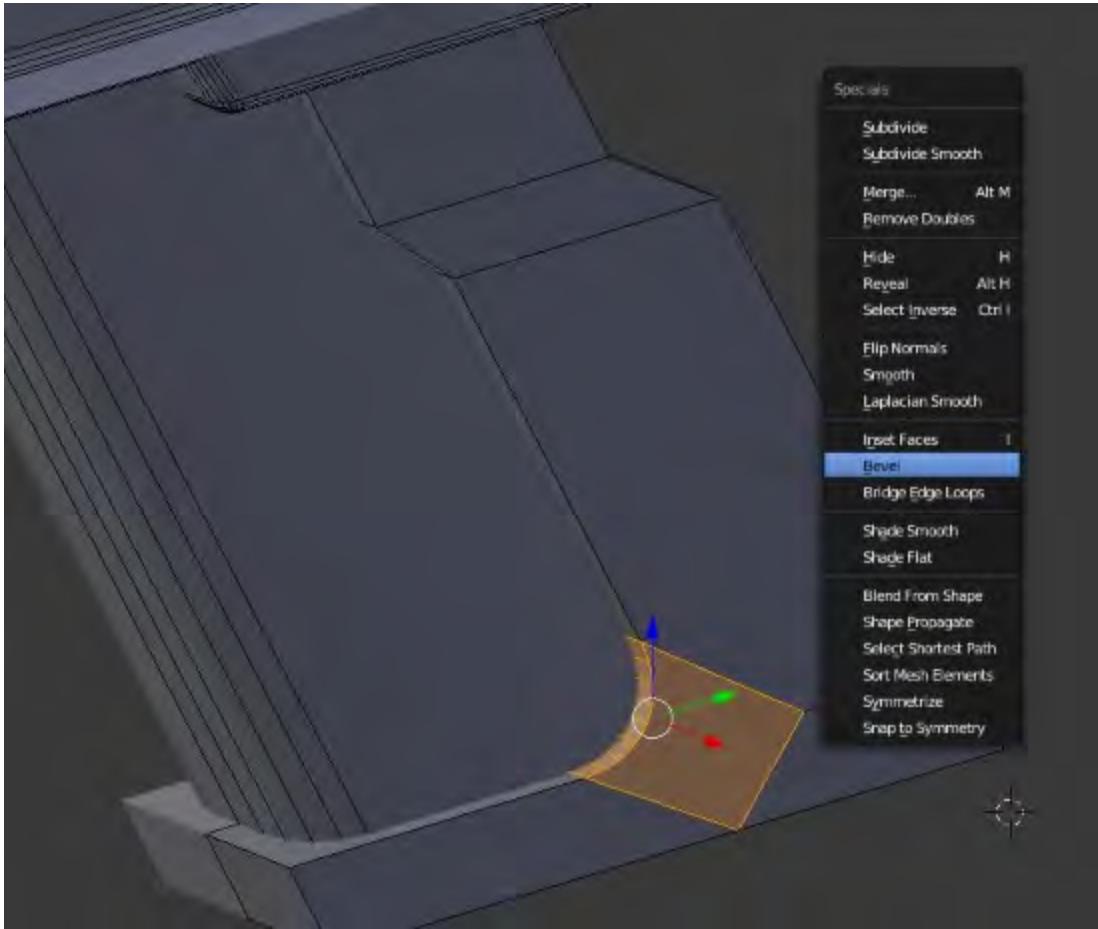


<pagebreak></pagebreak>

Then, we'll extrude everything across the whole plane to get the basic shape. Before going further, you'll want to remove the faces inside of the mesh (right where the two have joined at the mirror modifier). We don't want any faces inside of our mesh as it will affect shading later on.

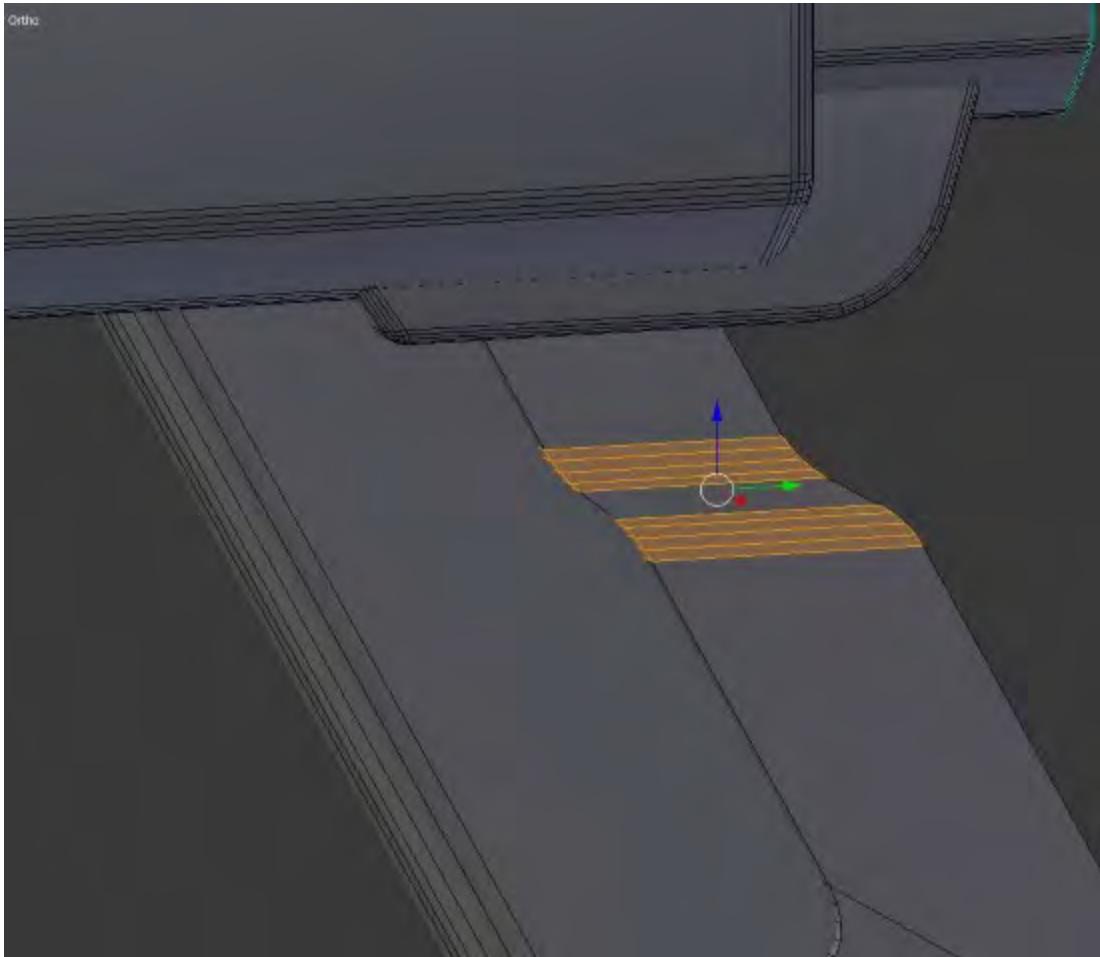


Let's **Bevel** this corner edge at the bottom to get a more rounded shape:

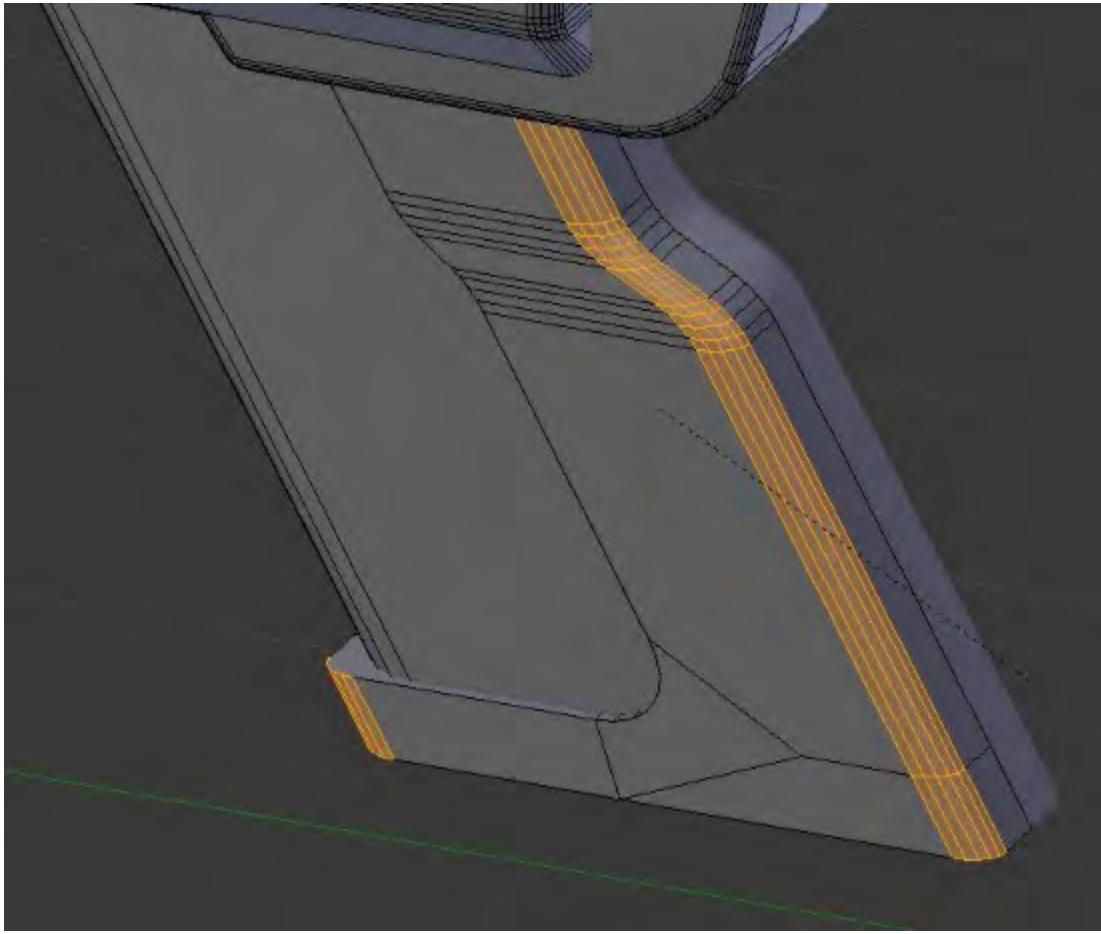


<pagebreak></pagebreak>

Next, we'll **Bevel** these edges here as well for a softer transition:

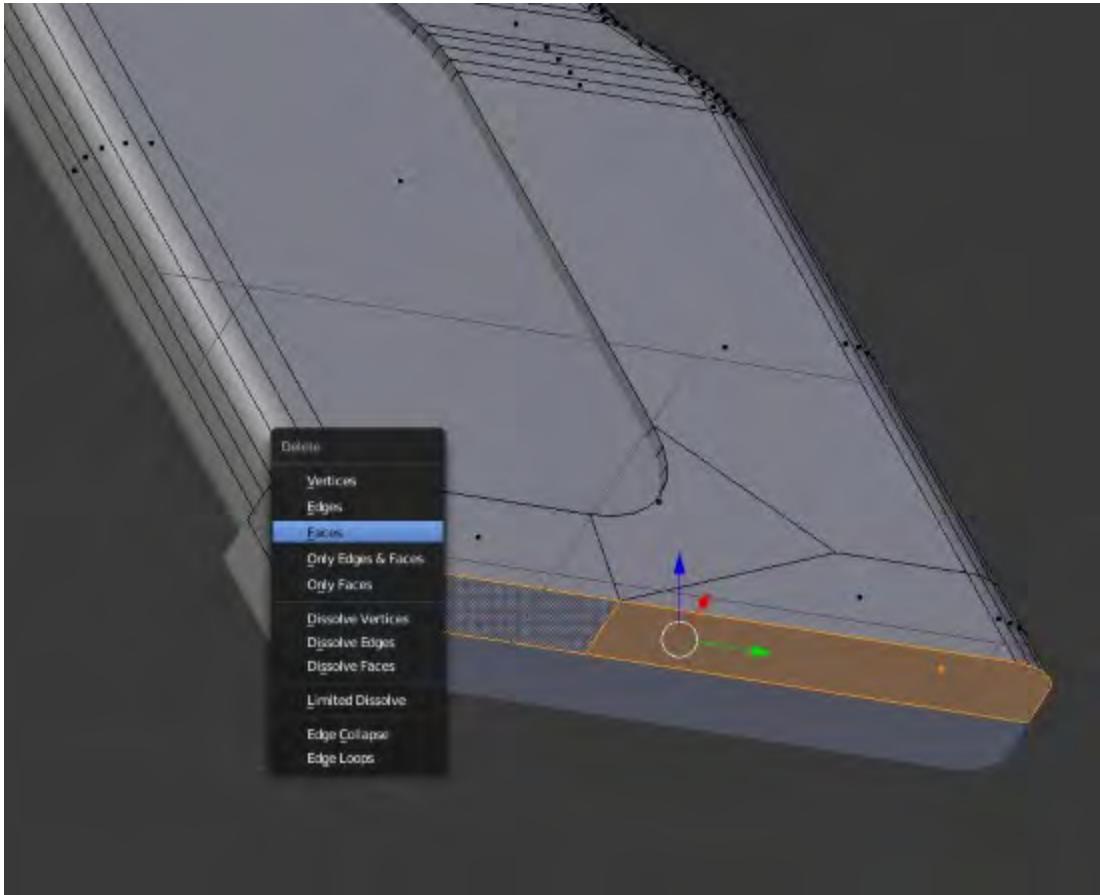


Next, we'll **Bevel** the corner edges so they're nice and smooth:

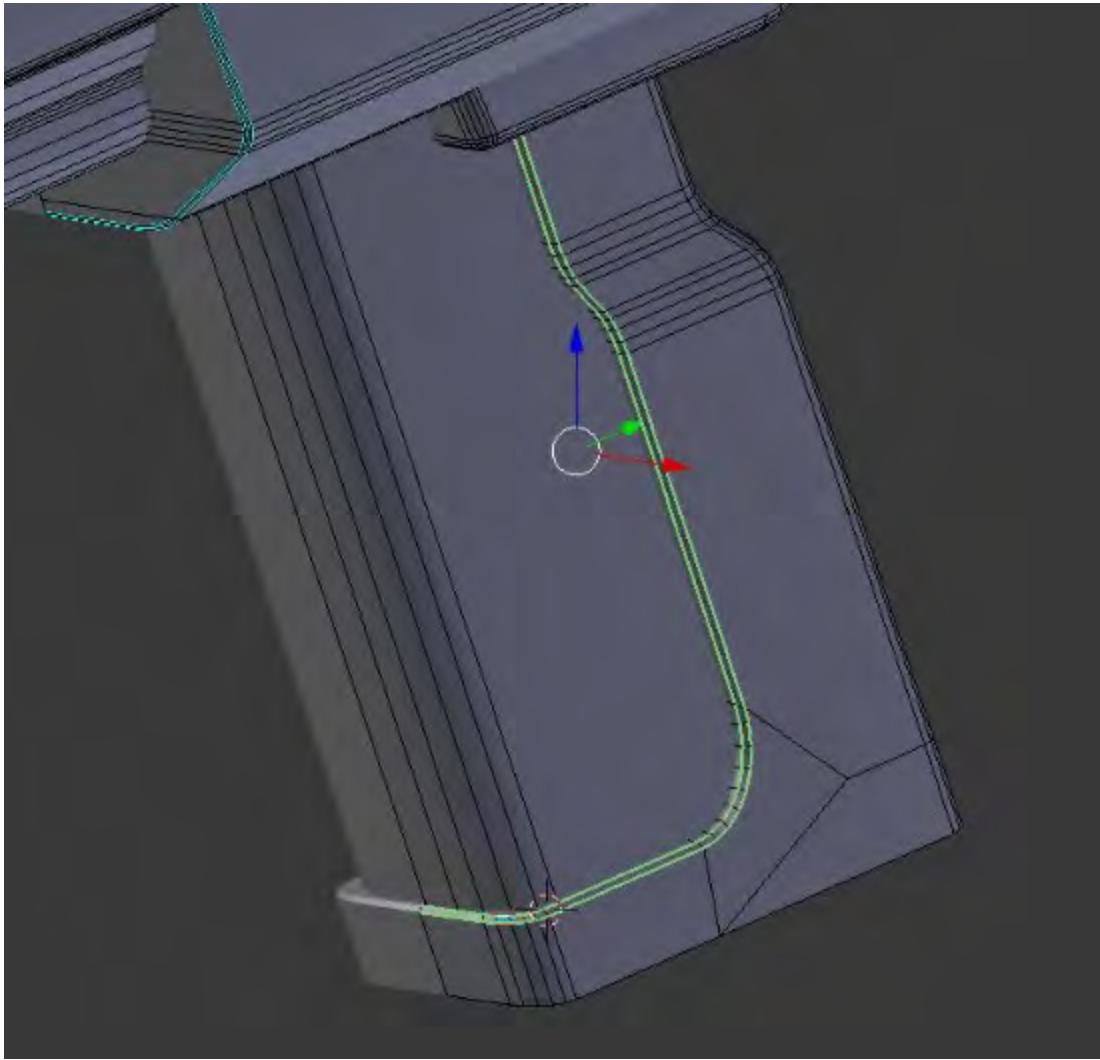


<pagebreak></pagebreak>

We can get rid of the **Faces** on the extreme bottom here as well:

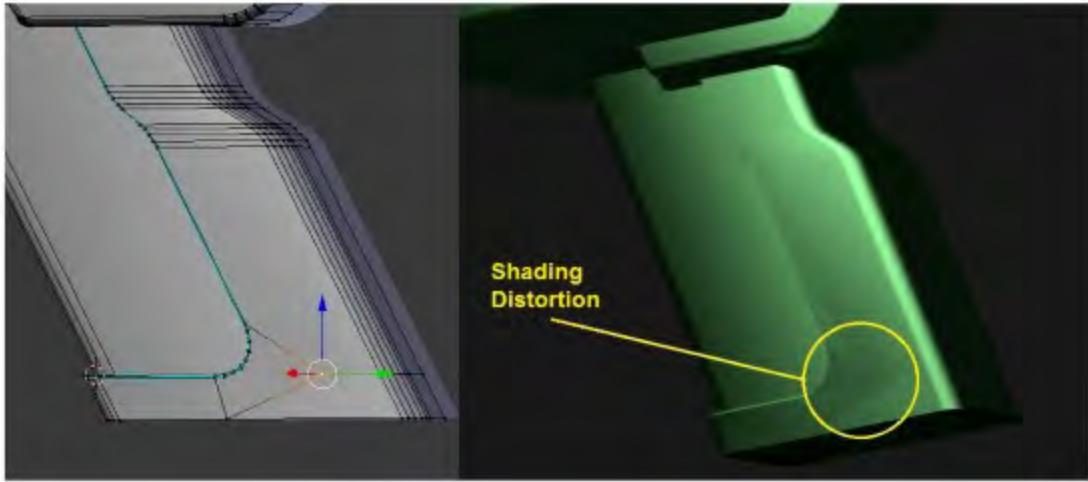


Next, let's run a hard (single) bevel along the edge that connects to the handgrip and mark it as sharp for the **Edge Split** modifier to work:



<pagebreak></pagebreak>

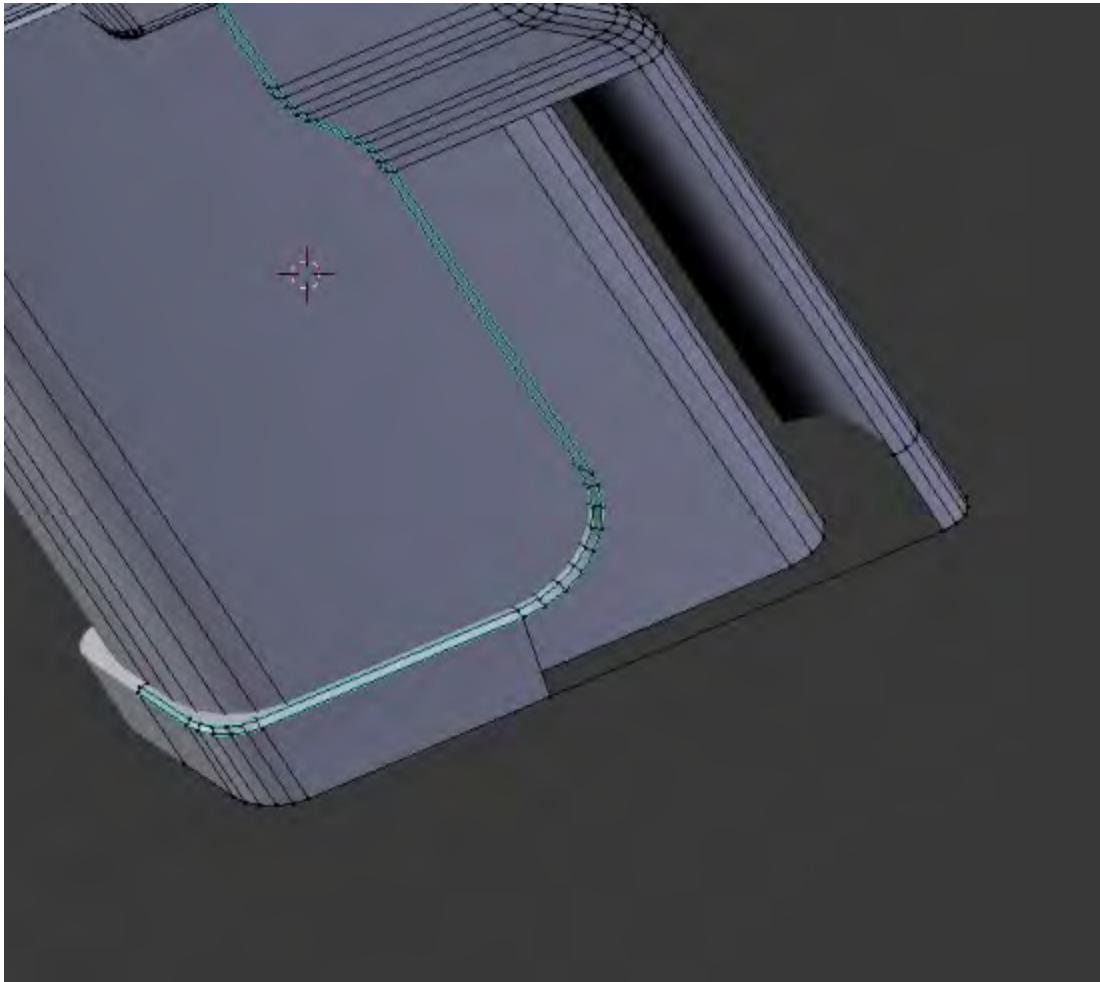
This looking pretty good; one problem we have here is that our earlier beveling has left a single vertex in the middle of the handle. If the back of the handle were hard (sharp), this wouldn't be a problem; because it's curved, it will affect the shading. I've added a temporary material and a rendering setup just to demonstrate what the problem will be here:



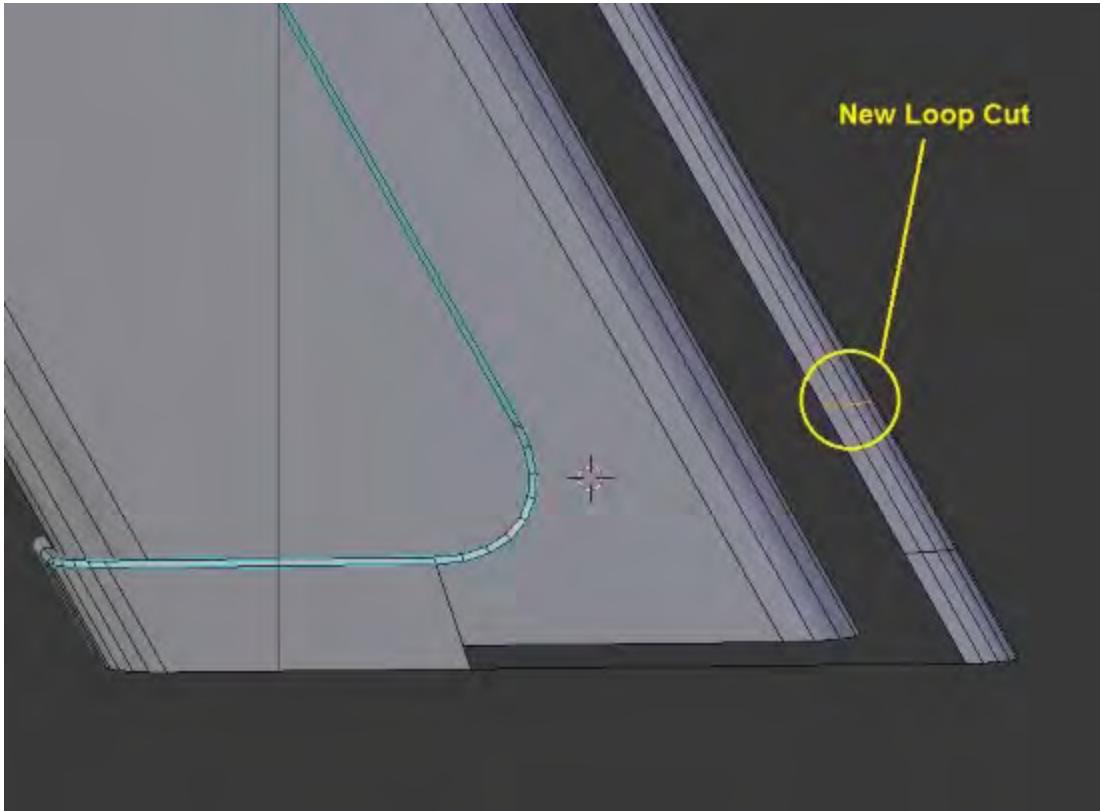
As you can see, the shading in the image is distorted. This is what can happen when you have a single vertex in the middle of a smooth (curved) surface. We don't want the vertex here, so let's fix it.

<pagebreak></pagebreak>

First, we'll just delete the vertex itself, which will eliminate the surrounding faces:

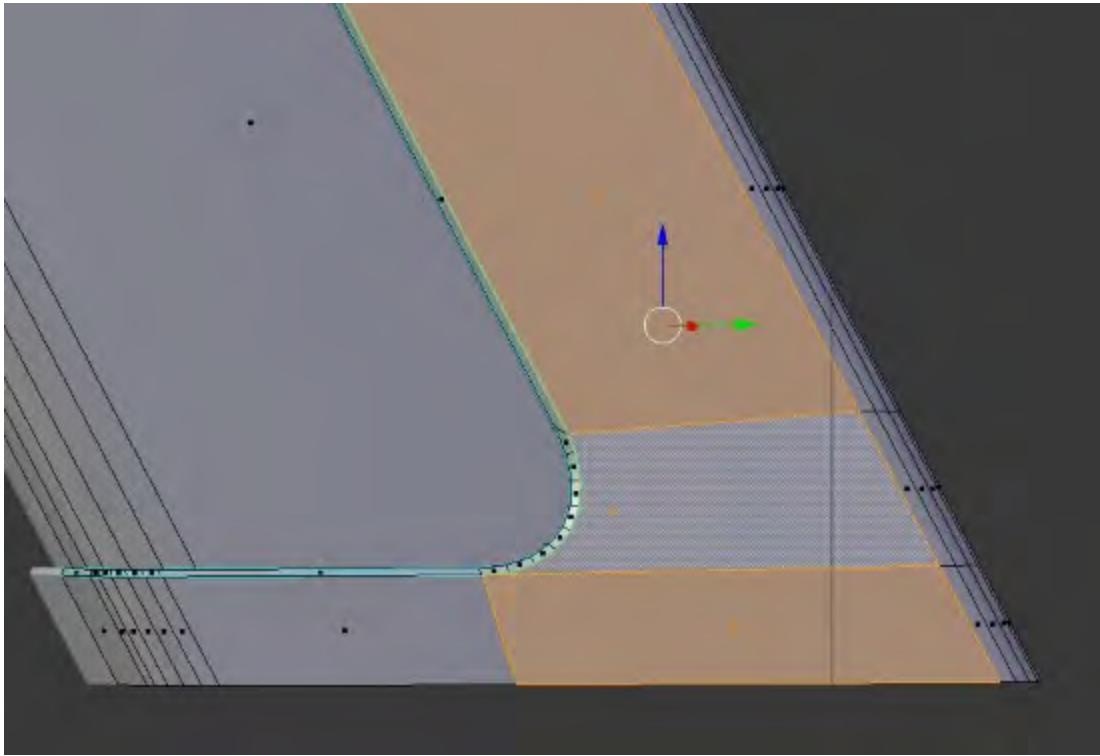


Then, we'll run a loop cut across the back of the handle:



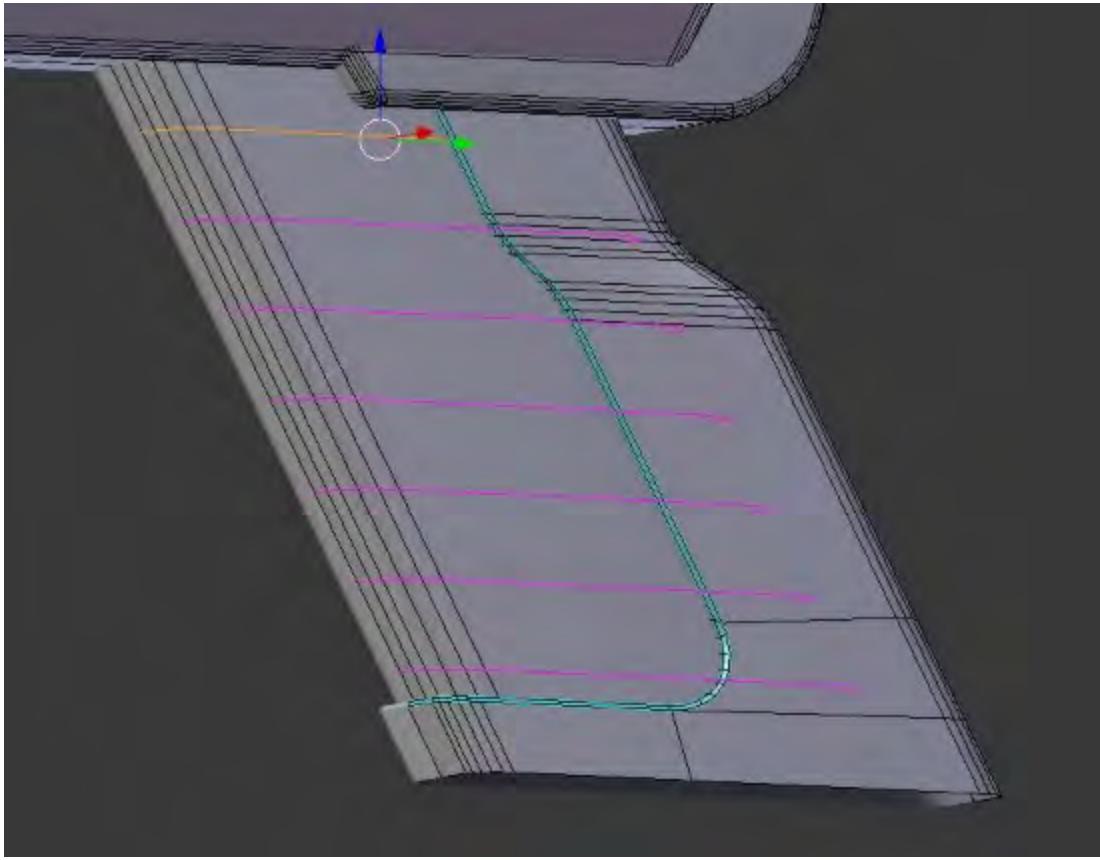
<pagebreak></pagebreak>

Next, we'll fill in some geometry:



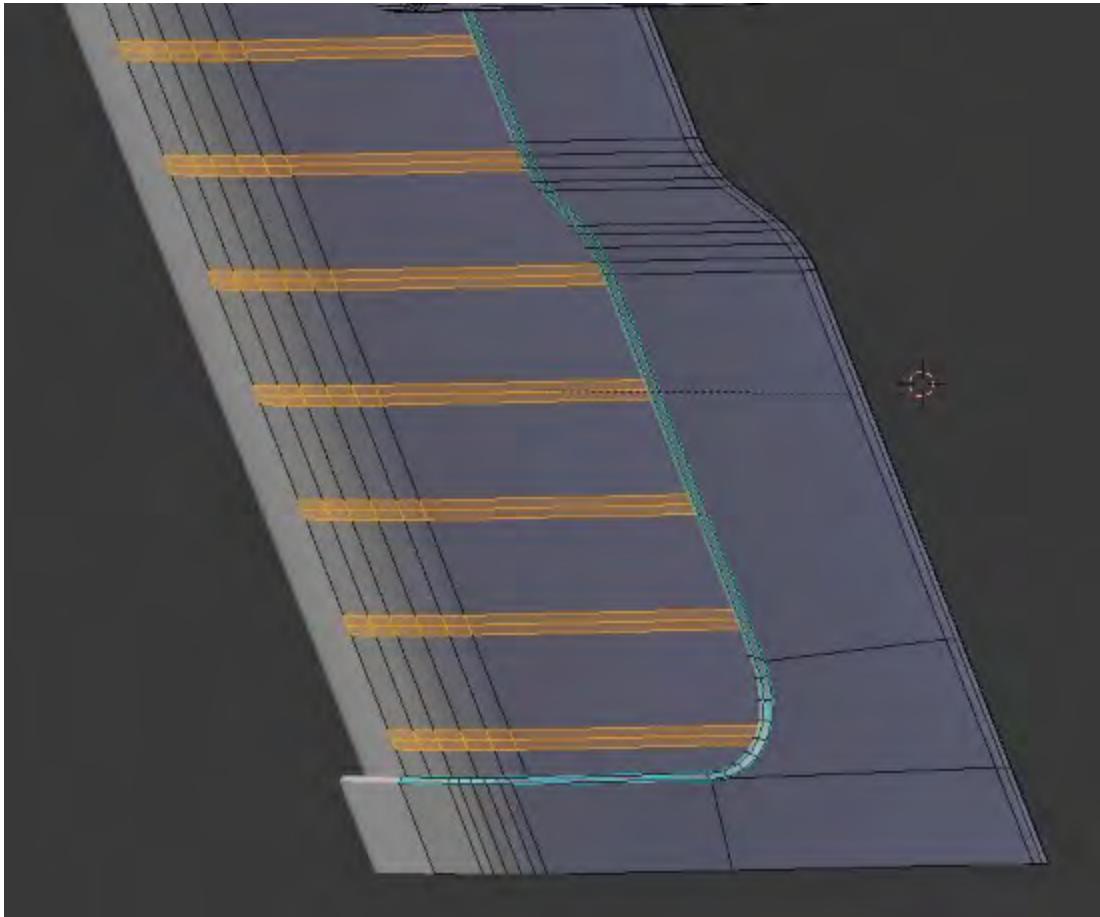
This will take care of our shading problem.

Next, let's run a few loop cuts along the pistol grip itself:

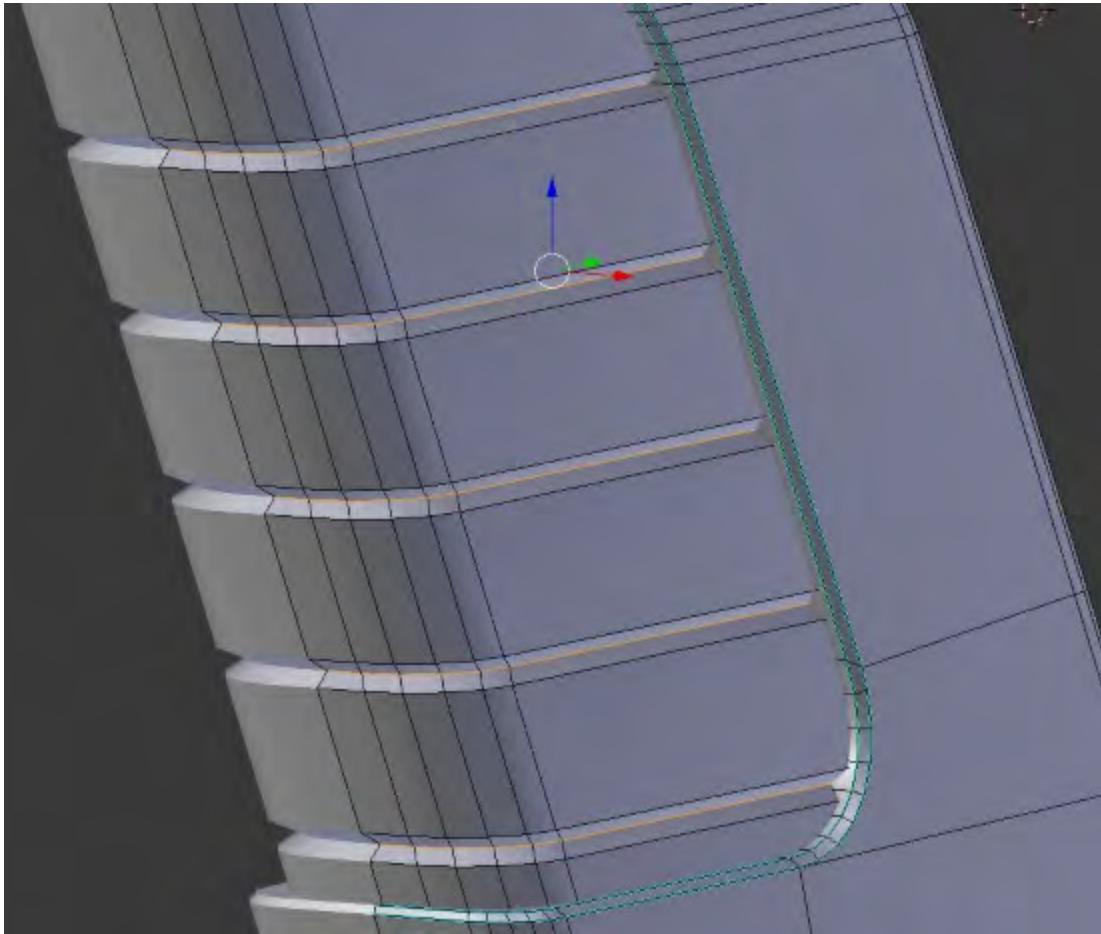


<pagebreak></pagebreak>

We can bevel these a little, rolling the mouse wheel as we do so to create a series of faces:

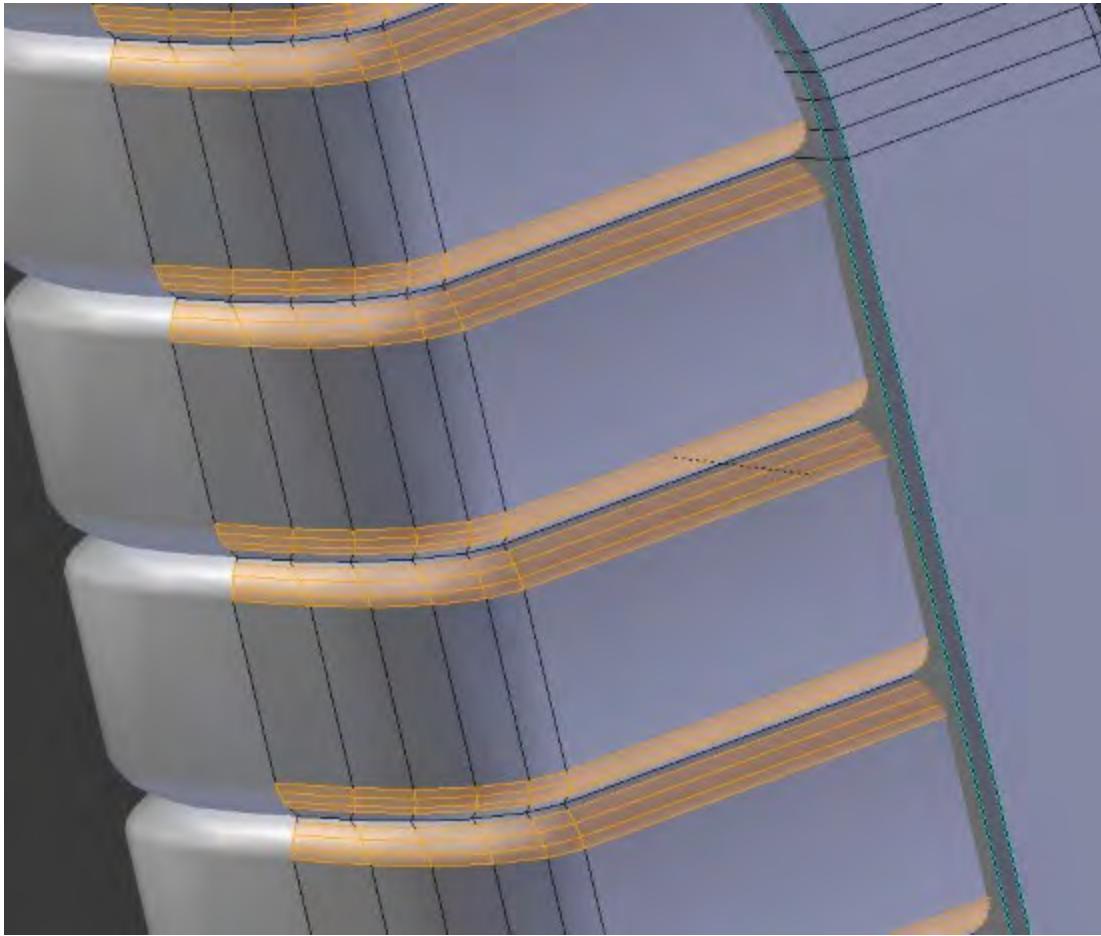


Next, we'll want to bring these middle edge loops a little bit inside. The easiest way to do this is to scale along the face normals. This is accomplished by pressing Alt + S and scaling them in.



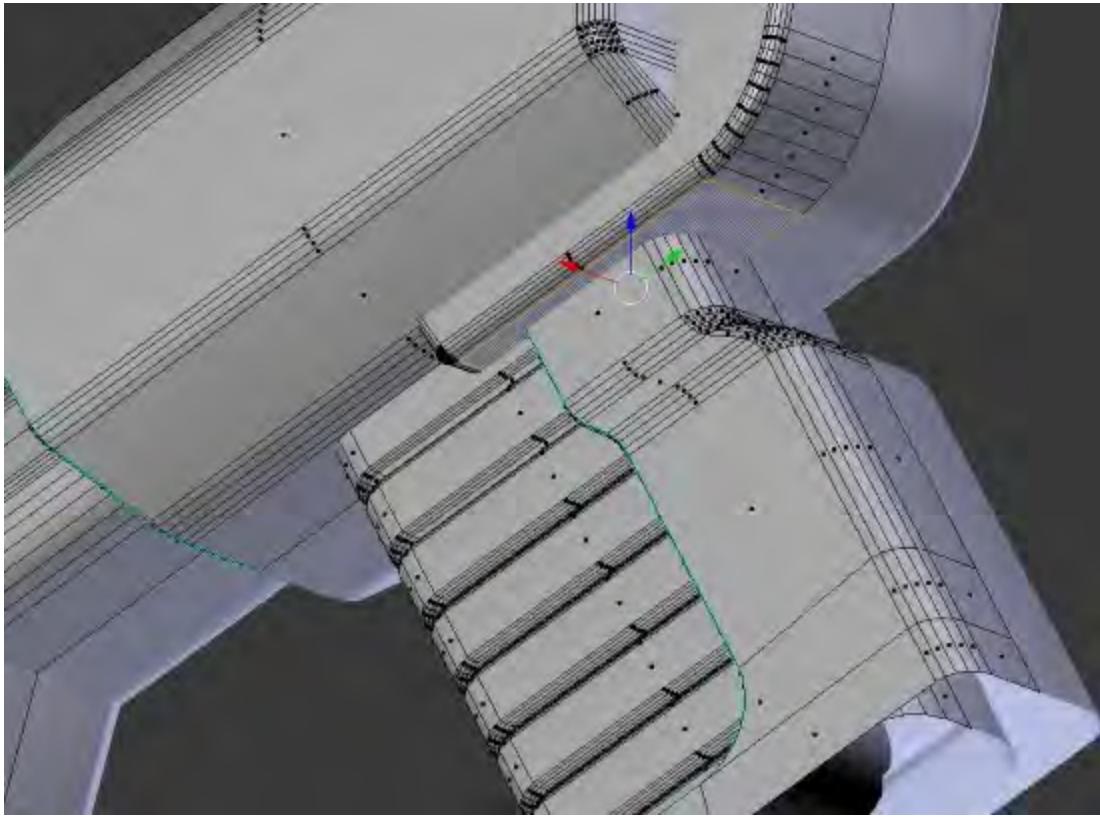
<pagebreak></pagebreak>

Now, we can bevel the edges a bit more to round things off:



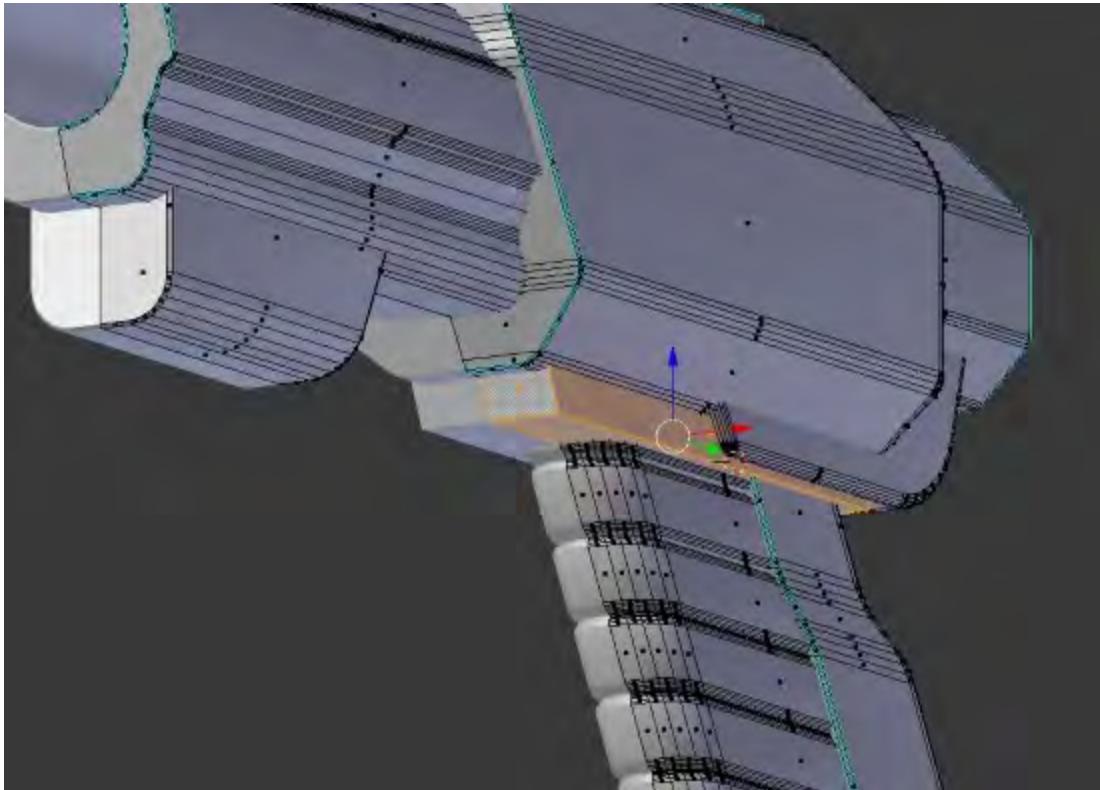
This looks pretty good.

Next, let's just duplicate an existing face with Shift + D.

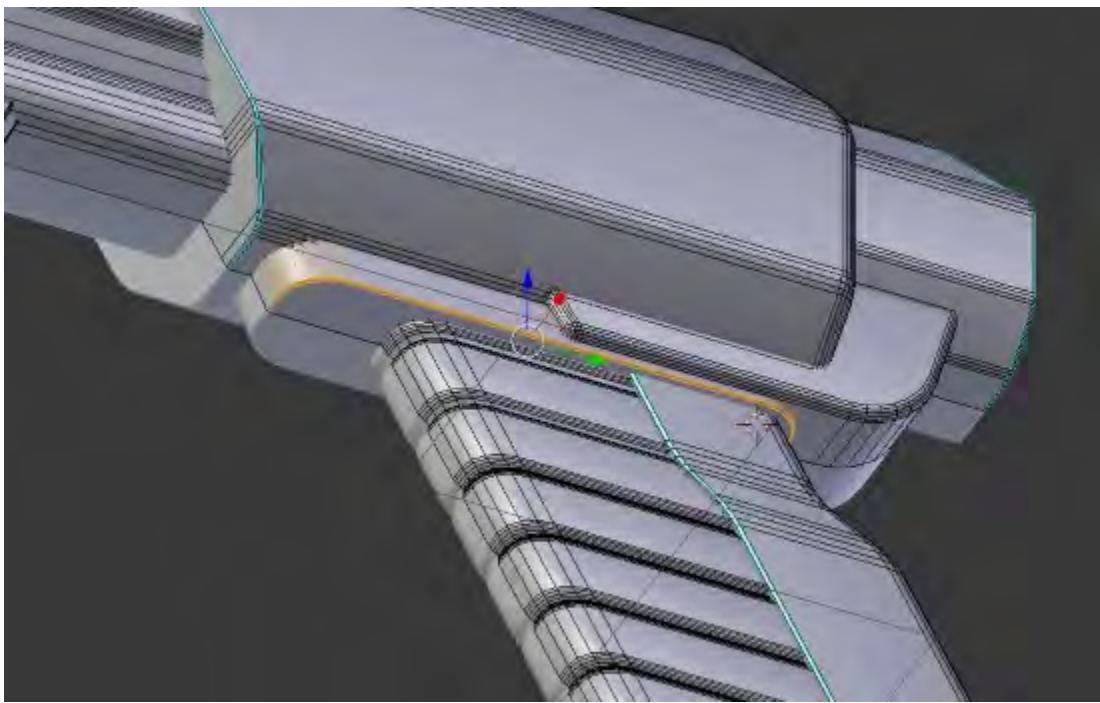


<pagebreak></pagebreak>

We can move and extrude it up to create a base for our pistol grip. This will just add a little more detail to the area where it meets the main body of the gun:

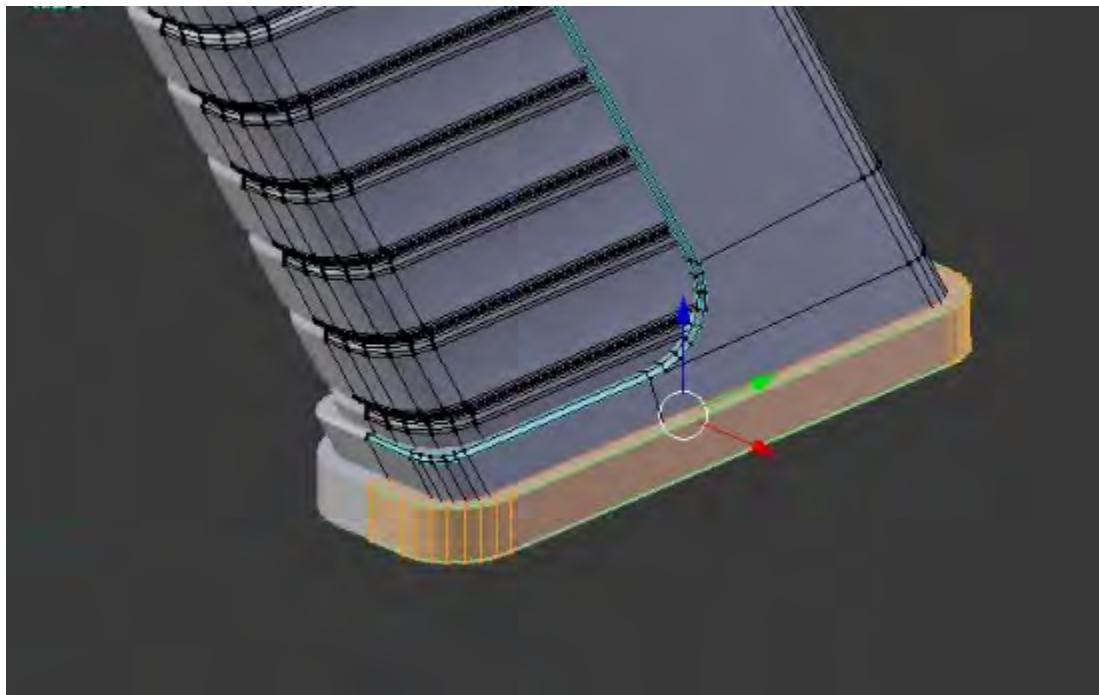


We'll bevel it until we get a nice shape using the same techniques that we've already used:

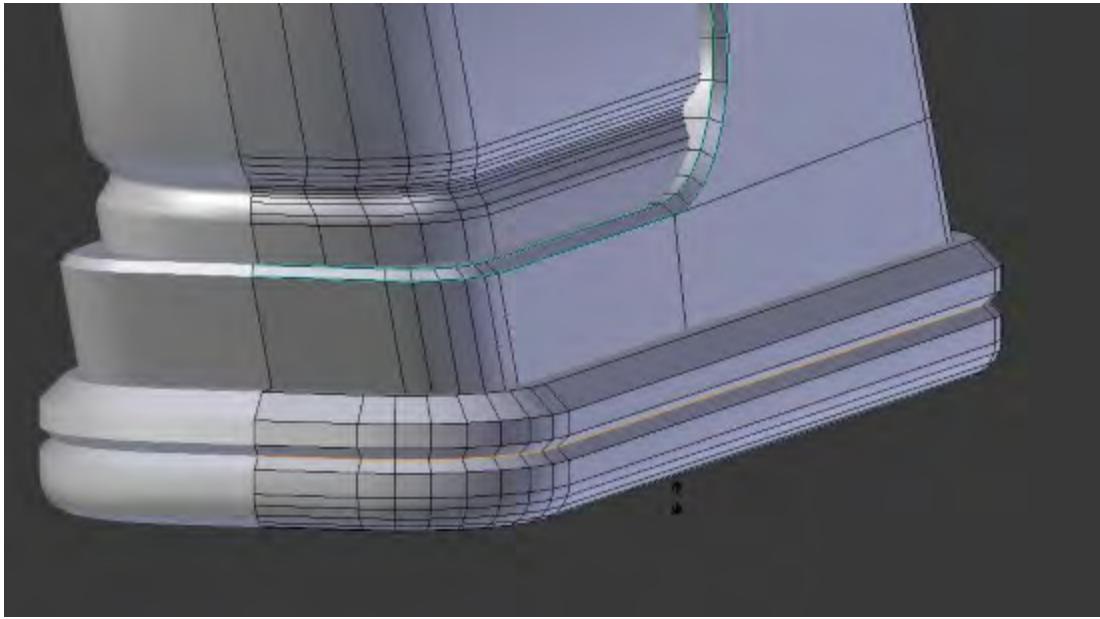


<pagebreak></pagebreak>

Then, we can duplicate this section (or just the bottom face if you'd prefer) and move it down to the base of the handle:



We can detail this area a bit using techniques that we've already covered.

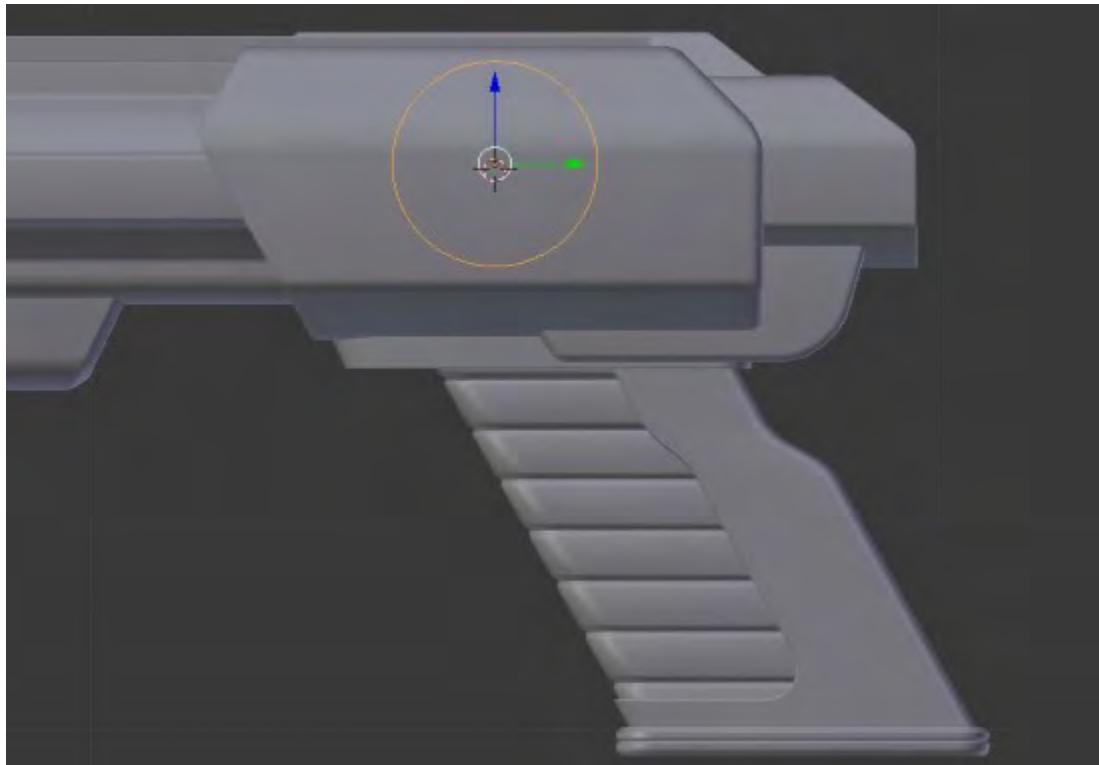


At least for now, this is pretty good for the handle. We'll add a bit more detail later, but now, it's time to move onto the main body of the gun.

<pagebreak></pagebreak>

Cutting shapes into our gun

To add detail (and enhance realism), we'll want to cut various shapes into the side of our gun. First, we want to create a circular cut-out in the middle of the body. Let's start by adding a circle in the **Object** model and lining it up where we'd like the cut to go:



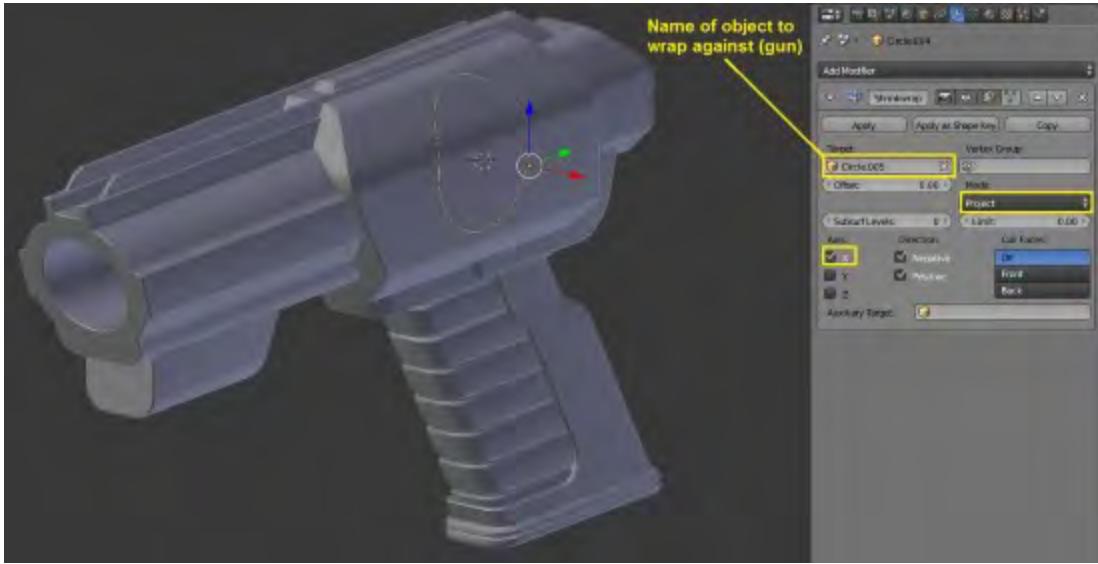
Next, we're going to add a **Shrinkwrap** modifier to the circle. This is an incredibly powerful modifier to create cuts or recesses in mechanical parts.

We need to first select the name of our gun object in the **Target** box. Then, we'll need to set the mode to **Project**. Finally, we'll need to specify the axis that we'd like it to project on. In this case, it will be the **X** axis.

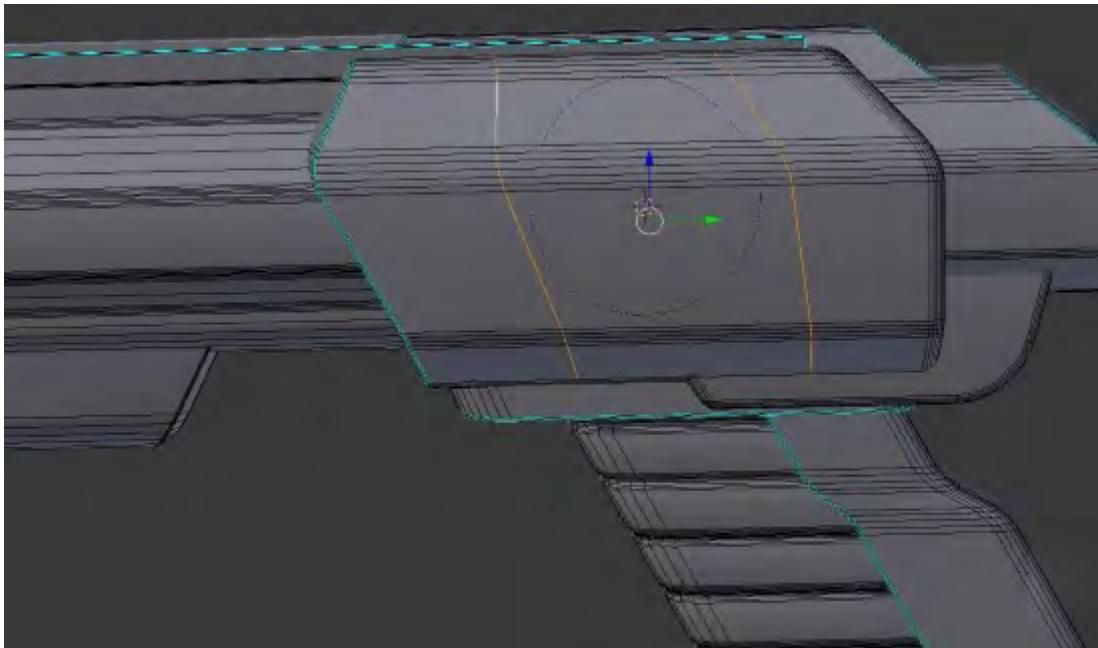
Once these settings are selected, you can drag the circle up against the gun, and it will conform to this shape.

Note

The more geometry (vertices/edges/faces) an object has, the better it will be able to conform to the shape of something else.

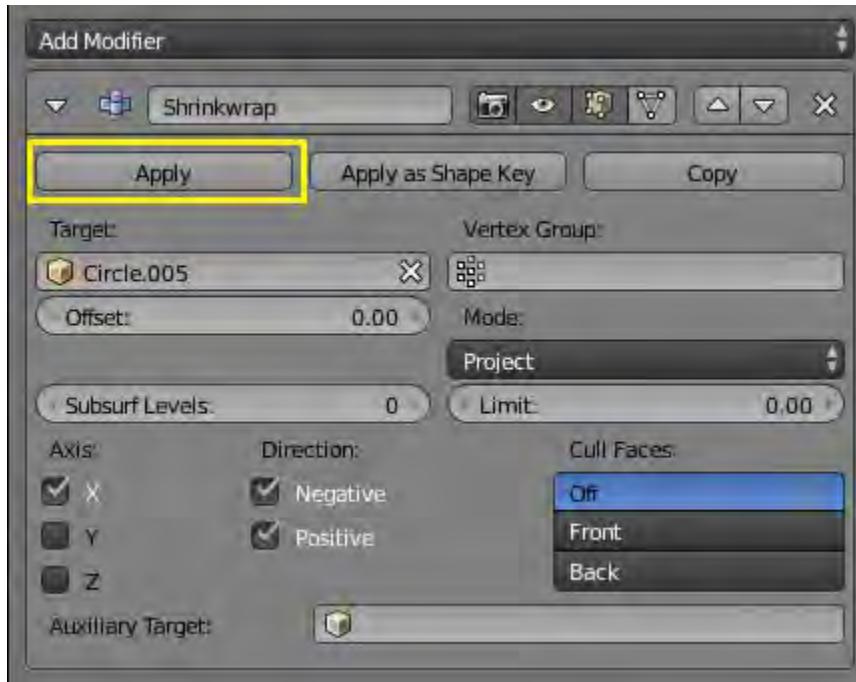


Let's pick our gun object again, and Tab back into **Edit mode**. We want to run a couple of loop cuts along the main body section. This will make it easier to fill in the shapes later:

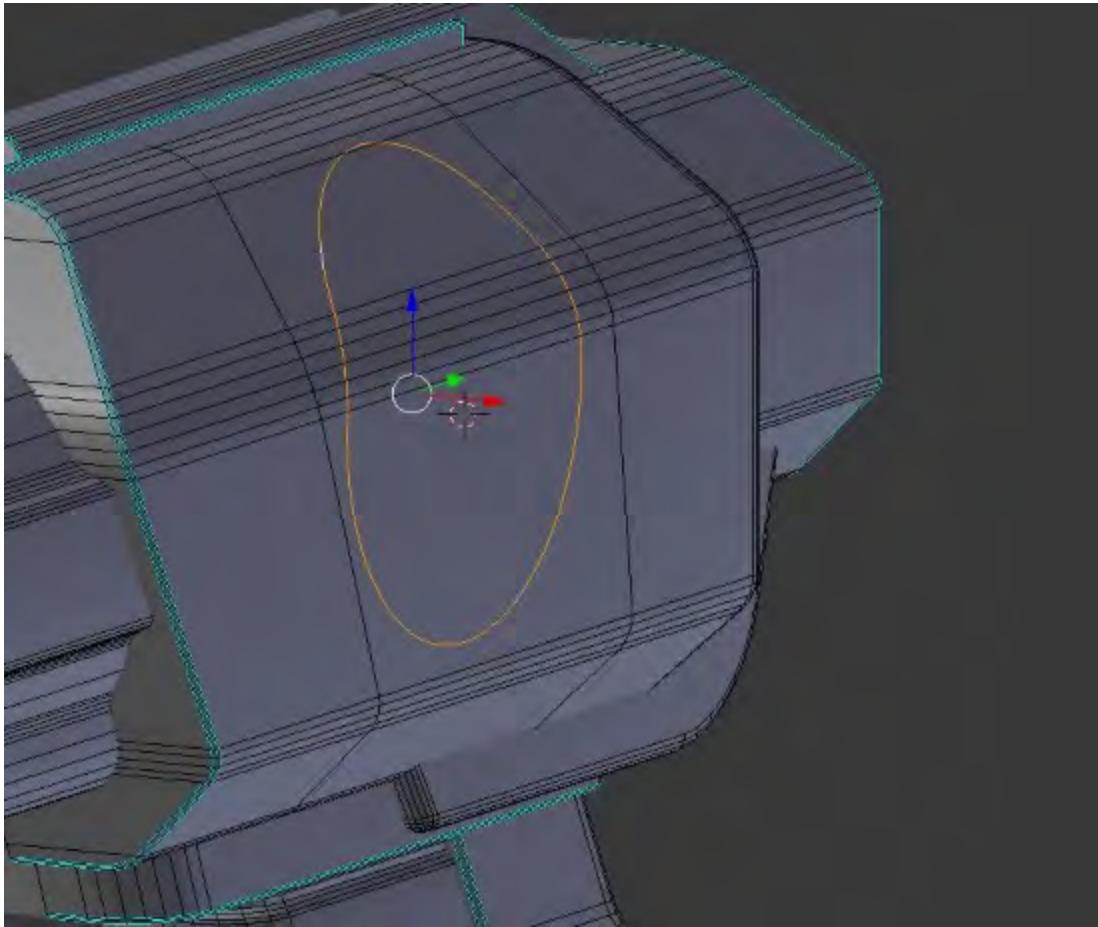


<pagebreak></pagebreak>

Once we've done this, let's go ahead and apply the **Shrinkwrap** modifier to our circle. This will permanently change its shape and conform to the gun:

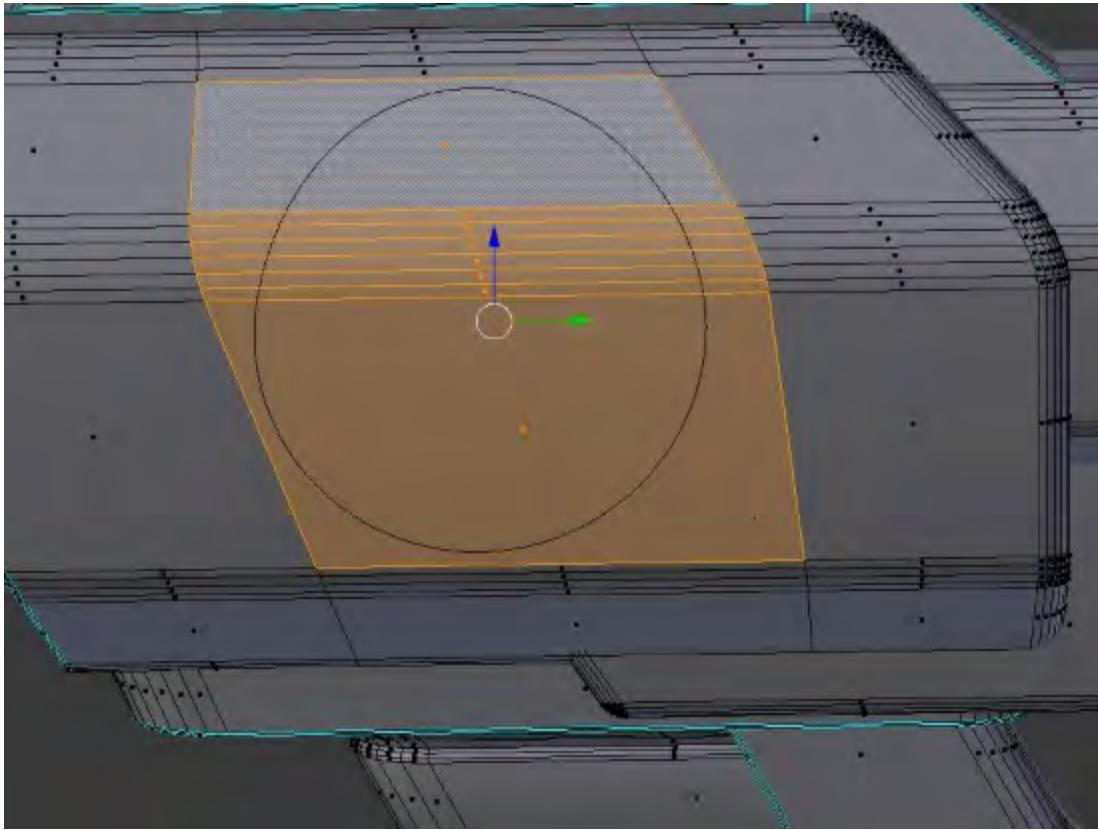


Next, we'll join the circle object to our gun by pressing **Ctrl + J**.



<pagebreak></pagebreak>

Let's delete the back faces here, so we can fill in our mesh:

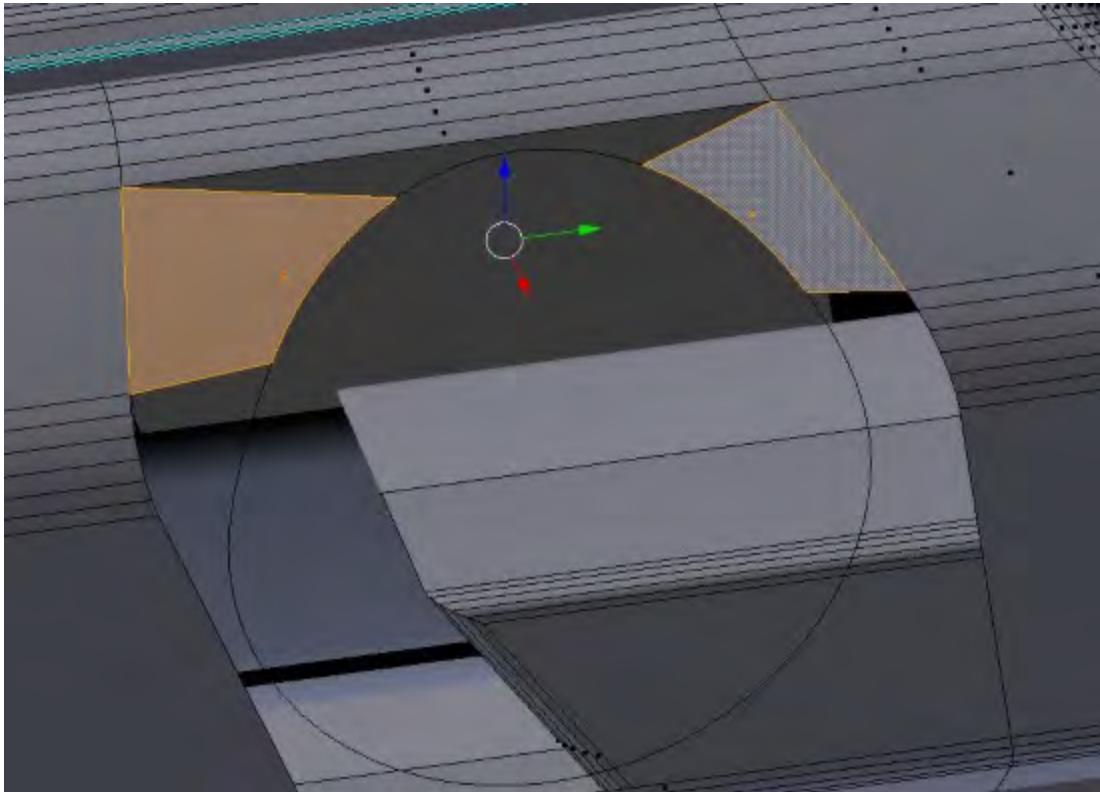


This next part is a bit tricky. We'll need to fill in the spaces between the circle and the rest of our mesh.

I should mention that cutting shapes into curved objects is one of the hardest skills to master. There are several tools that can help you (such as Knife Project, which we'll look at momentarily), but they all have their limitations. At the end of the day, it really comes down to practice.

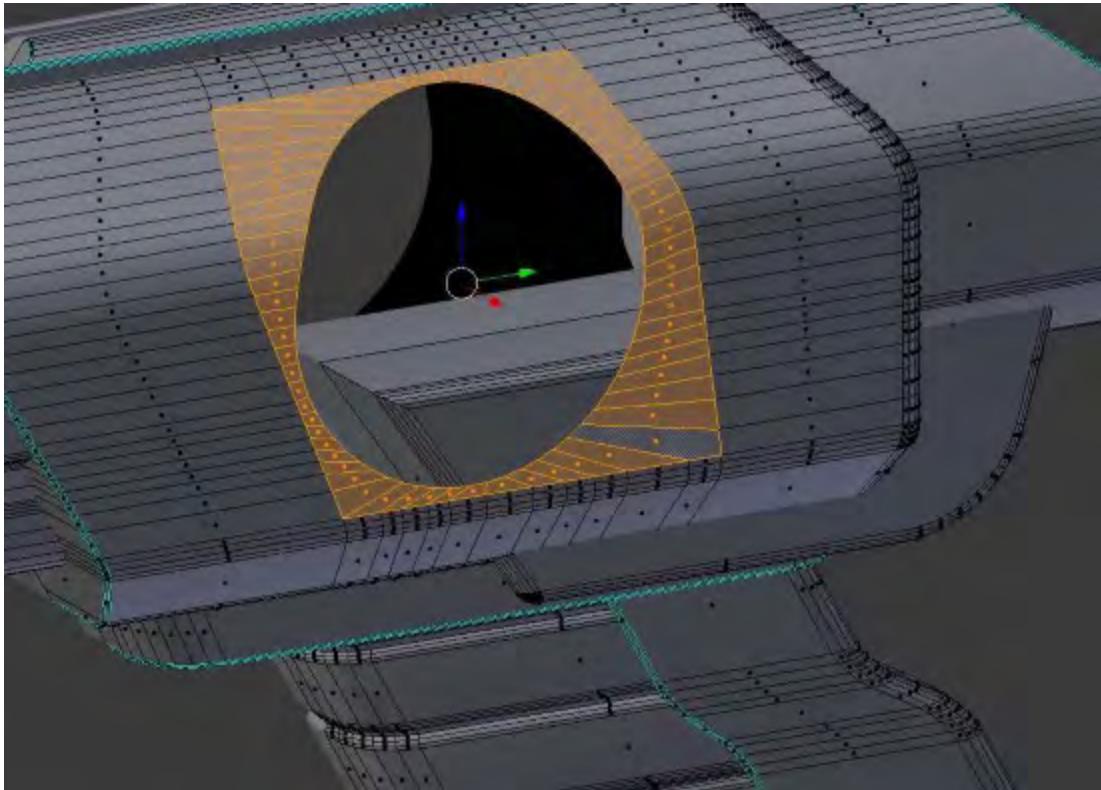
<pagebreak></pagebreak>

When filling in the curved areas of your mesh (such as this one), the general rule is this—Quads are preferable to Tris, and Tris are preferable to N-Gons. At the end of the day, it's the result that counts. If N-Gons work in a given scenario, don't avoid them just on principle.



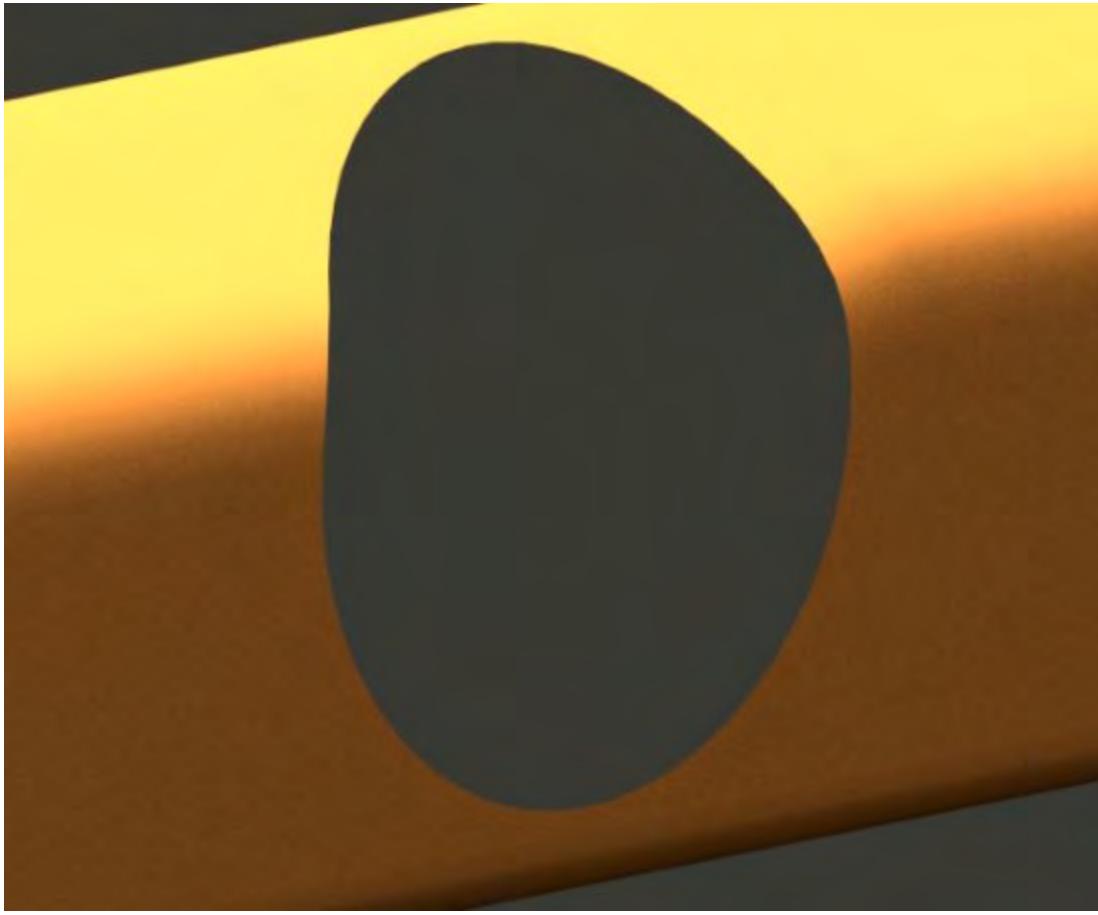
As you go through this part, you'll probably find yourself needing to add loop cuts to the existing gun body. This is fine; go ahead and do it. More geometry tends to give you better definition, so don't be afraid to add some. As much as possible, we'd like our new geometry to follow the lines/curves of what was there before.

Here's what I ended up with:



<pagebreak></pagebreak>

It's not perfect, but this helps to illustrate the point. When we render this, it produces the following result:



This looks pretty good. There are no significant shading distortions around the curved areas, which is what we want.

Now, is it possible to smooth out the geometry so that it looks more appealing? Of course, but it's not necessary for rendering. A clean geometry generally produces better renders... but remember, the result is the ultimate determination of what is (and is not) a "good" mesh.

Before we move on to the next part of our project, it's worth taking a moment to examine the **Knife Project** tool. Its purpose is to cut shapes into existing objects; some readers will undoubtedly wonder why we didn't use it here.

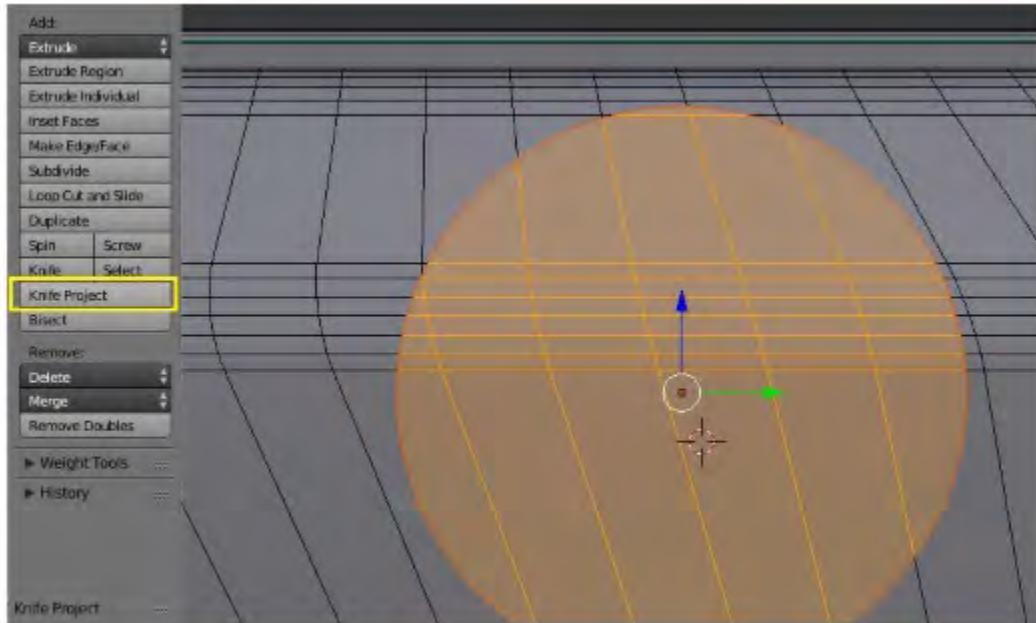
Let's take a look.

To use the **Knife Project** tool, you simply need to add a circle object and line it up with where you wanted your cut to go.

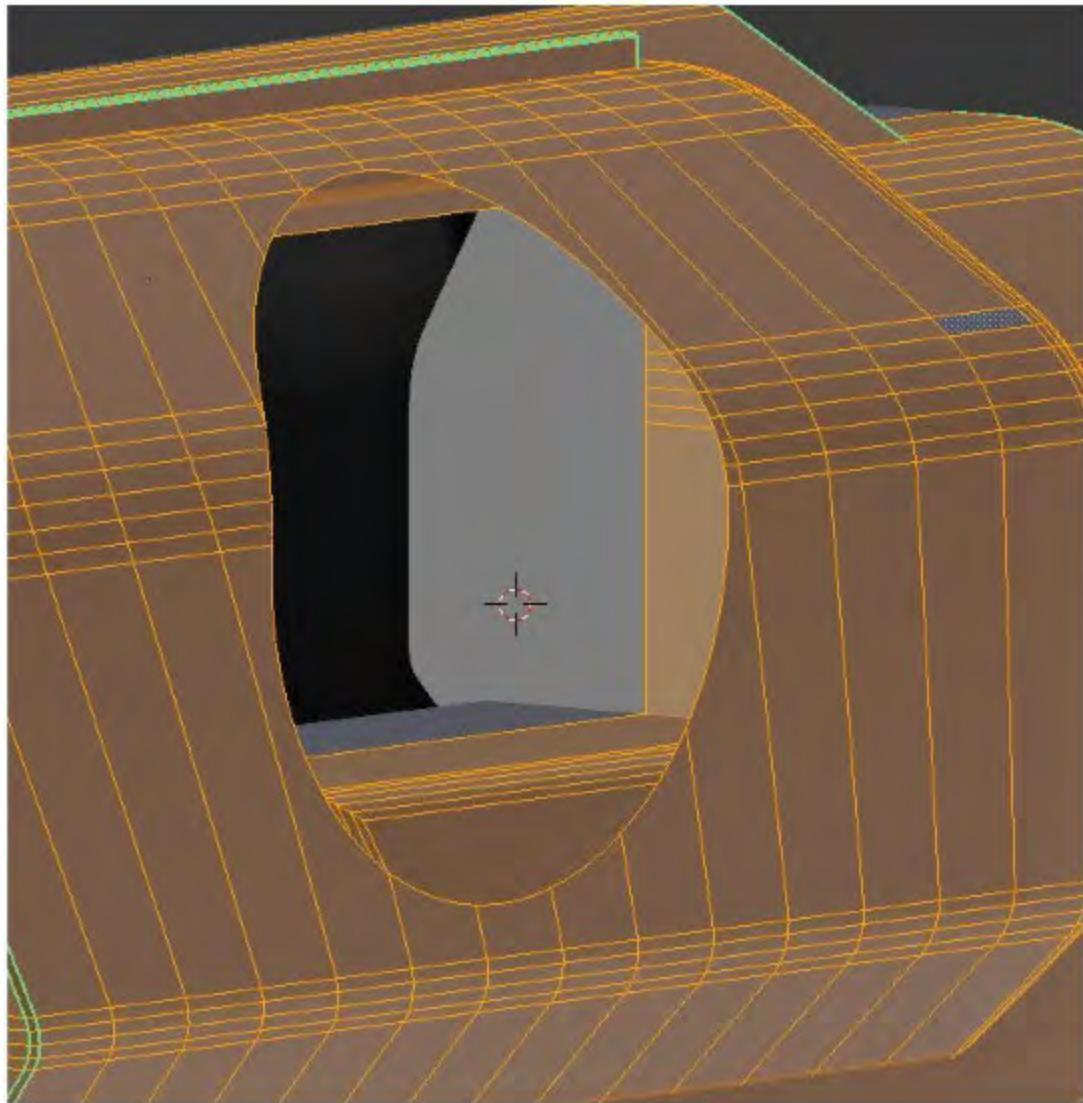
<pagebreak></pagebreak>

Then, (in **Object** mode), you need to select your circle (the object that will do the cutting). Then, select your gun (the object that will be cut). Tab into **Edit mode**, and then press **Knife Project** under your **Mesh Tools** tab.

Here's what that will look like in our case:



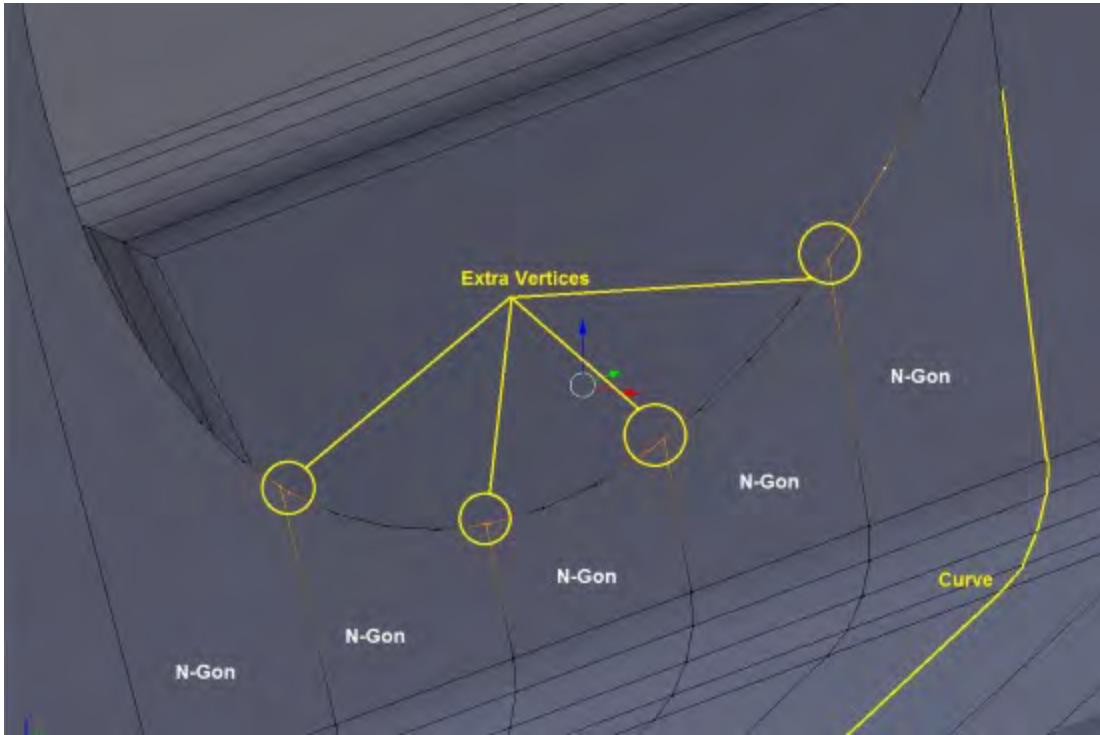
Once we've removed the faces, we'll be left with this:



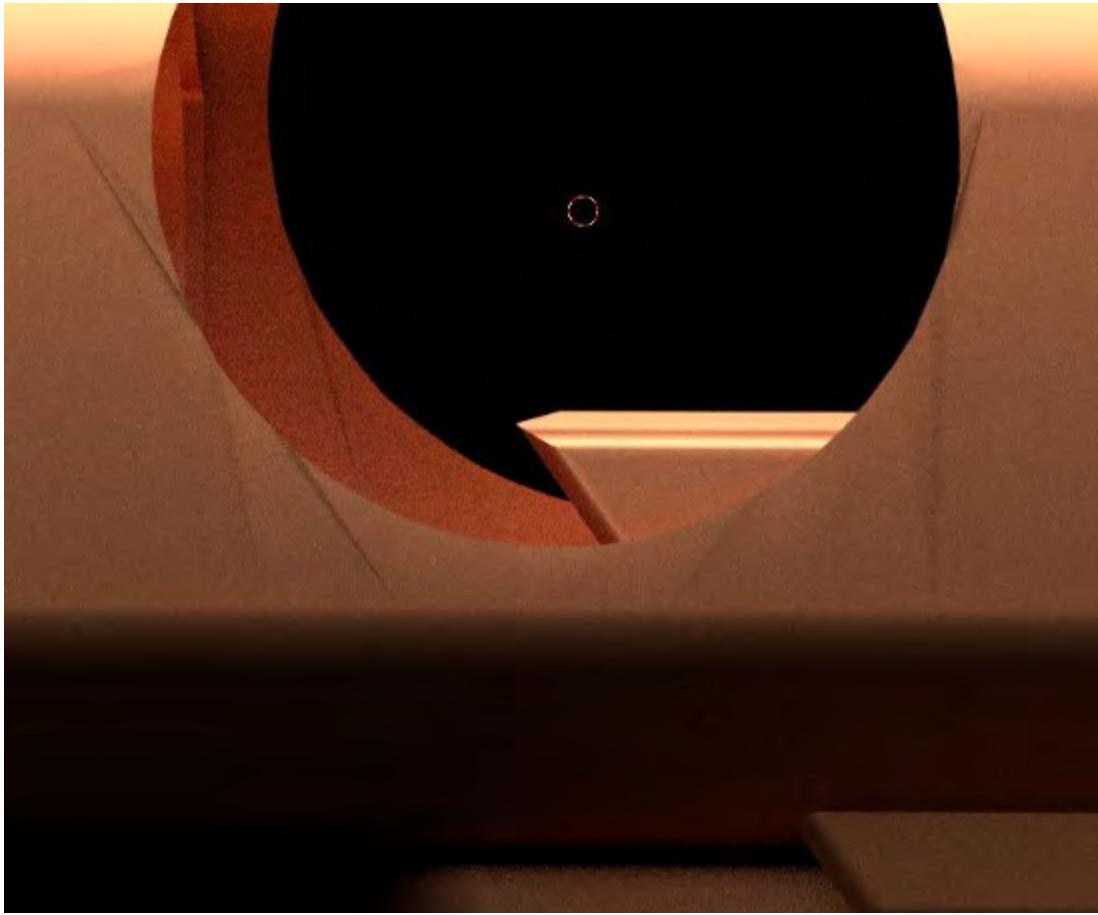
It looks pretty good, doesn't it? Nice and clean. So why don't we use it?

<pagebreak></pagebreak>

The problem is that **Knife Project** keeps the vertices of both objects. As you can see here, we've created a number of N-Gons along a curved surface and each of them have many vertices in close proximity to each other.



As we've seen before, this can have an adverse effect on shading.

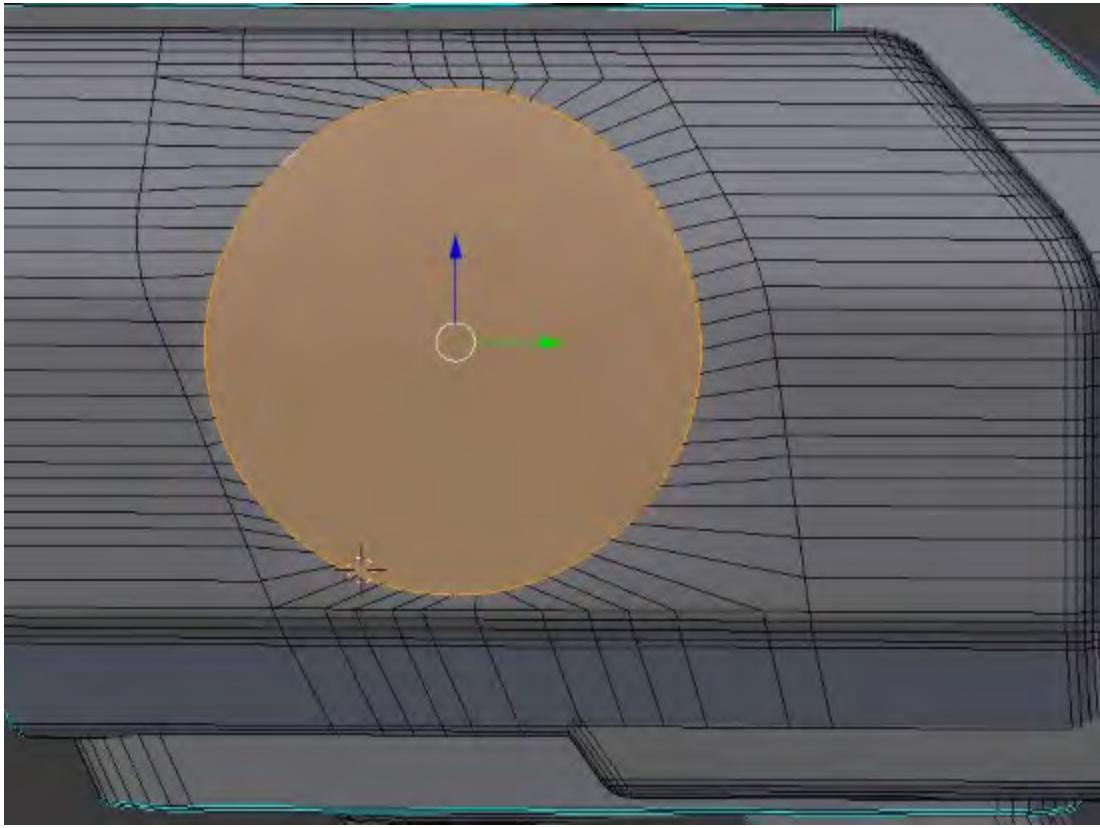


You can see here that extra vertices create a few shading problems. Is it possible to fix these problems? Sure! You can join (merge) the vertices together and move them around until the shading looks good.

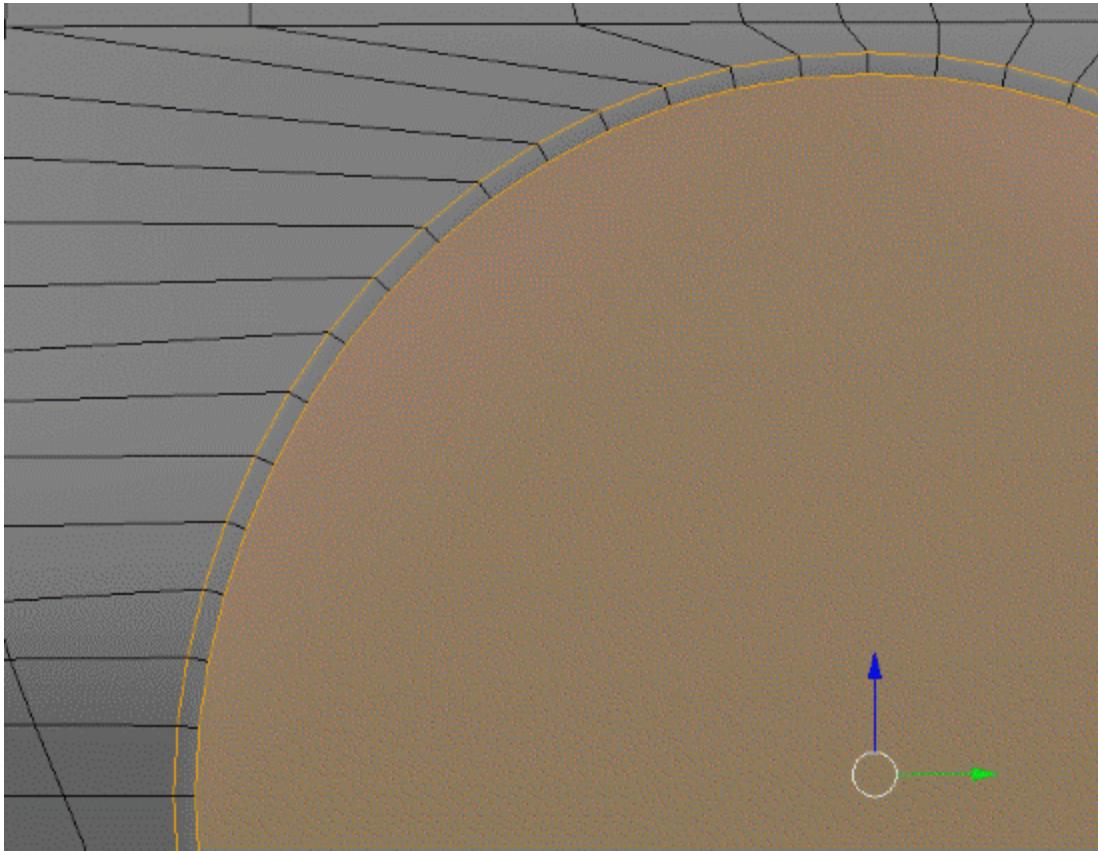
<pagebreak></pagebreak>

There's nothing wrong with **Knife Project**. Just like **Subdivision Surface** modifier (we'll cover this later), this one is also a powerful modeling tool. Unfortunately, tools like these tend to be overused (particularly by beginners). This often prevents people from learning to manipulate and optimize meshes by hand.

So now, we're back here with our gun's body. Let's create a temporary N-Gon face to fill in our new circle.



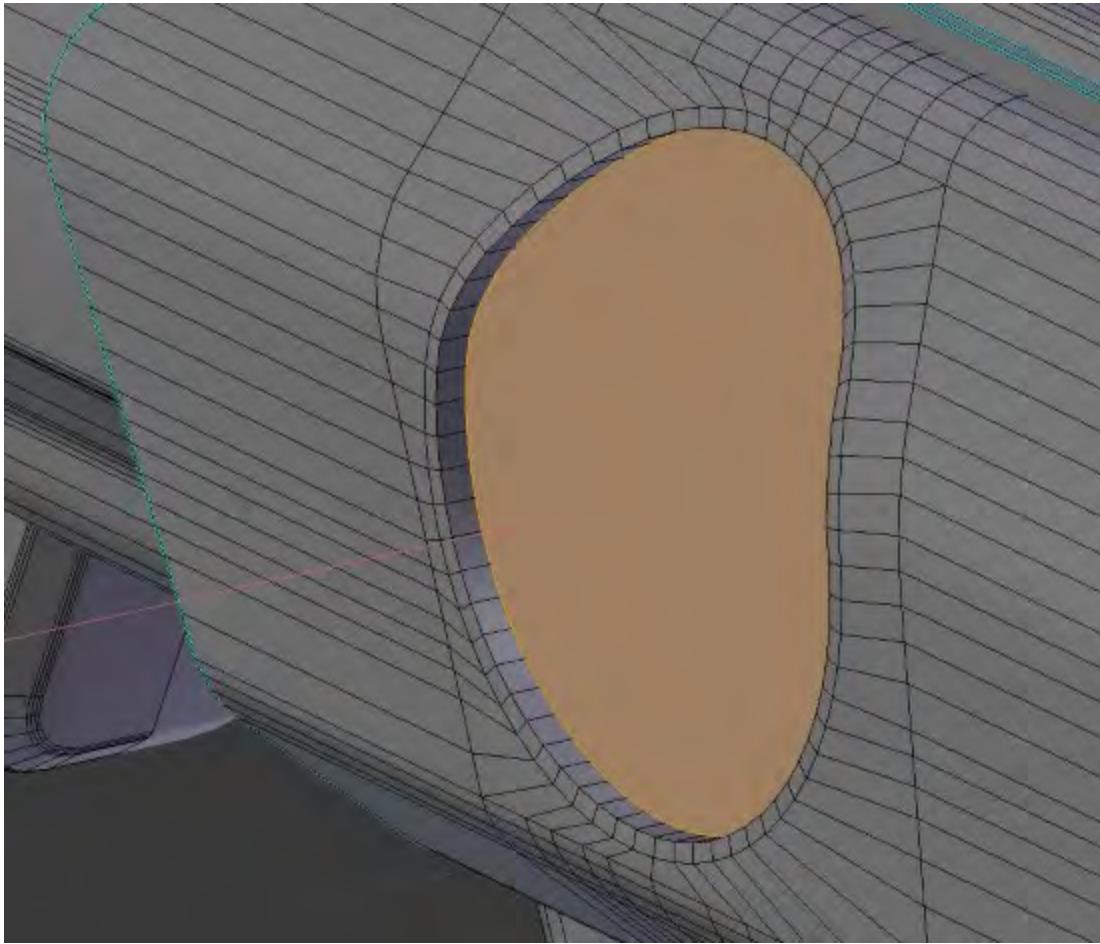
Then, we'll use **Inset** tool by pressing the I key. This will extrude a new face towards the center, leaving a nice even border.



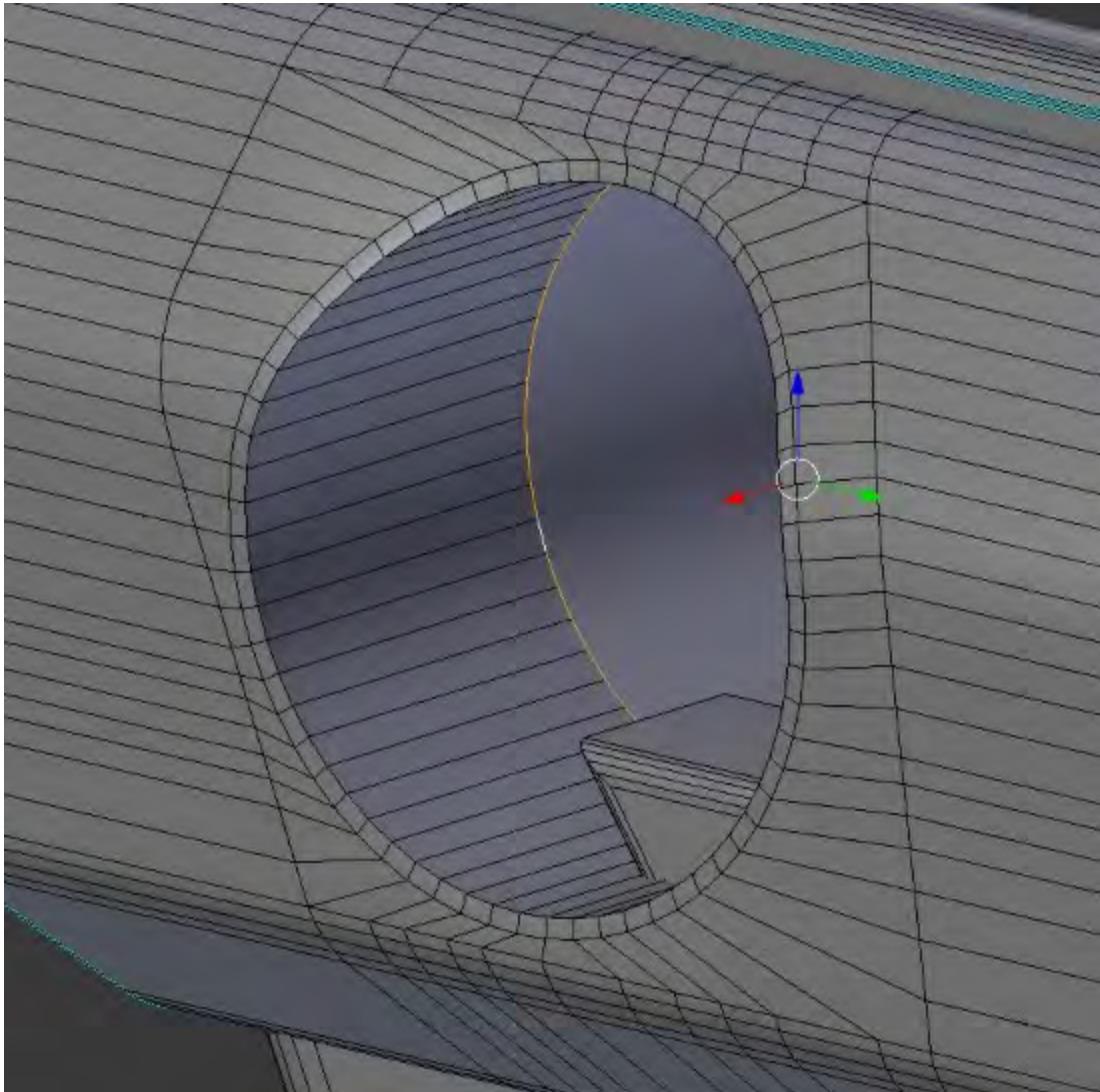
This is a key step, and it often gets overlooked. By adding the extra ring of edges (with the **Inset** tool), we preserve the shading around the edge of recessed areas.

<pagebreak></pagebreak>

Once we've got the ring, we'll again extrude the circular face towards the center of the gun.

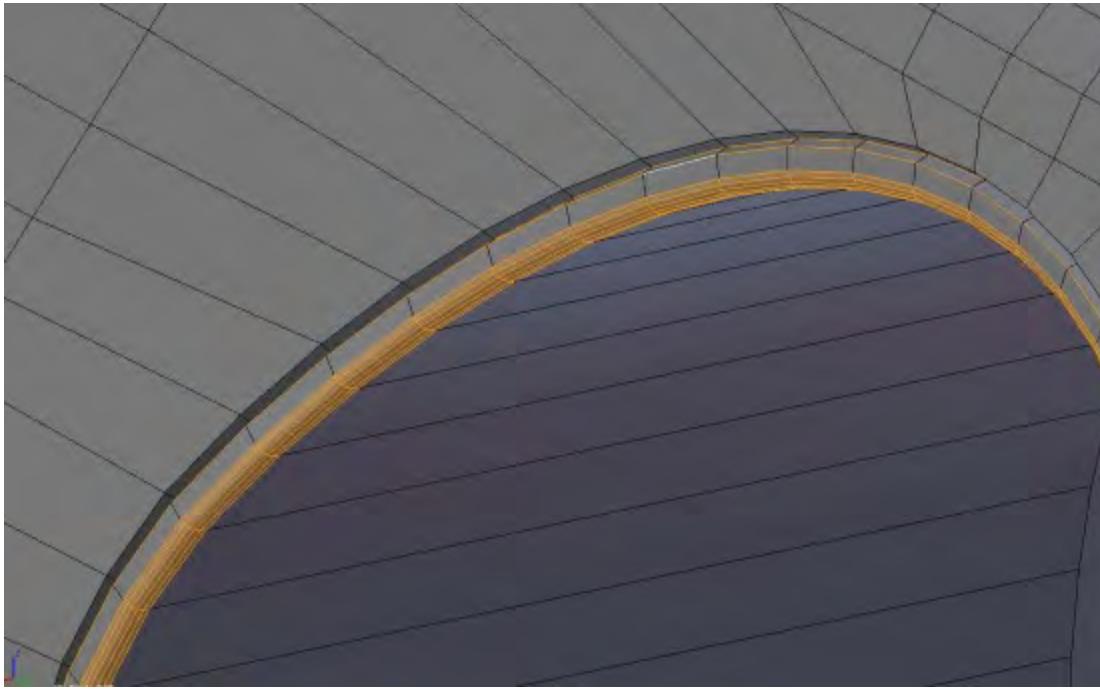


As the face hits the center point of our gun object, the **Mirror** modifier will force it to become flat. Once this happens, delete the face, and you're left with a nice, cylindrical hole through the center of the gun's body.

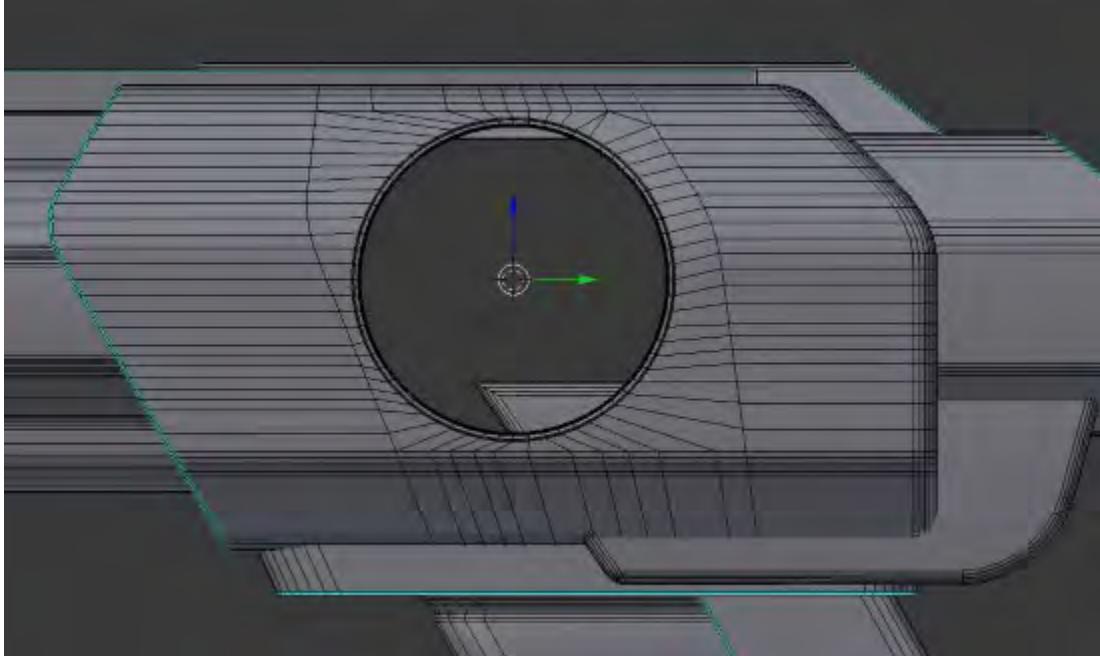


<pagebreak></pagebreak>

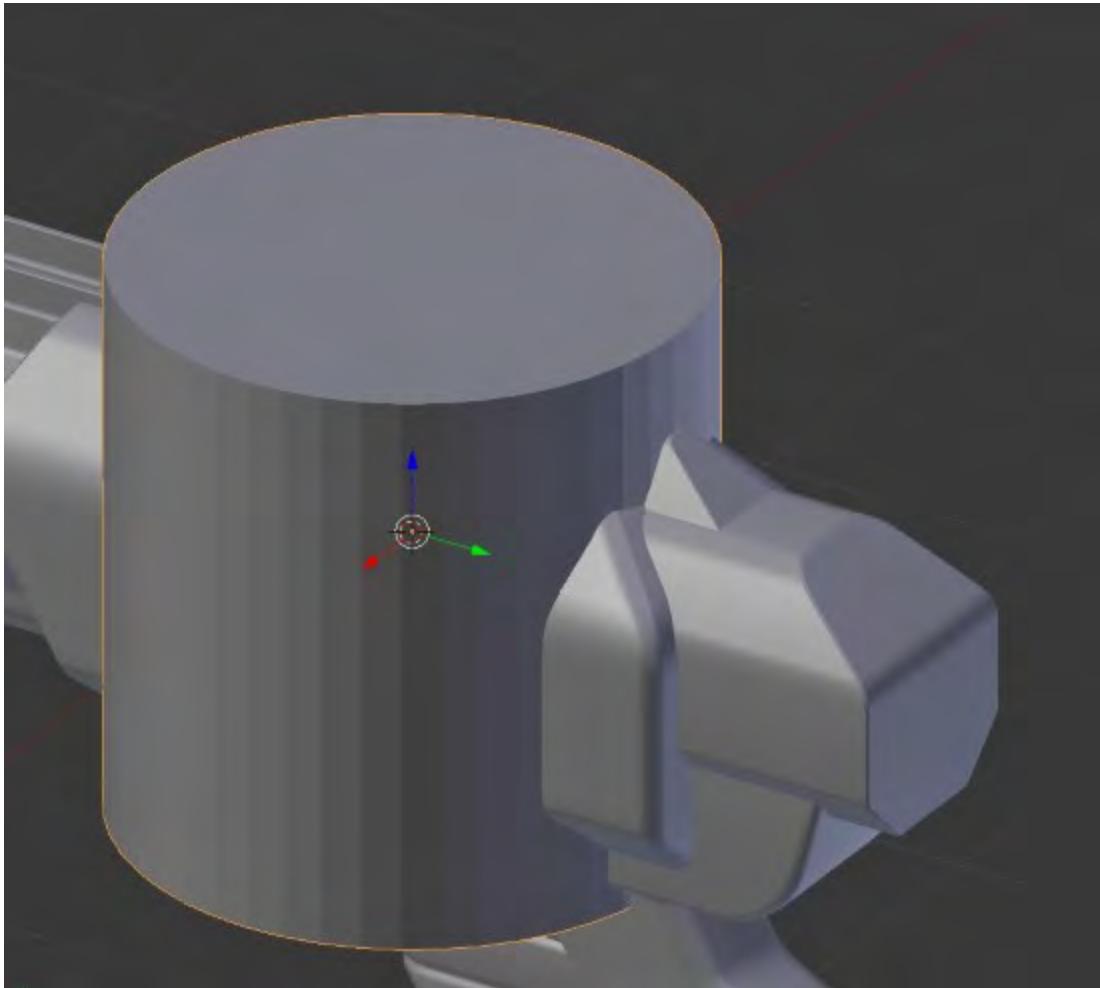
If you'd like, you can now go through and add a little detail to it by beveling the edges.



Once this is done, the gun's body will look like this:

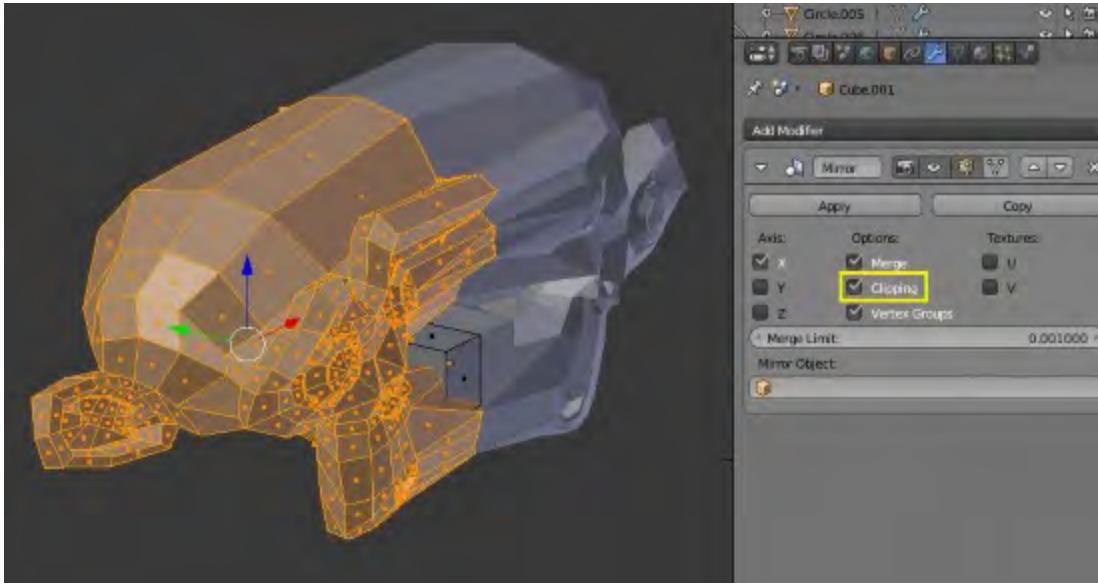


Next, we'll go ahead and add a cylinder in **Object** mode.



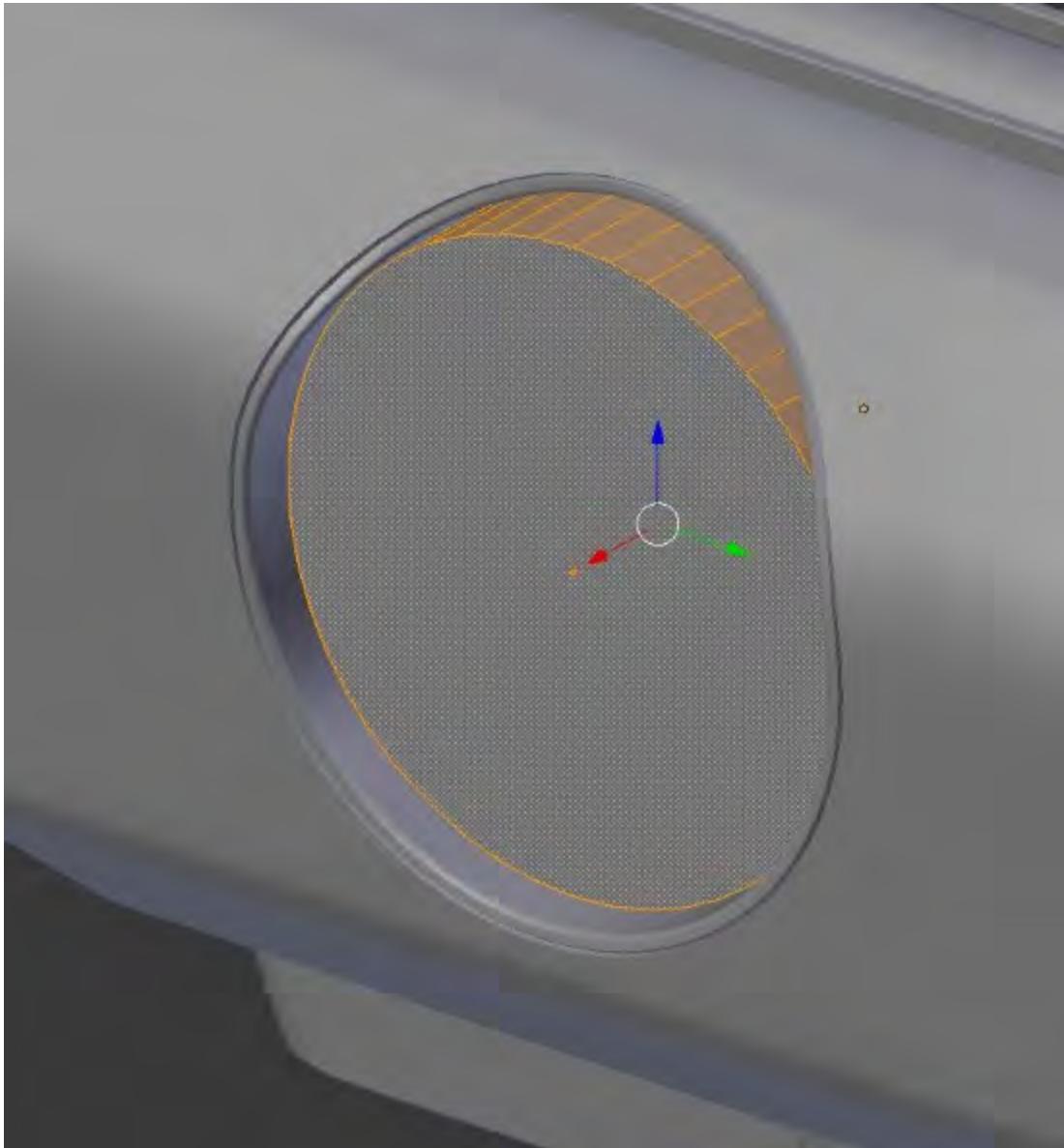
Note

Adding Parts to Mirrored Objects When using the Mirror modifier, it's often convenient to add parts in Object mode (versus adding them in Edit mode to the existing model). When Clipping is enabled on your Mirror modifier, you may accidentally distort your object if it's overlapping the center point.

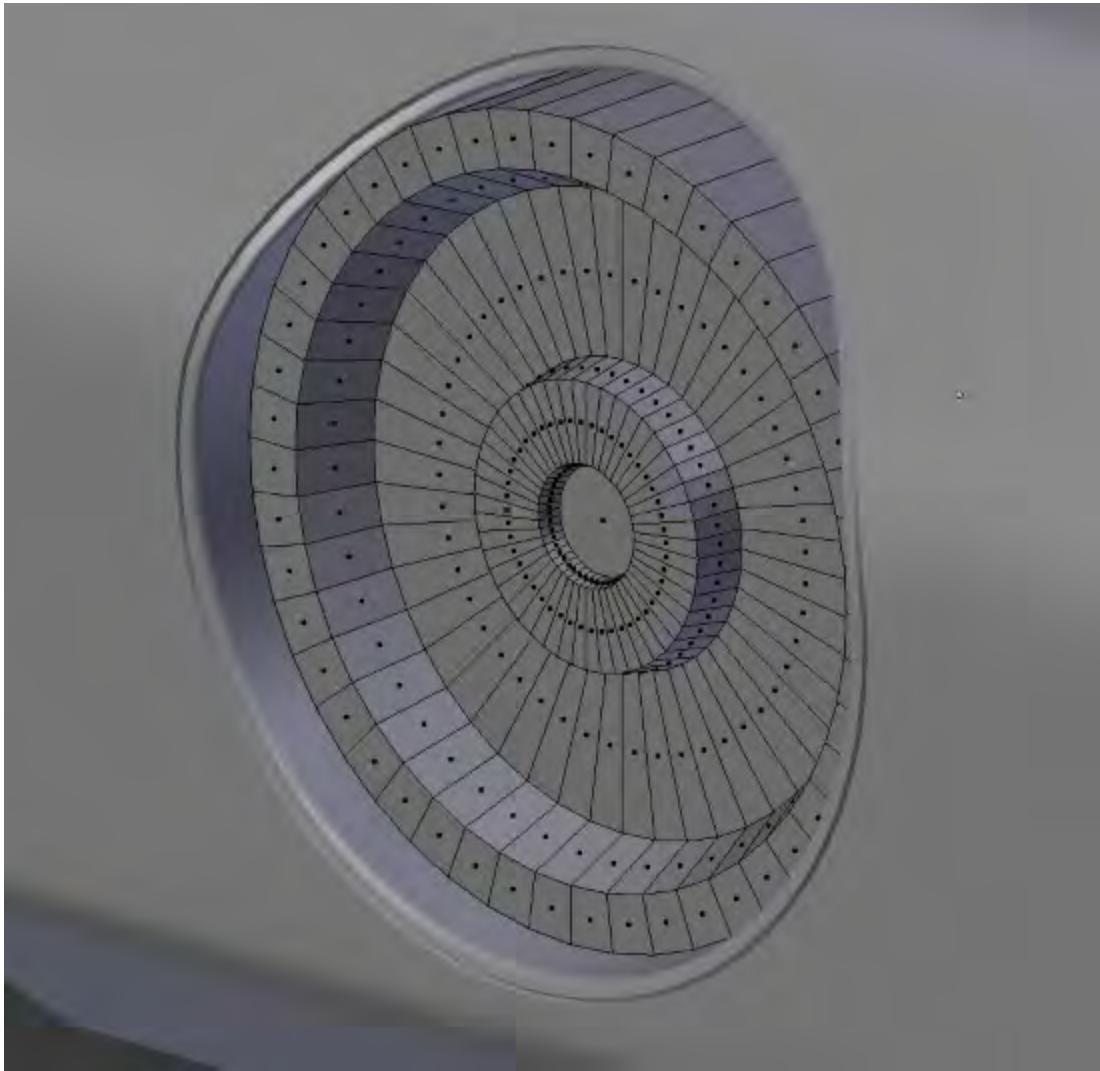


<pagebreak></pagebreak>

Next, we'll scale and move the cylinder into position:

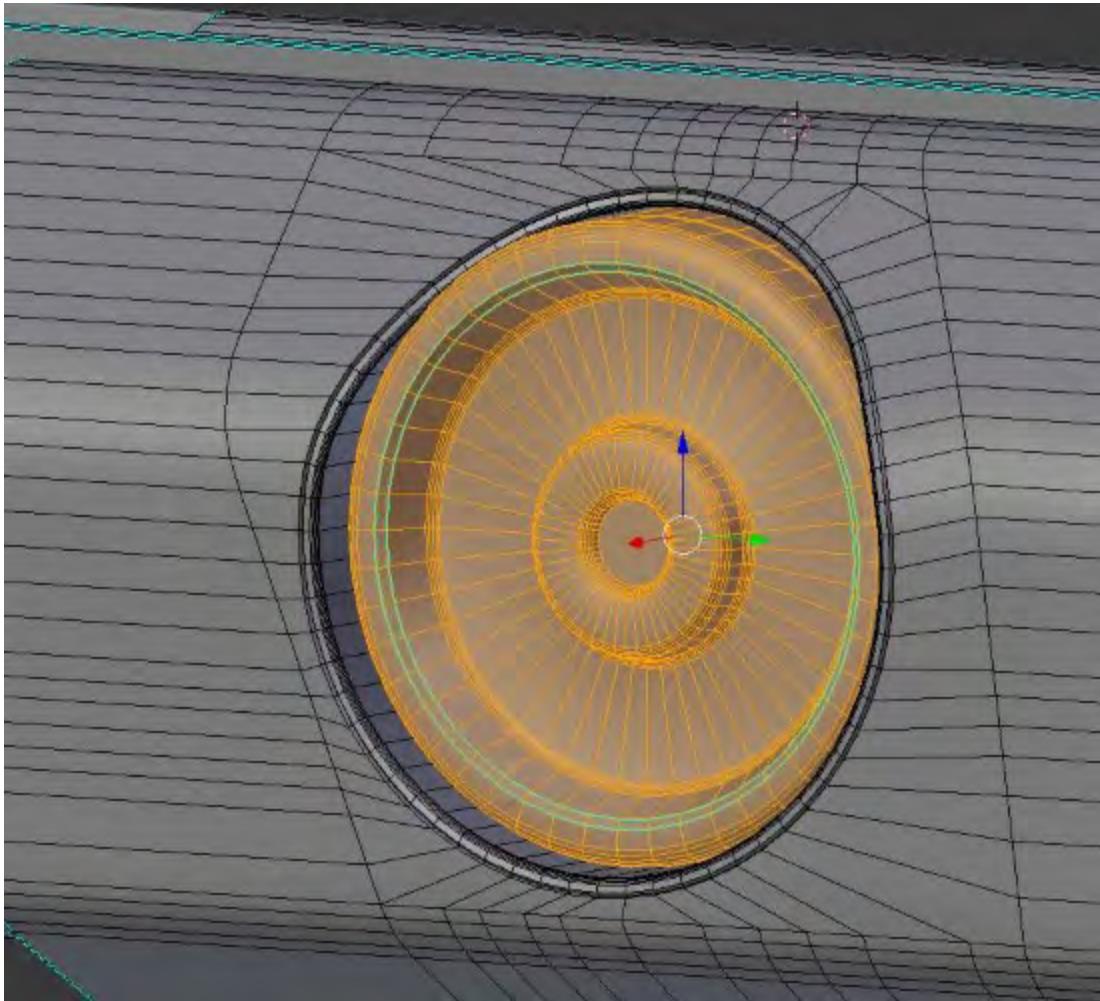


Using the **Inset** and **Inset Extrude** tools, we'll create some basic shapes:

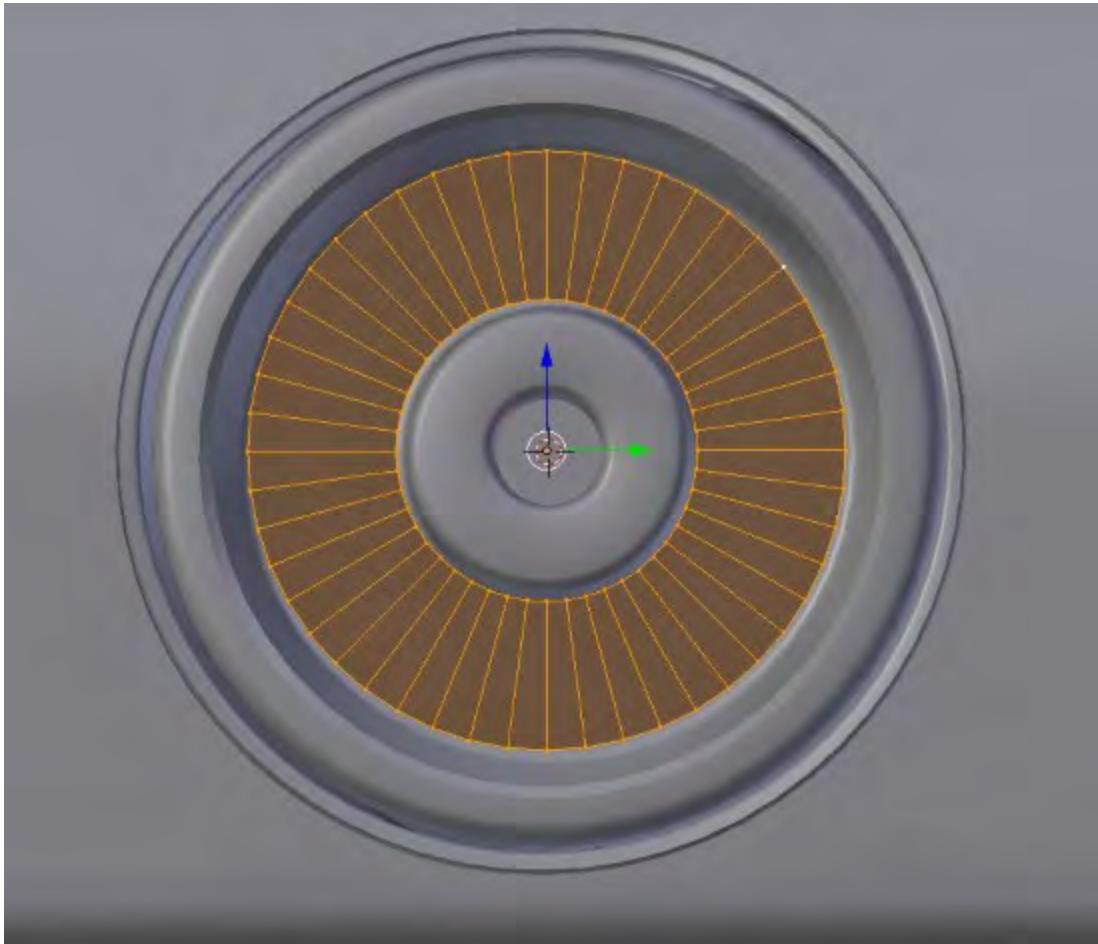


<pagebreak></pagebreak>

Then, we'll bevel our edges to add some detail. Once the cylinder's in position, you can go ahead and join it to the gun model by pressing Ctrl + J.

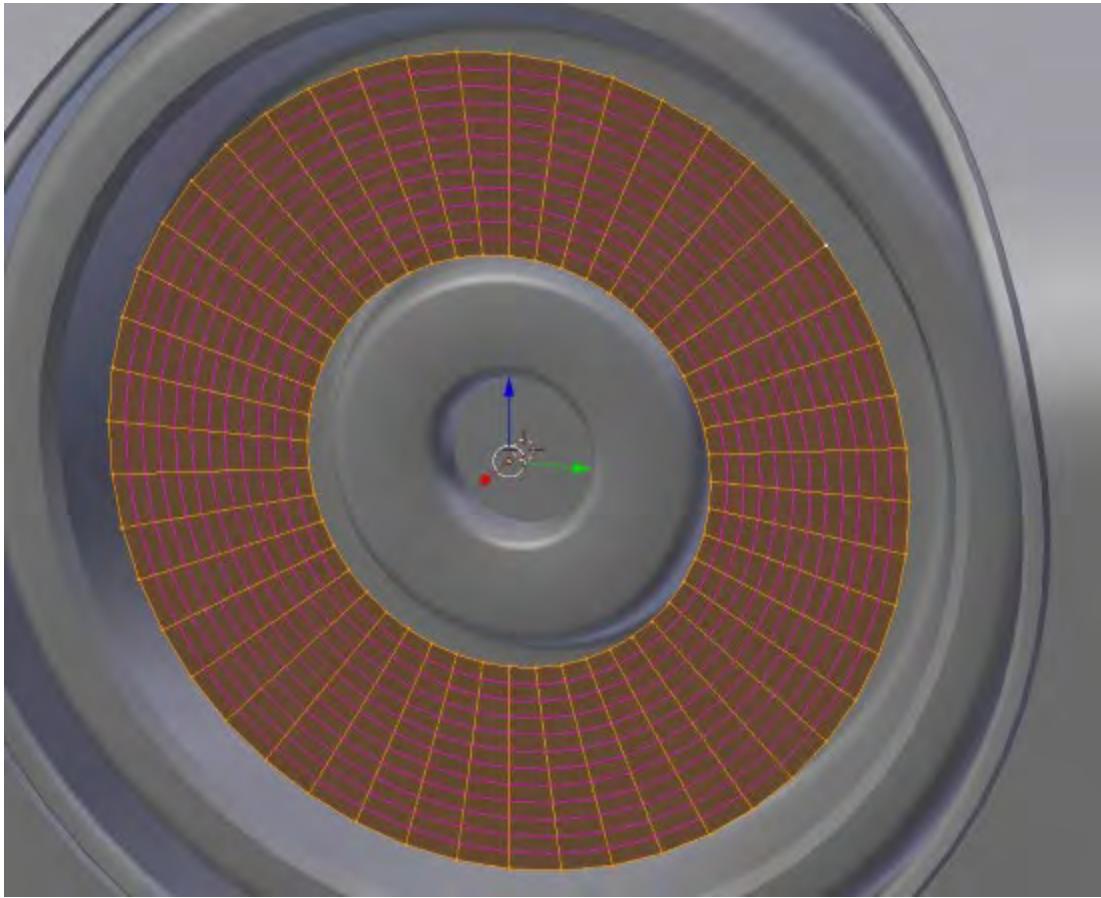


Next, let's select a ring of faces and duplicate it by pressing Shift + D. Then, we'll separate it to another object by pressing the P key.



<pagebreak></pagebreak>

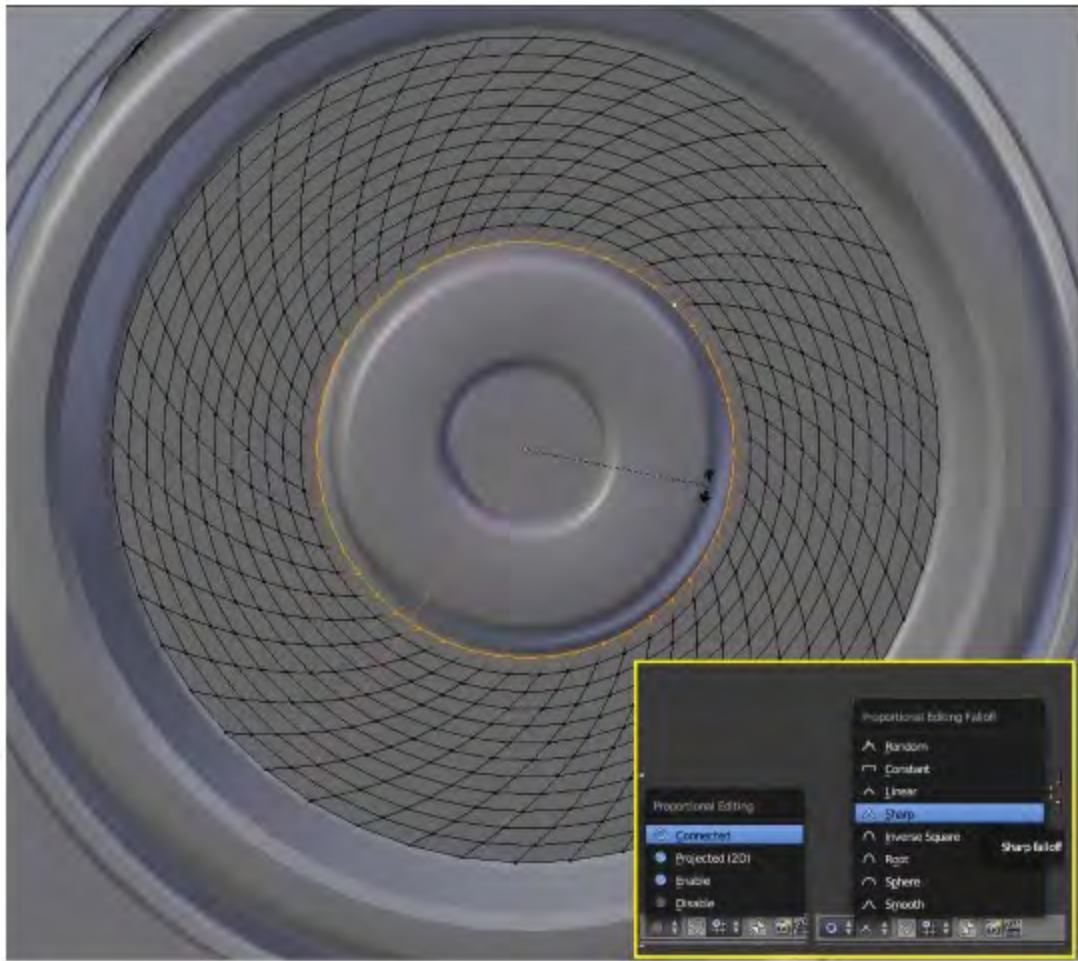
Next, we're going to create some curved cooling fins on the gun. In order to do this, let's first run a series of loop cuts around these faces. This will enable us to use **Proportional Editing** to give a nice, curved shape to our fins.



<pagebreak></pagebreak>

Now that we've done this, let's turn **Proportional Editing** on at the bottom of our screen and select **Sharp**.

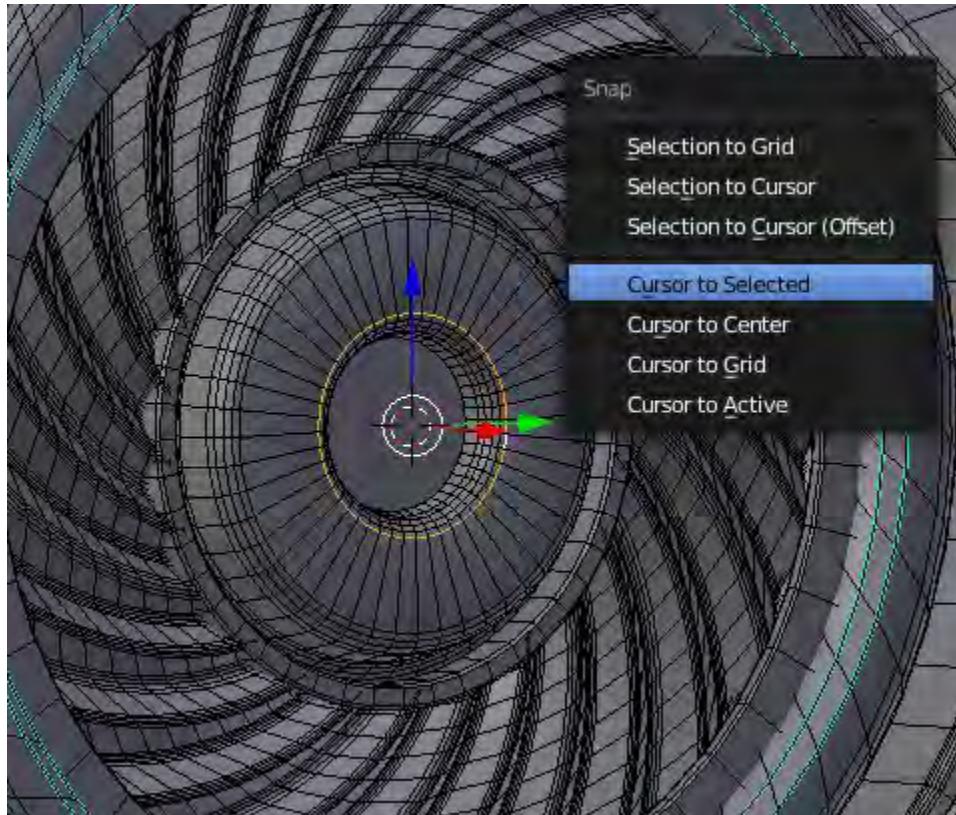
Then, we can rotate the middle ring of vertices, and we can scroll the mouse wheel to adjust the area of influence. Just rotate and adjust until you have given a nice shape to your circle.



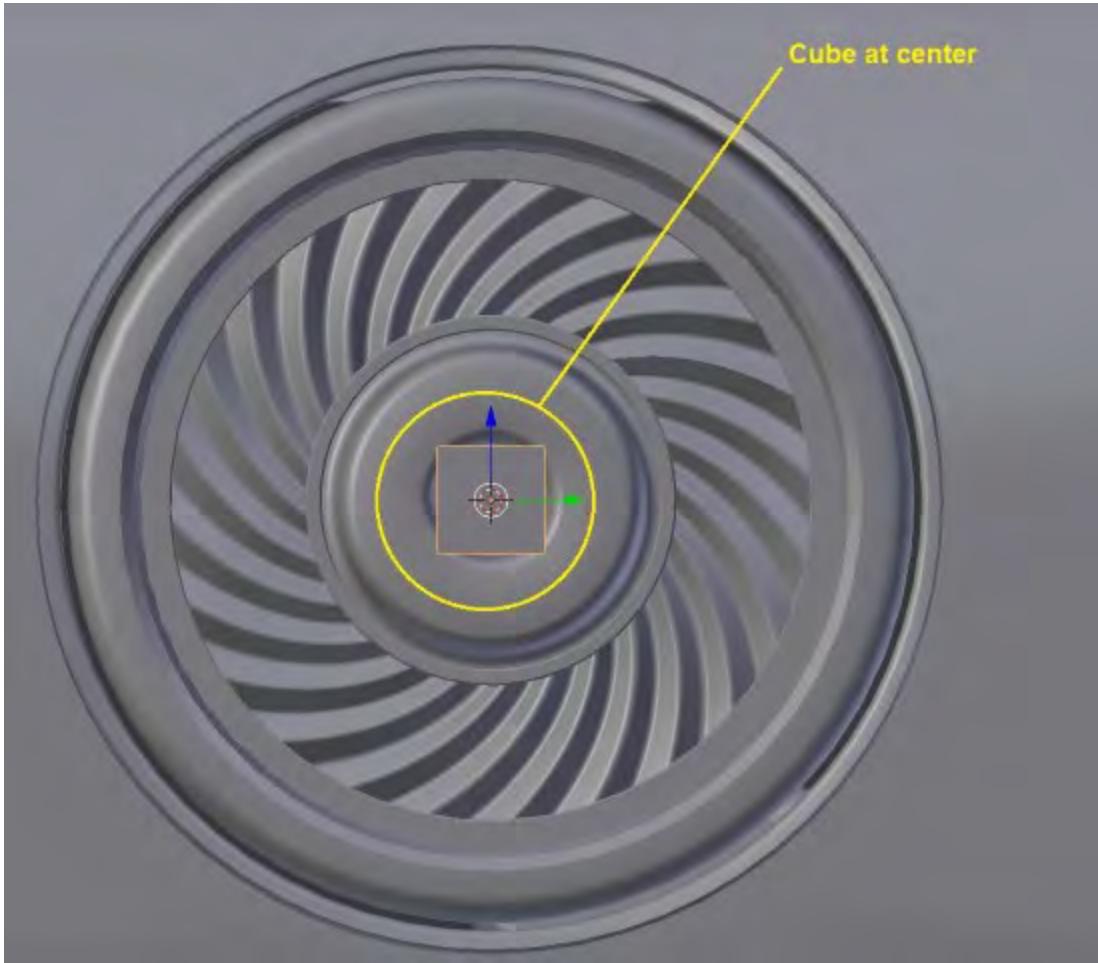
Next, we'll select all other edge loops and push them back towards the center of the gun. Once we have the basic shape of our fins, we can go in there and bevel them. If you'd prefer, you can also delete a few rows of faces to create gaps or holes for heat to escape (this is not shown here).



The next thing we'll do is add a little circular detail to the outside of our recessed circular area. Start by placing the **3D cursor** at the center of the circle. An easy way to do this is to pick a ring of edges and press **Shift + S, Cursor to Selected**.

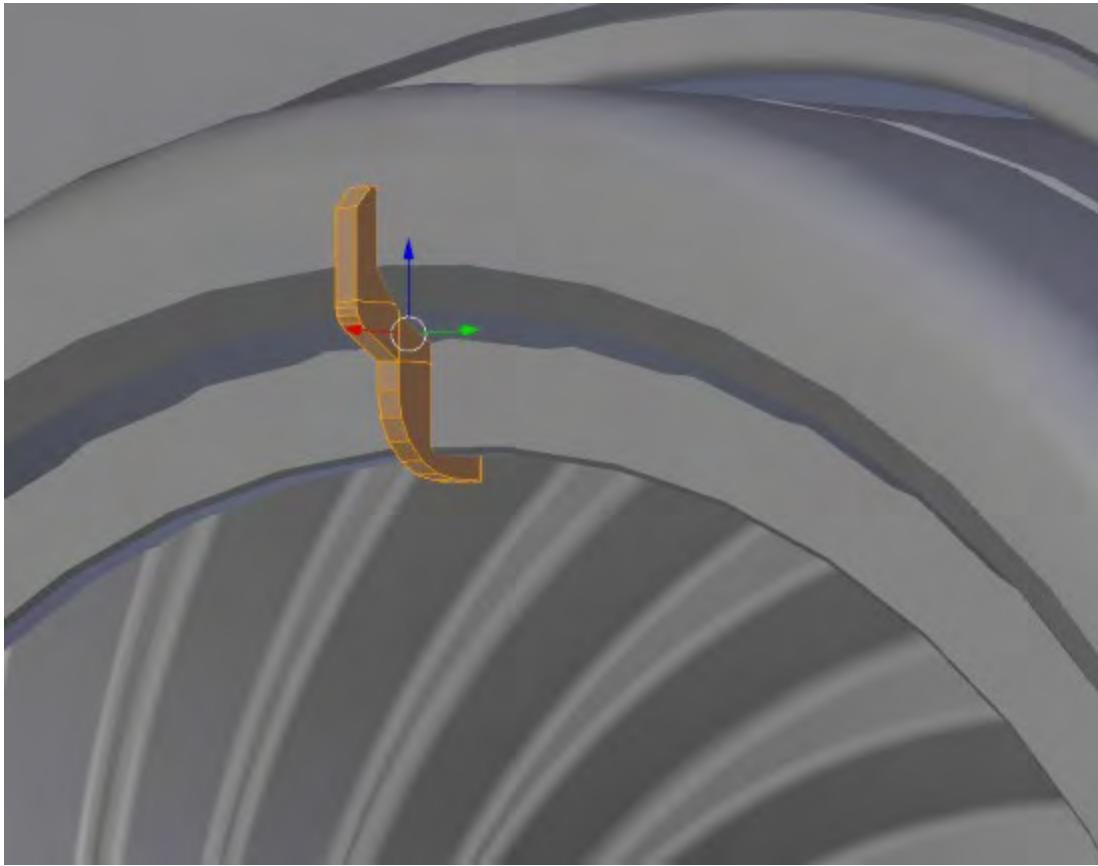


Then, go ahead and add a cube.



<pagebreak></pagebreak>

Next, Tab into **Edit mode** on the cube and pull it up to the border of our circle. Using techniques already demonstrated, go ahead and add a bit of detail to it.



Next, ensure the cursor is still at the center of the circle. If it isn't, go ahead and put it there.

At the bottom of your screen, change the **Pivot Point** around which things rotate to **3D Cursor**.

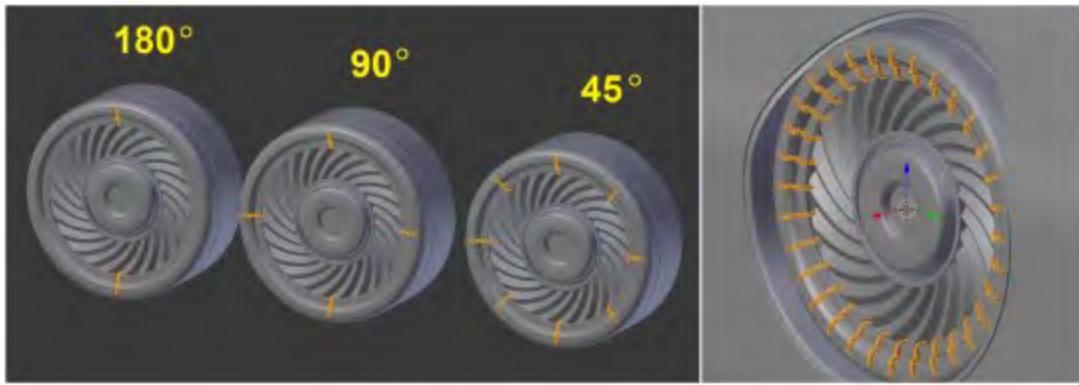


Then, select the entire geometry of the cube (in **Edit mode**), duplicate it, and rotate it by 180° around the **x** axis (press R, X, and 180 to do this). You will now have a cube at both the top and bottom of your circle. Select everything, duplicate it again, and rotate it by 90° around the **x** axis.

<pagebreak></pagebreak>

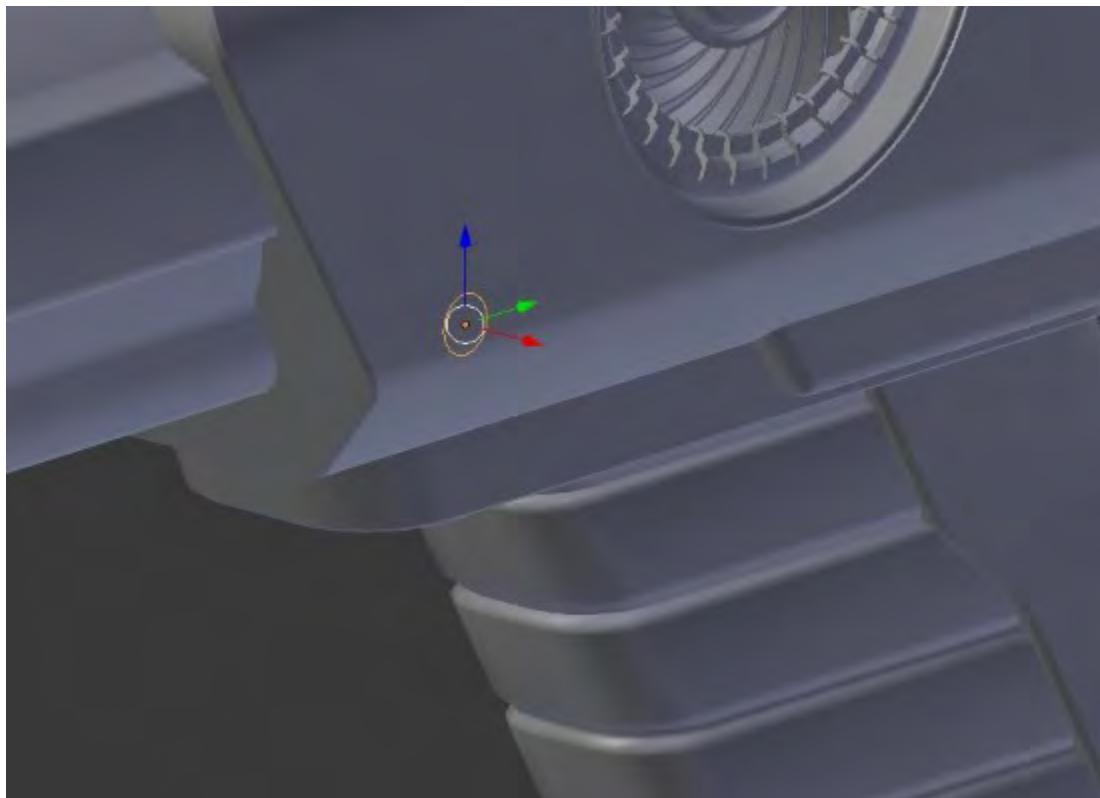
Repeat this process as desired, dividing the angle in half each time. The math will break down as follows:

- 180°
- 90°
- 45°
- 5°
- 25°
- 625°



When you're finished, go ahead and join the object back to the main body of the gun.

Next, we'll add some circular cuts for screw holes. Start by adding a 16-sided circle to the scene:



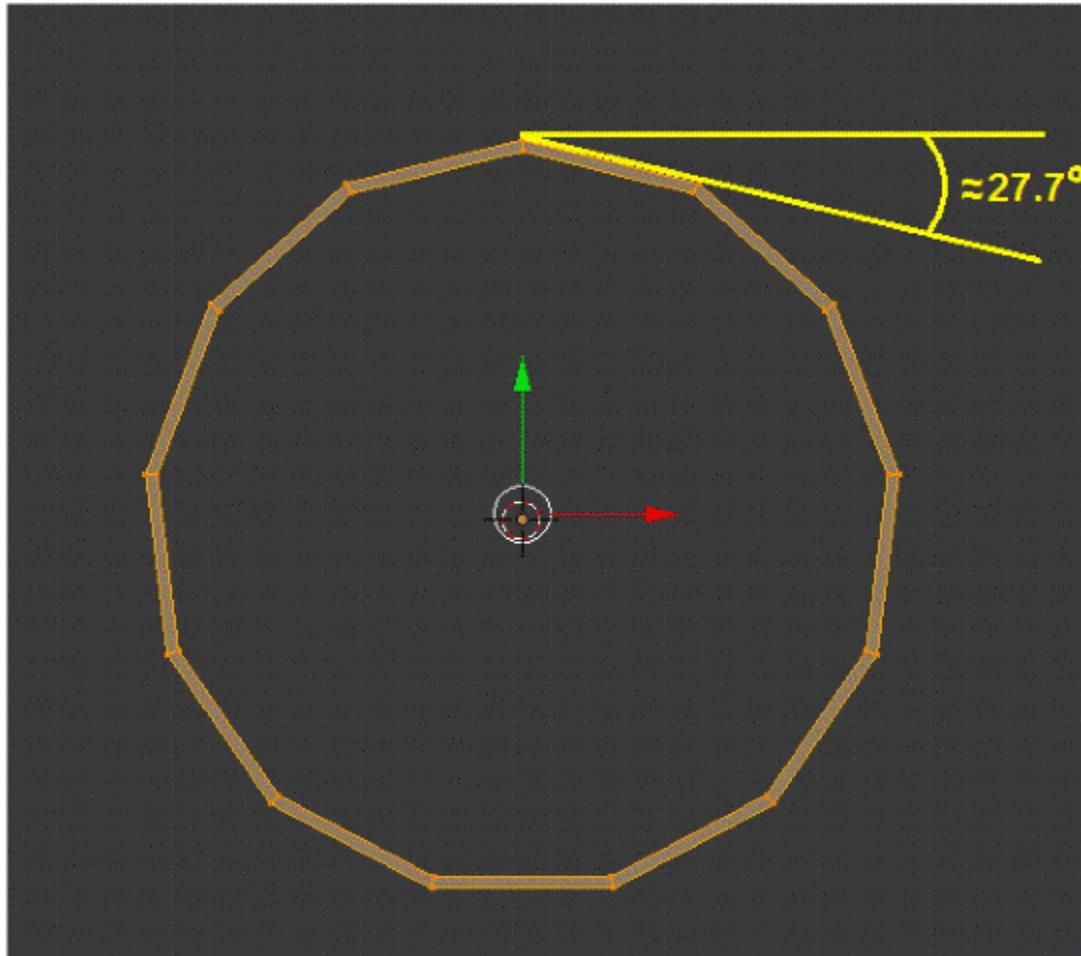
<pagebreak></pagebreak>

Going into **Edit mode** on your circle and move it around until it's in a location where you'd like a screw/bolt to go. Duplicate the circle and repeat this until you're satisfied with the number/location of holes:



Circles, angles, and edge splits

If your Edge Split modifier is set to 30° (by default), then you should never add circular shapes with less than 13 sides to them. Why? Because a 360-degree circle divided by 13 sides equals approximately 27.7° per side.



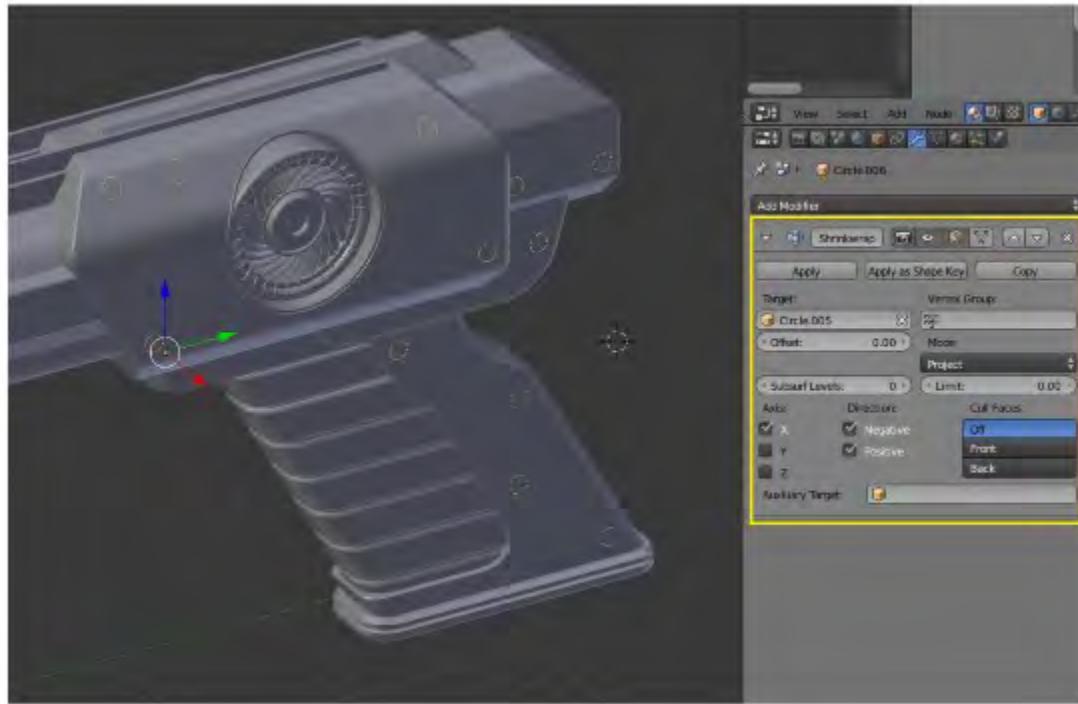
Any fewer sides and your angles are going to break the 30° mark and become sharp. So, 13 is the mathematical minimum. In practice, however, it's not an easy number to work with.

<pagebreak></pagebreak>

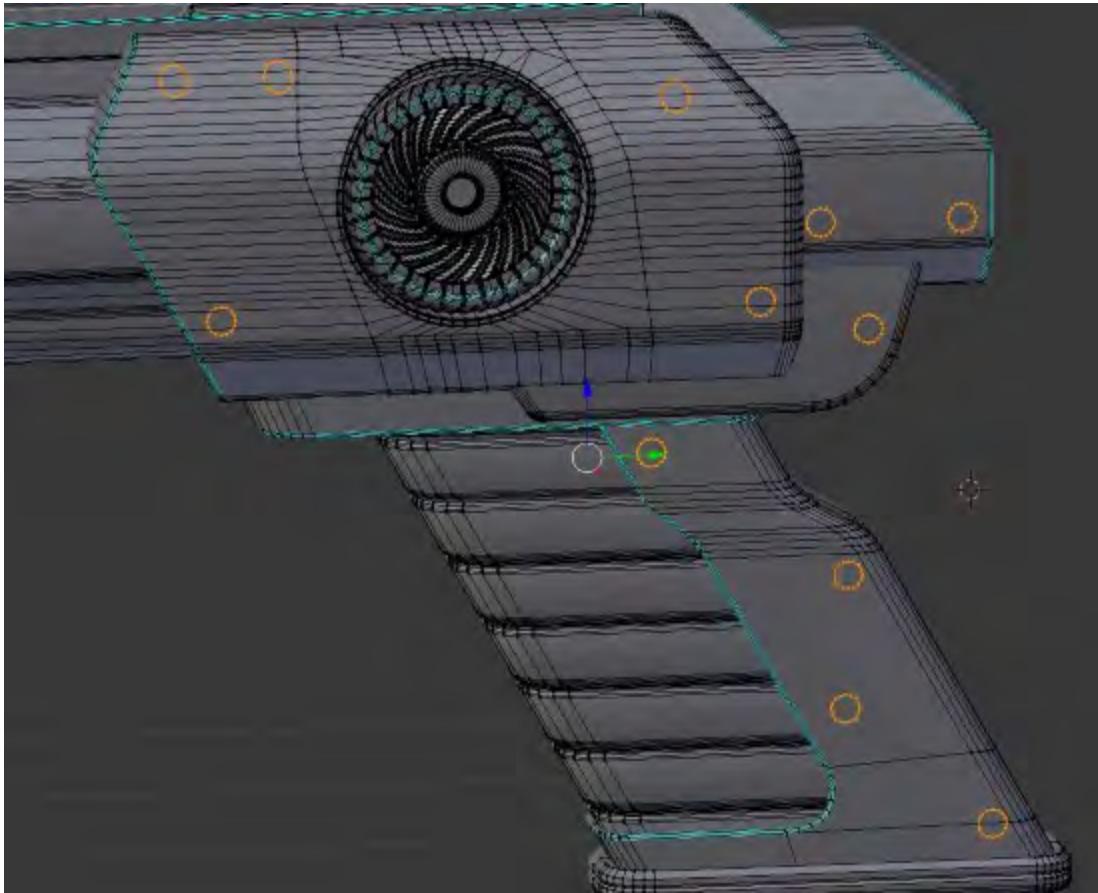
You can't cut it into halves or thirds, and each angle is a long decimal number (27.6923076...). This makes it hard to rotate something (such as a cut out) around a cylinder by "one side".

Instead, I suggest using a minimum of 16 sides per circle. It's easily divisible by factors of 2, and the angles work out nicely.

Next, we'll add a **Shrinkwrap** modifier to our circle object, and press it up against the gun. This is exactly the same process that we just used cut out the large circle.



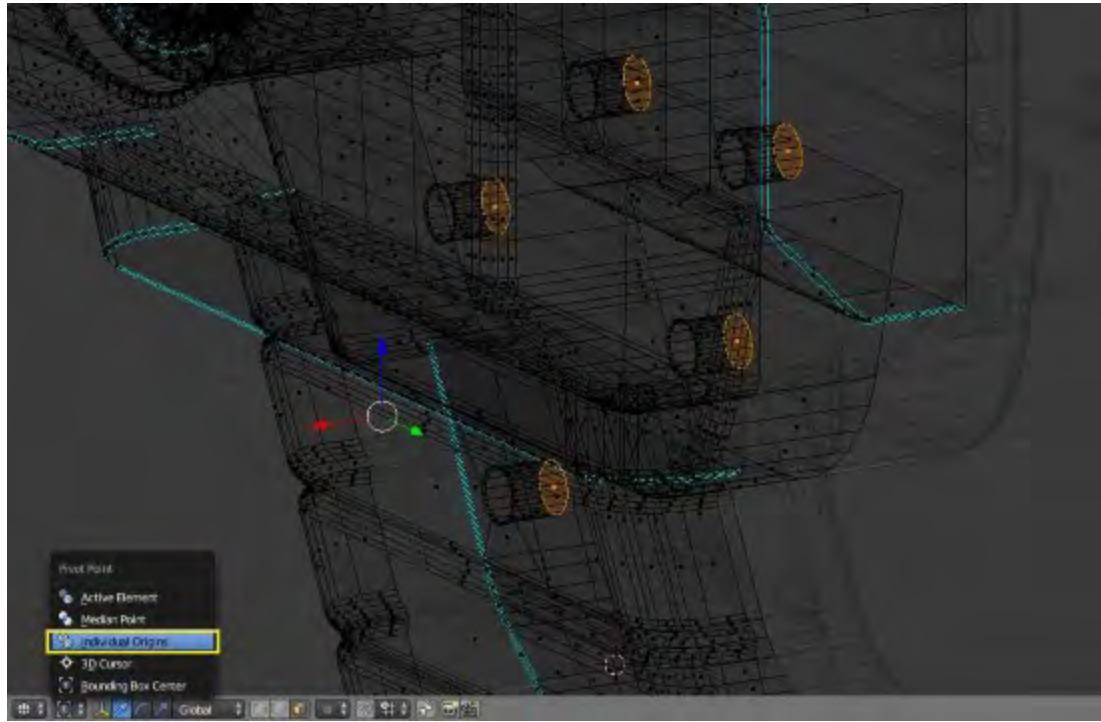
Just like before, apply the **Shrinkwrap** modifier when you're done, and then join the circle object to the gun.



<pagebreak></pagebreak>

Using the same techniques from before, go ahead and build these small circles into the gun's body. As they're smaller than the other circle (and many areas are flat), you will be able to get away with using N-Gons, which will really speed things up.

Once this is done, we'll extrude the edges of these circles back. Unlike our larger circle from earlier, we won't take them all the way to the centre of the object. Instead, we'll pull them back just a little bit. Then, we'll select **Individual Origins** for our **Pivot Point**. Once this is selected, we'll scale all of the faces to 0 on the X axis so that they become perfectly flat. Choosing **Individual Origins** will make each face perfectly flat, but it won't force all of them to be at the same point on the X axis.

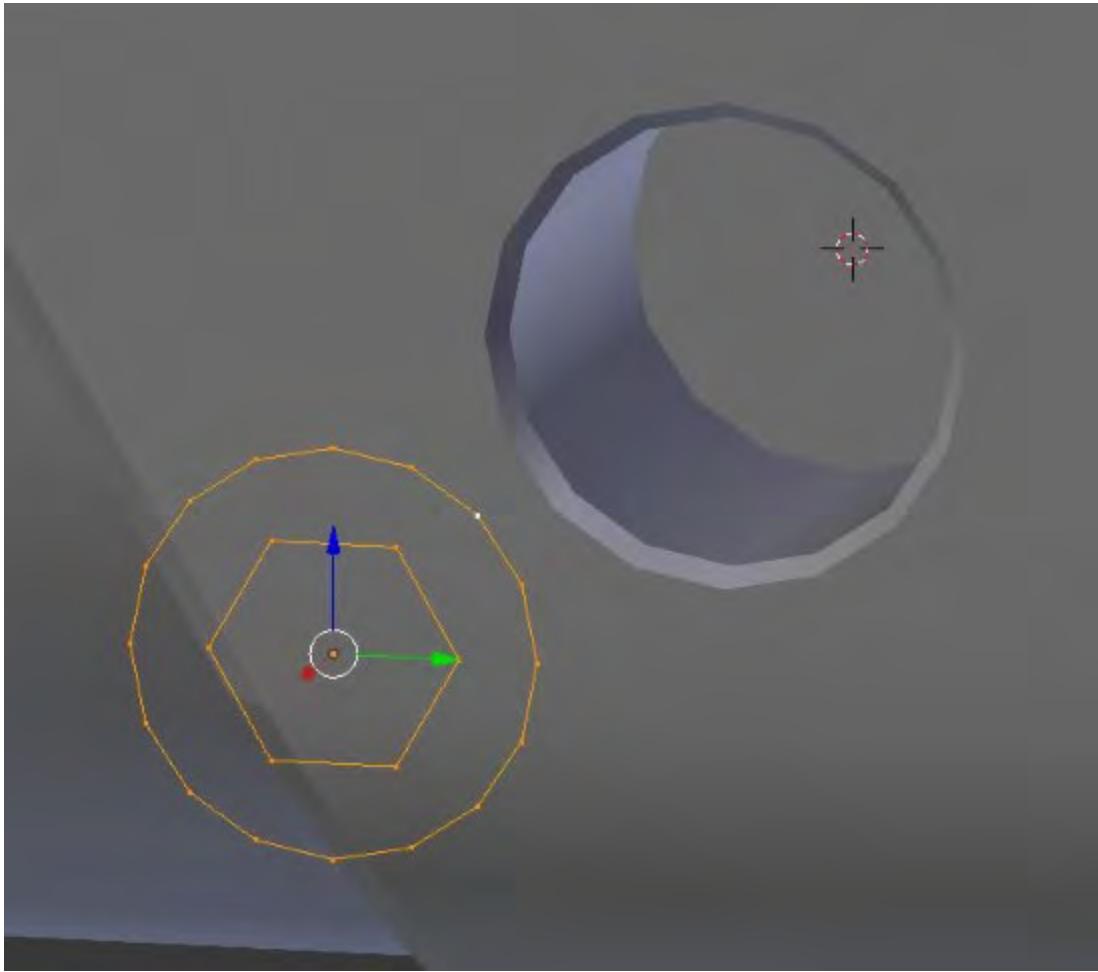


<pagebreak></pagebreak>

You will now have some good holes for screws/bolts to go in:

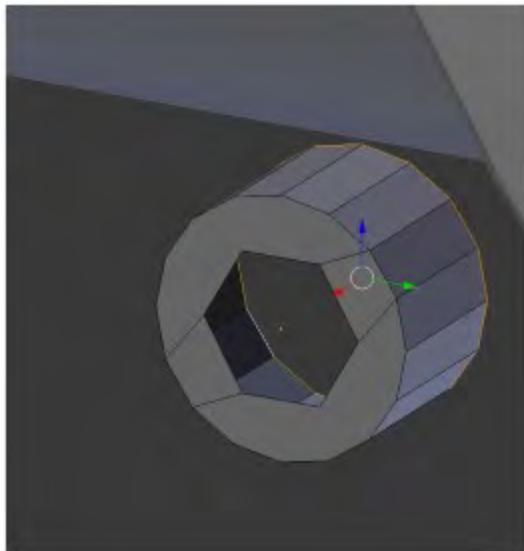


Starting with a circle, go ahead and build a quick bolt/screw:

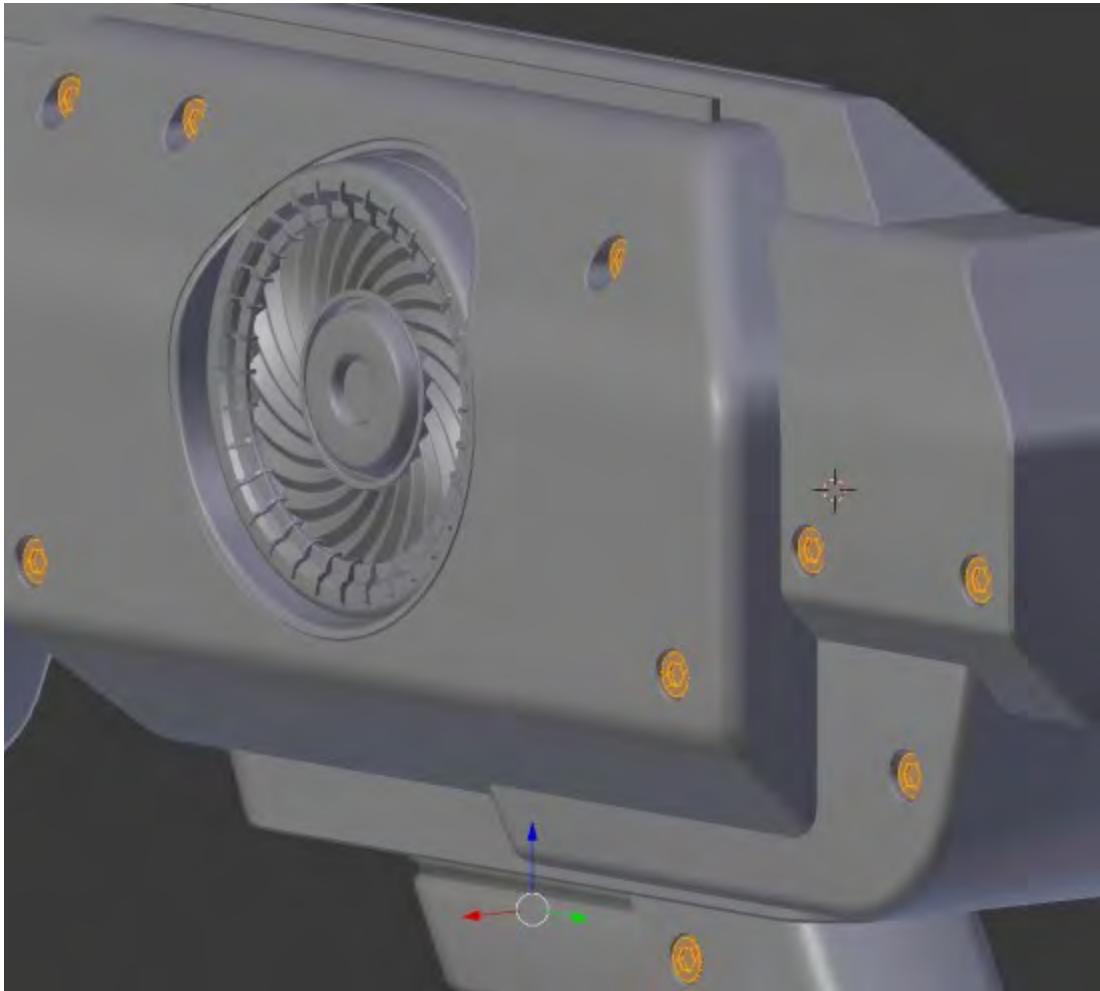


<pagebreak></pagebreak>

You can also extrude some 2D shapes back to create it:

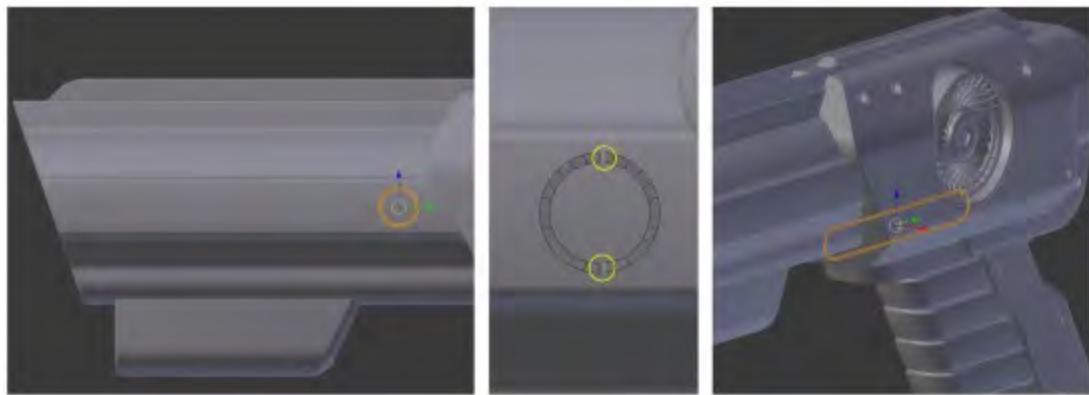


Then, move this into position. Duplicate the bolt/screw, and move the copies around until all of your holes are filled in:

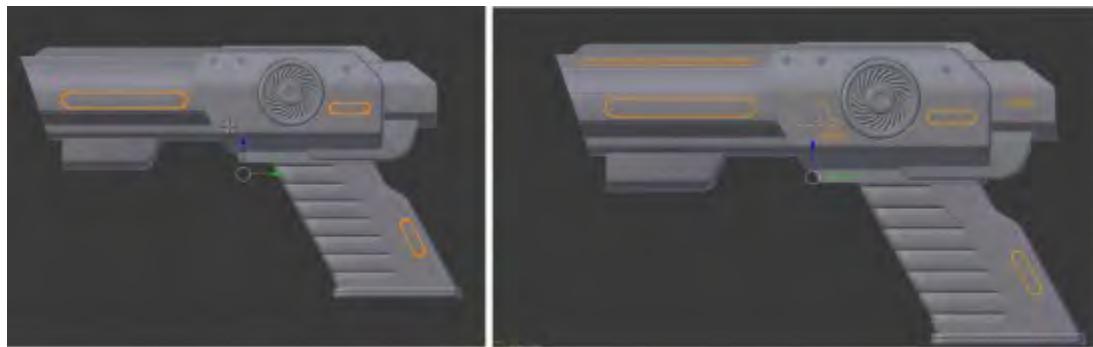


<pagebreak></pagebreak>

Next, we'll make a few other cuts to our model. One technique is to start with a circle, and then rip the two edges along the center. You can then drag half of the circle forward or backward and fill it in. This gives you more of a rounded rectangle, which can look pretty good on sci-fi models:



We'll add a few other cuts as well in various shapes. Use your imagination here! One thing we want to do is leave a nice rounded rectangle cut-out to add a pipe in later:



<pagebreak></pagebreak>

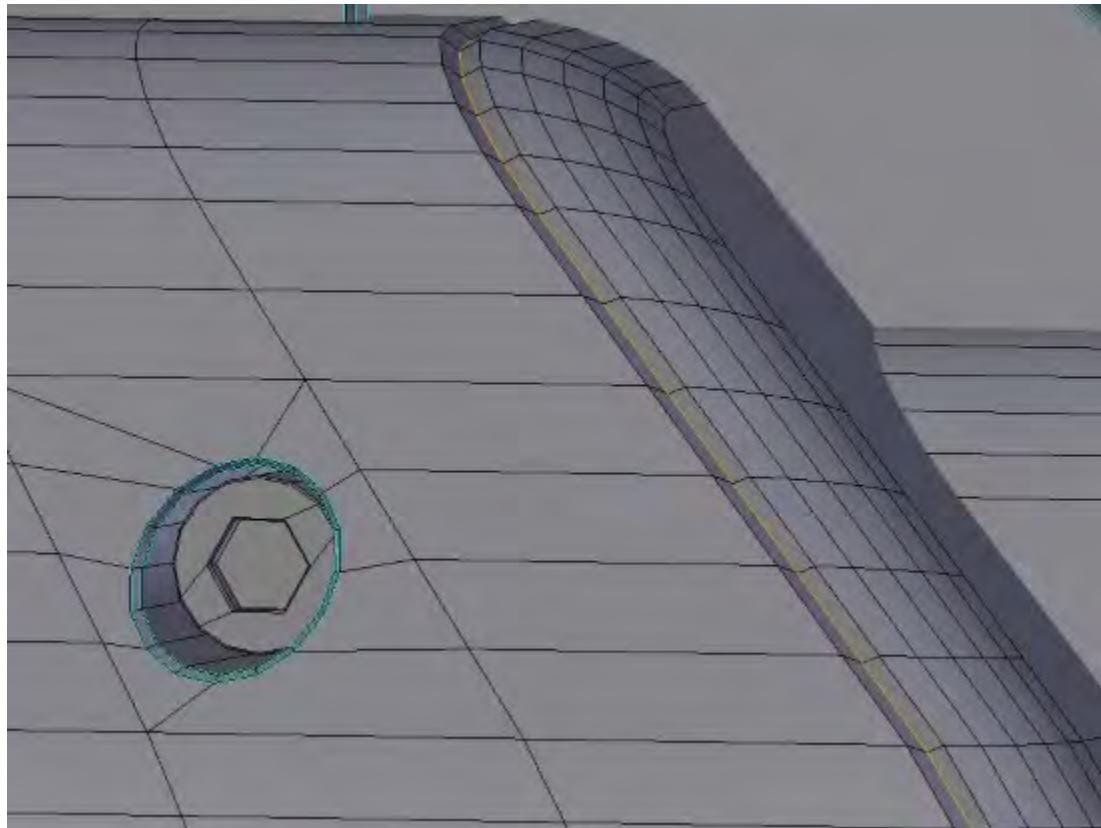
Here's what this gun looks like with all the cuts/holes in place:



Adding final details

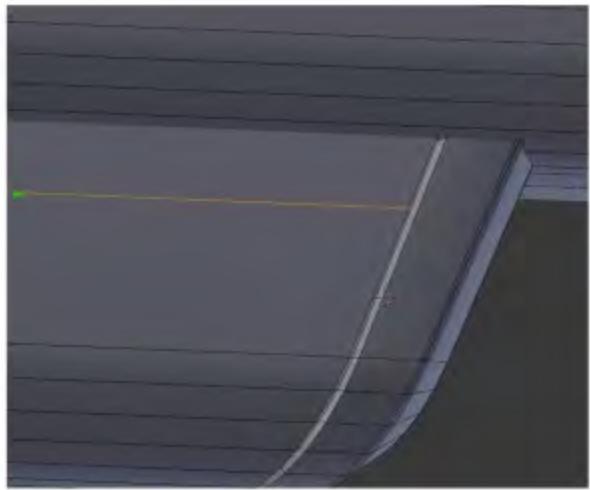
At this point, the gun is really starting to take shape. Let's add a little more detail to finish it up.

A great way to add detail (without too much work) is to run loop cuts across the existing faces, and then scale these new edges in by pressing Alt + S. This gives the appearance of a panel line or cut between different portions of an object:

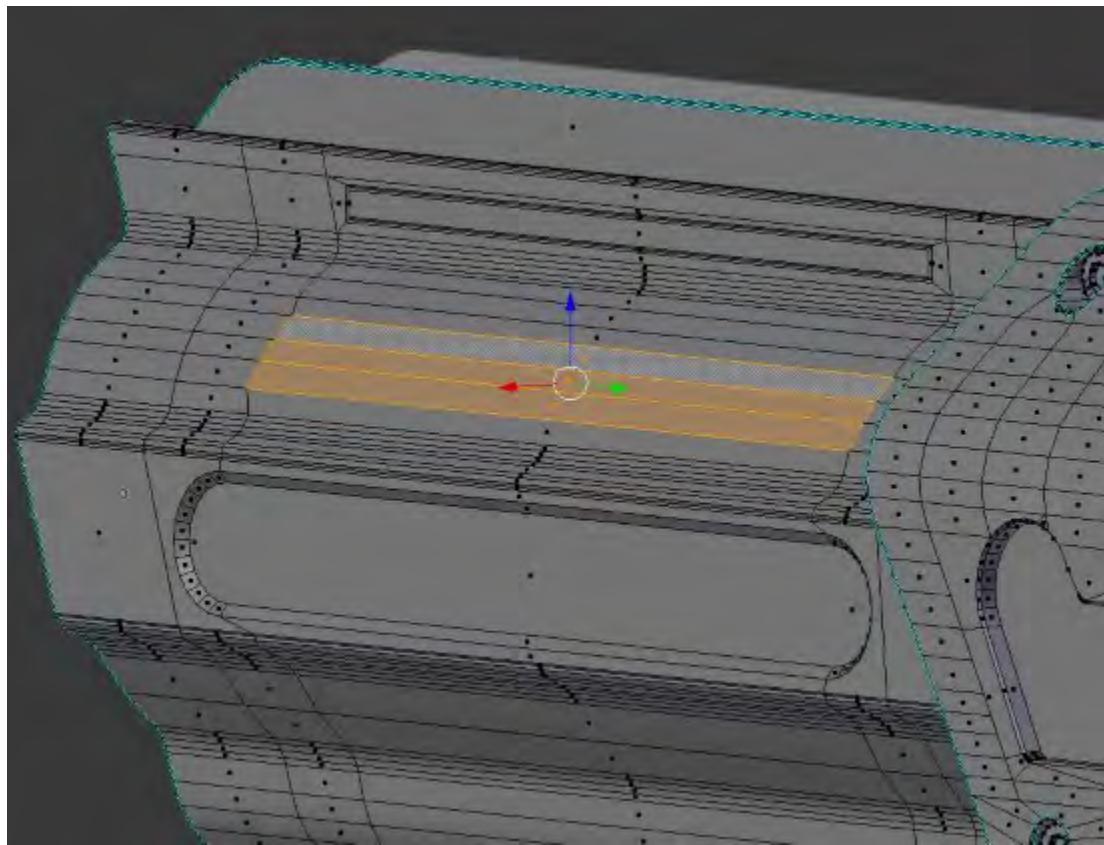


<pagebreak></pagebreak>

You can also create one such cut, and then rip the center edges. This will enable you to create a second line or cut that doesn't go all the way across the geometry:

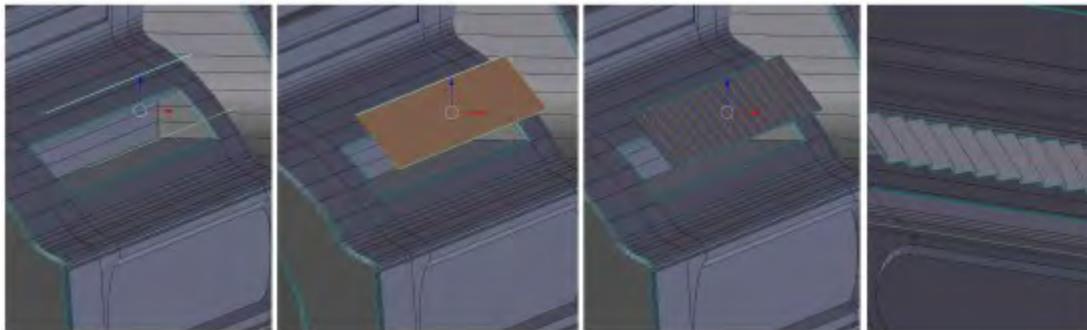


Here, we'll create some additional vents for fins by picking a series of faces on the upper barrel and eliminating them:



<pagebreak></pagebreak>

Then, we can pick the two edges at the top and bottom and fill in a face. With a series of loop cuts across the face, we can easily create our fins:

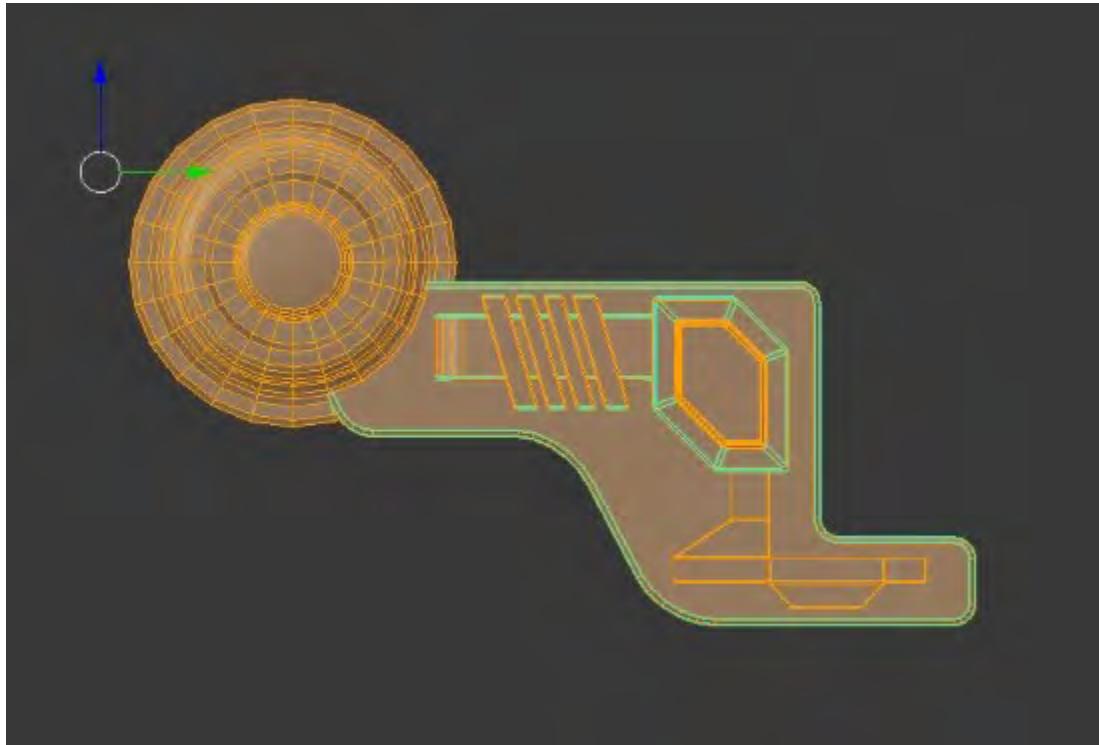


We'll continue the detailing by adding a series of basic shapes to our gun:



<pagebreak></pagebreak>

Then, using techniques that we've already covered, you can go ahead and detail these small pieces. Here's an example of one small piece on the side of the gun's body:



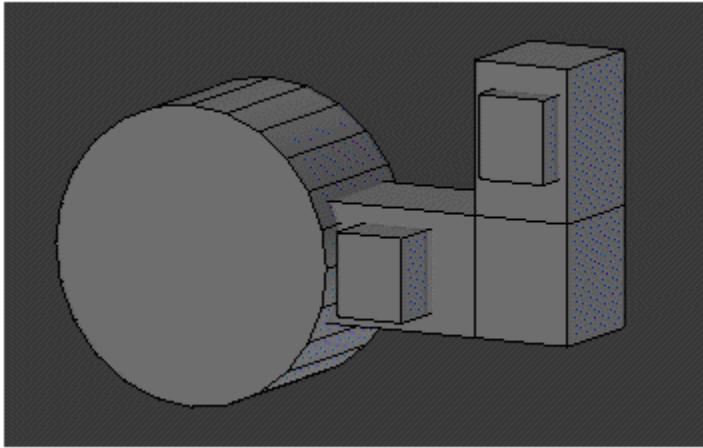
If you'd like, you can look at some real weapons for inspiration. Otherwise, feel free to get creative!

<pagebreak></pagebreak>

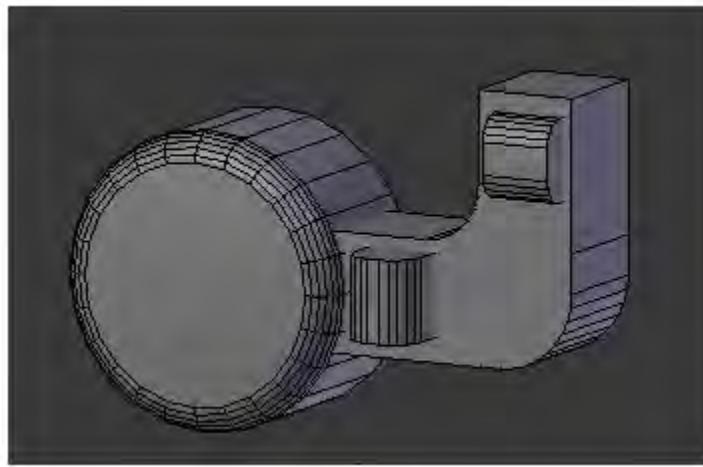
General workflow for detailing

As you do more modeling projects, you'll find that there are certain processes that you use very often. Here's a quick workflow for creating small mechanical details:

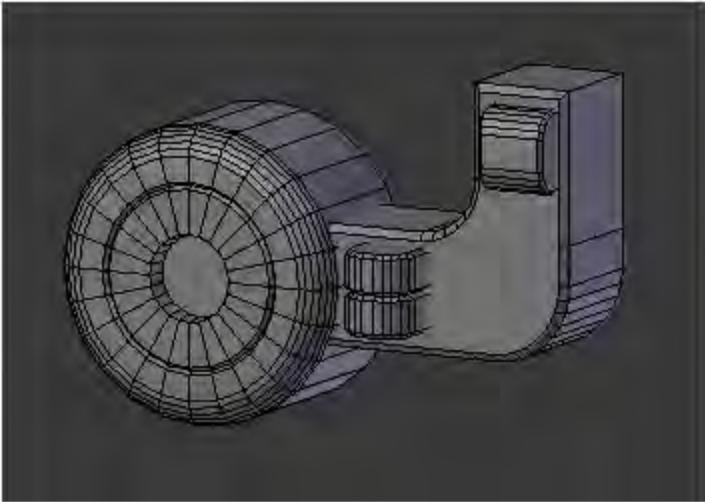
- Add the basic shapes.



- Do your "big" beveling to add curves.



- Do your "small" beveling on the sharp edges.



<pagebreak></pagebreak>

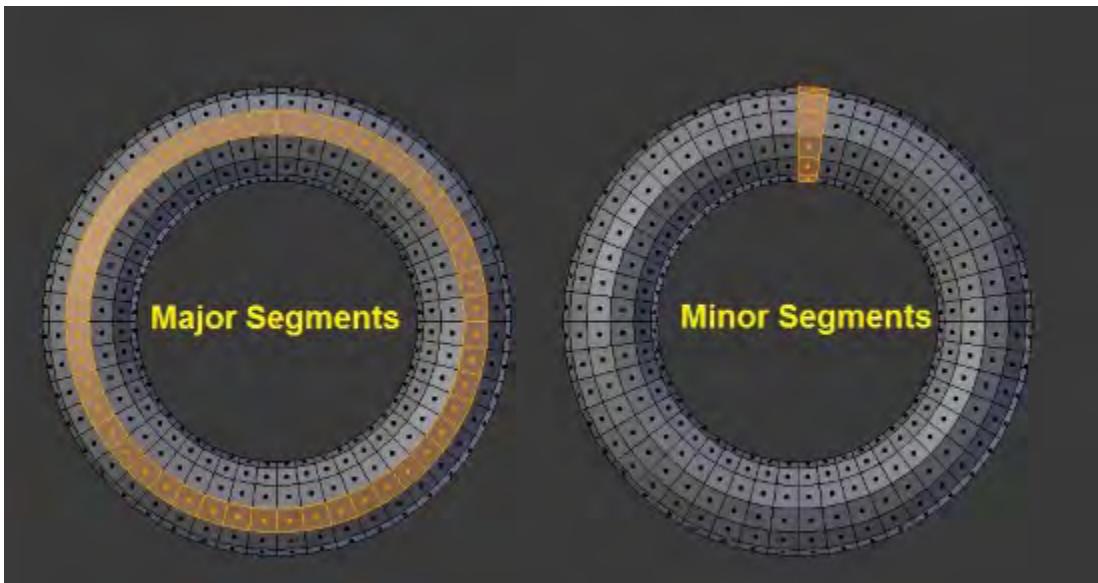
Next, we'll add some pipe-style detailing to our gun. Torus objects are fantastic for creating pipes.

Unlike other types of conduits (hoses, cables, and more), pipes largely consist of straight cylinders. Since they only have curves at certain points, an extruded section of a torus is a great solution. It gives you perfect angles without adding unnecessary polygons.

To do this, let's first add **Torus** to the scene. We'll use **48 Major Segments** (by default) and **16 Minor Segments**.

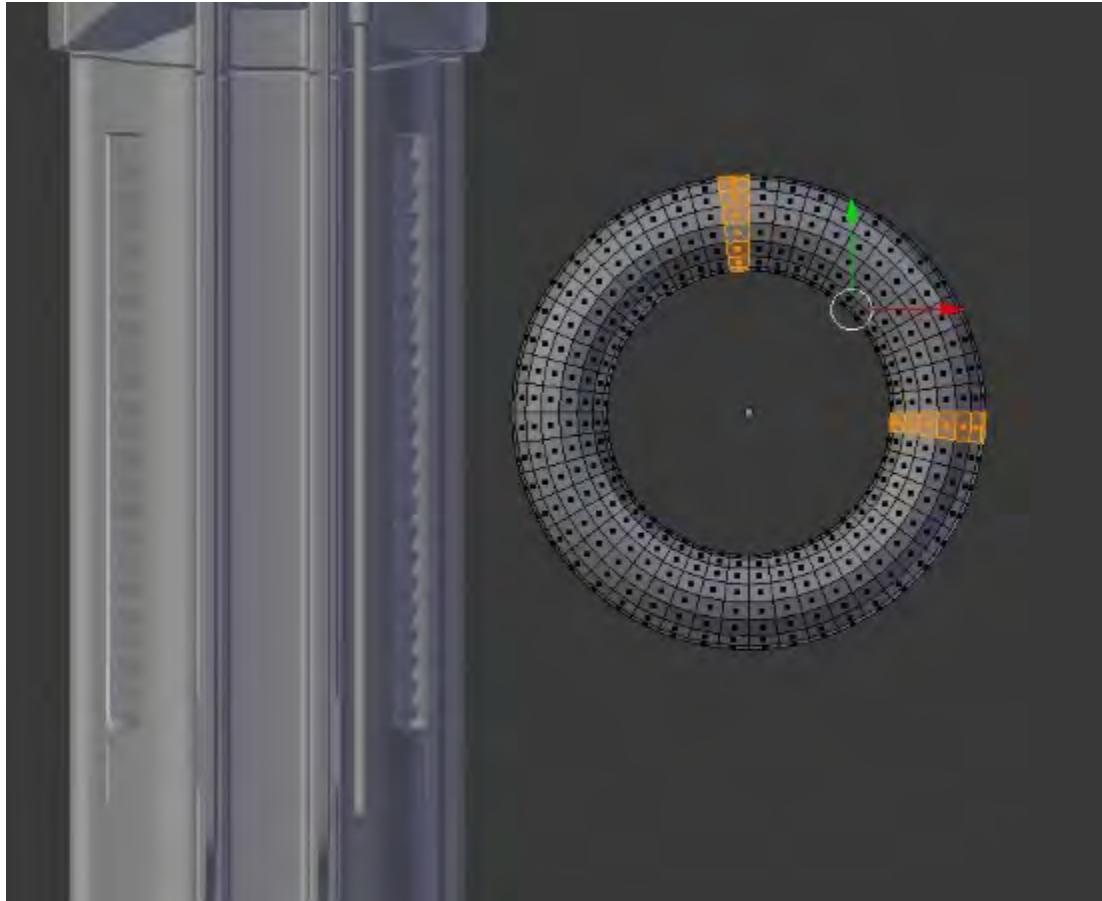


<pagebreak></pagebreak>Major **Segments** is the number of edges around the larger circle. Minor **Segments** is the number of edges around the tube that makes up the torus.

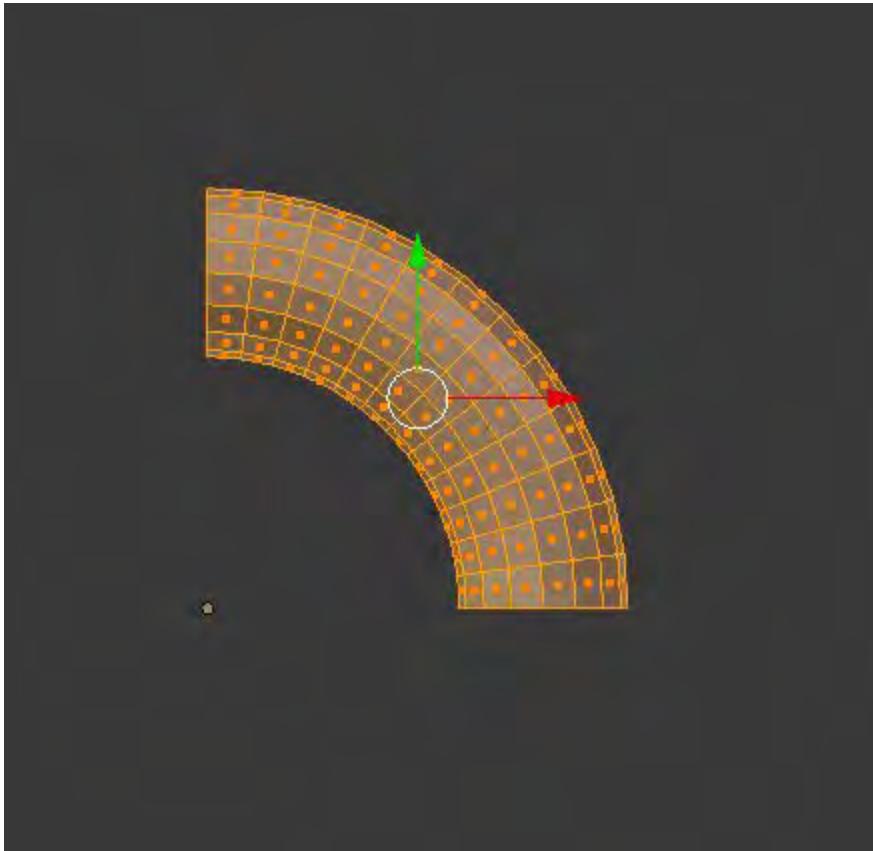


<pagebreak></pagebreak>

Let's start by deleting three quarters of the torus:

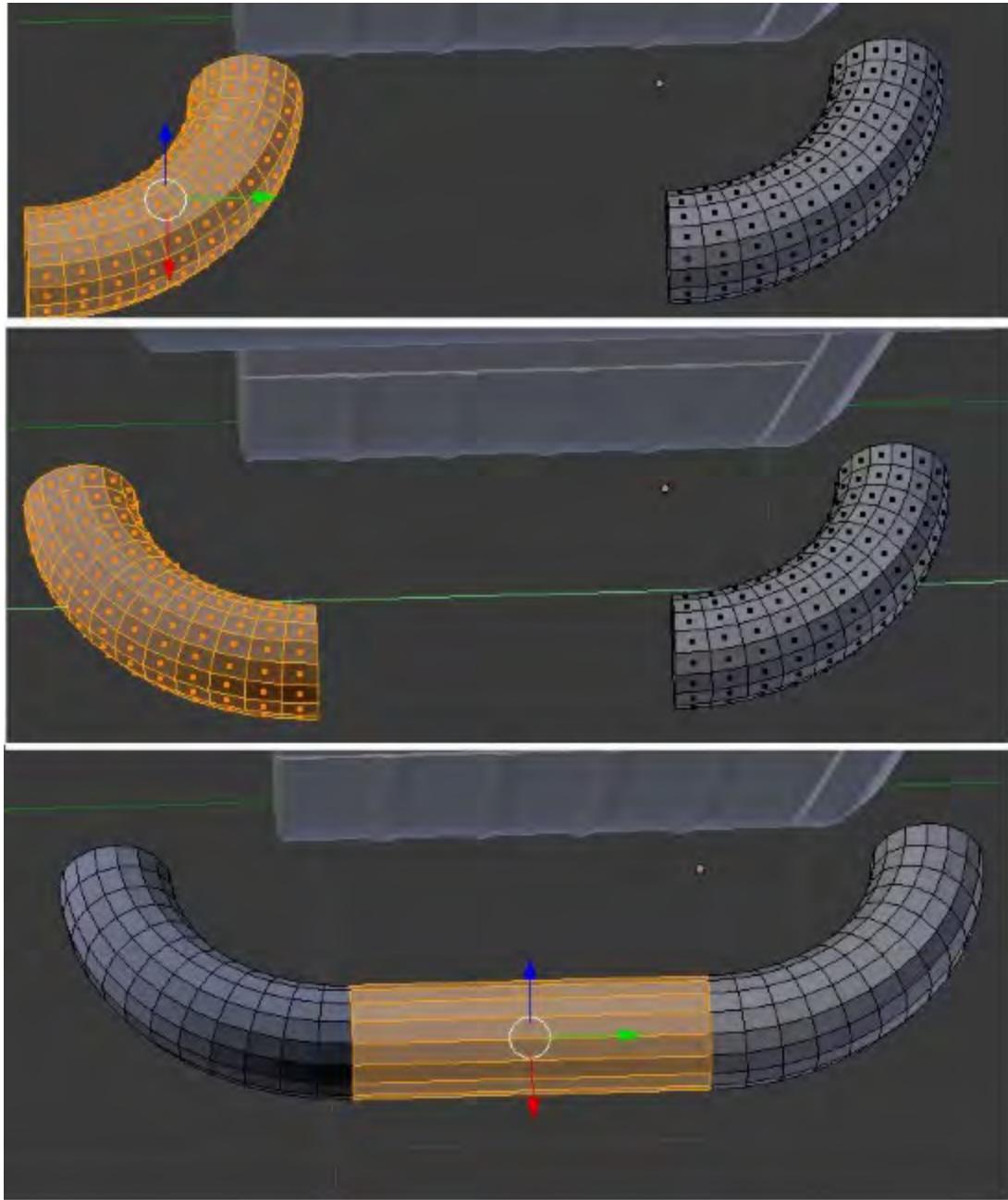


This will leave us with the small section that we need:

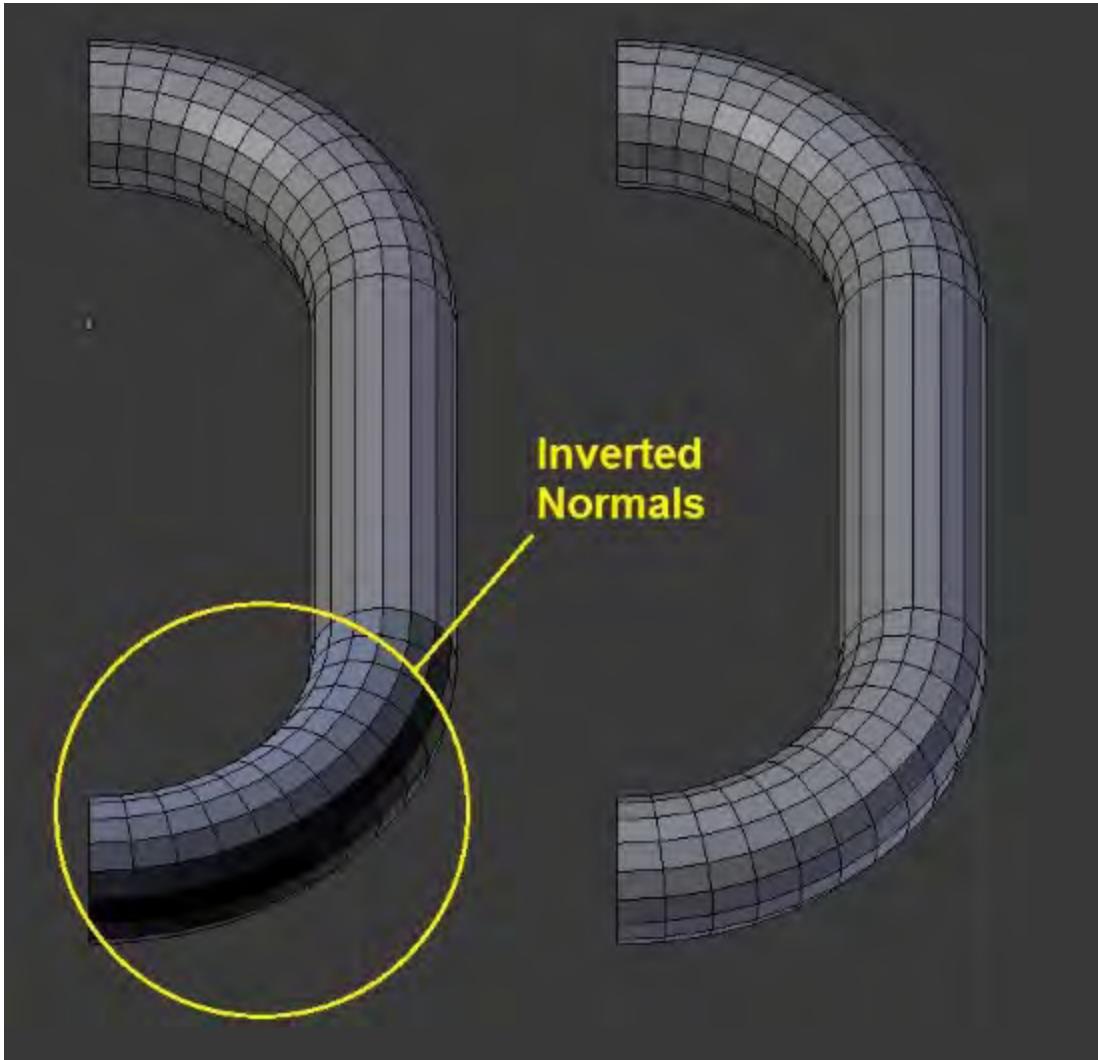


<pagebreak></pagebreak>

You can then duplicate this and flip it by pressing **Ctrl + M**. Then, you can fill in the faces:

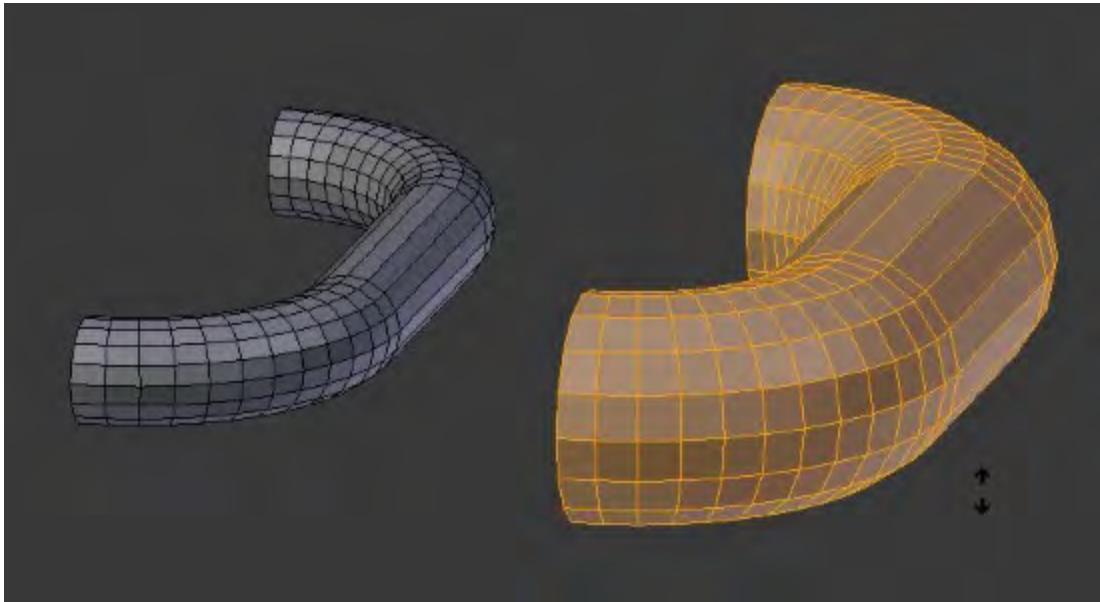


The only downside to doing this is that when you **Mirror** an object (Ctrl + M), you invert the normals. Go ahead and fix them by selecting everything and pressing Ctrl + N:

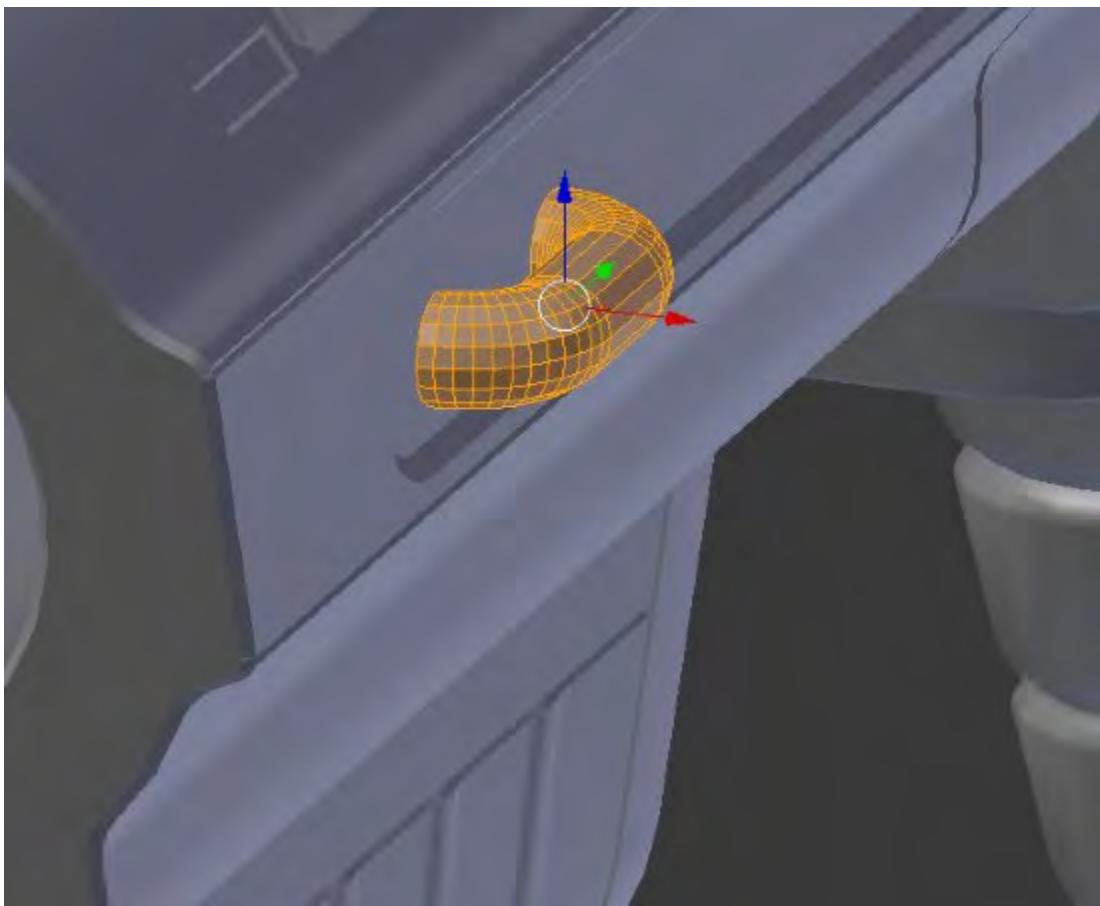


<pagebreak></pagebreak>

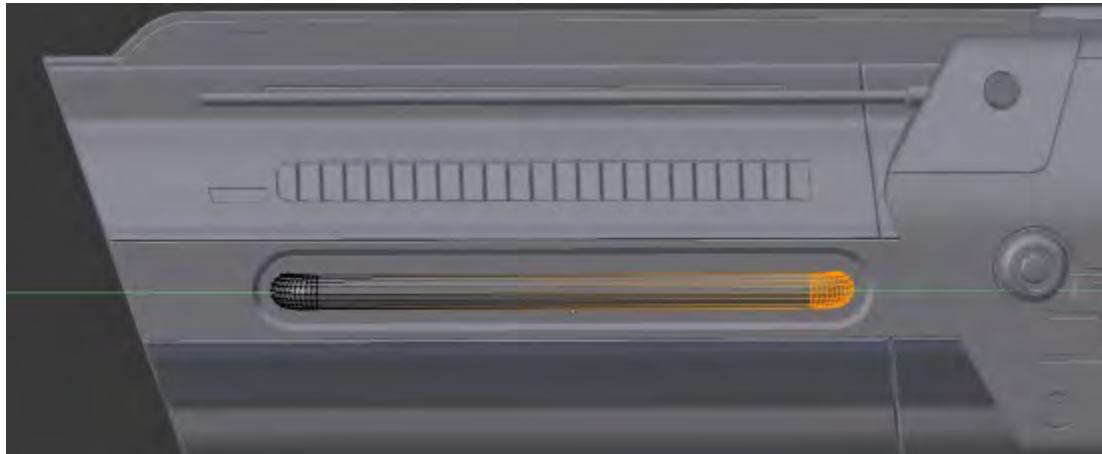
You can also change the width of your torus by scaling along the face normals (Alt + S):



When you have your torus the way you want it, go ahead and move it into position:

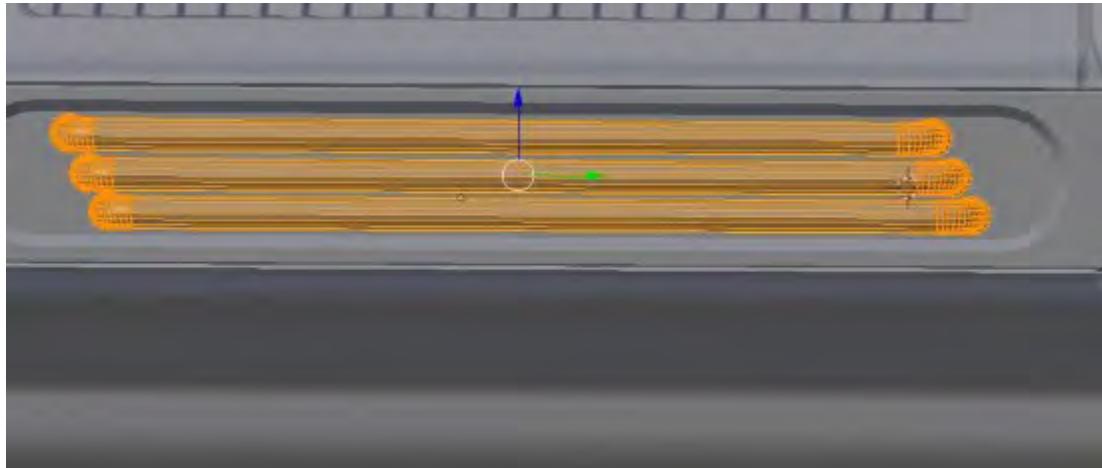


Then, you can select the back half of it and drag it back along the Y axis (or whichever is appropriate):

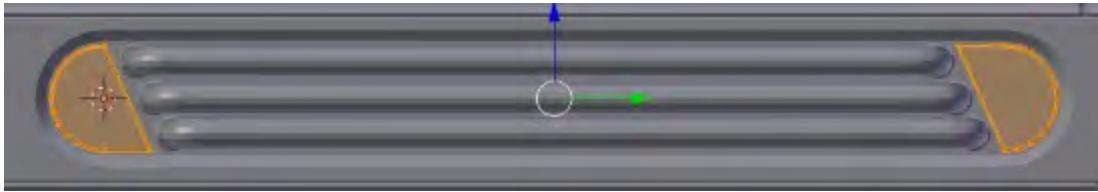


<pagebreak></pagebreak>

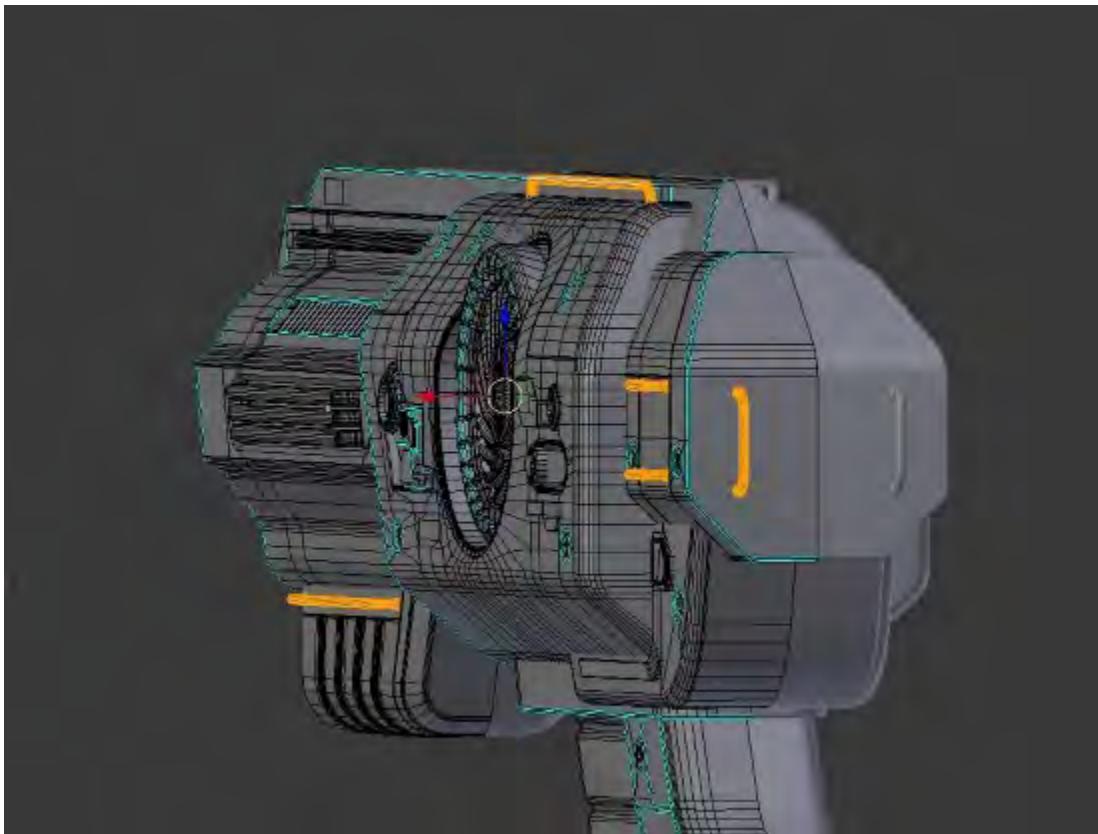
If you'd like, you can add two or more pipes...this is purely a stylistic choice:



You can also add some detail to the ends of your pipe:

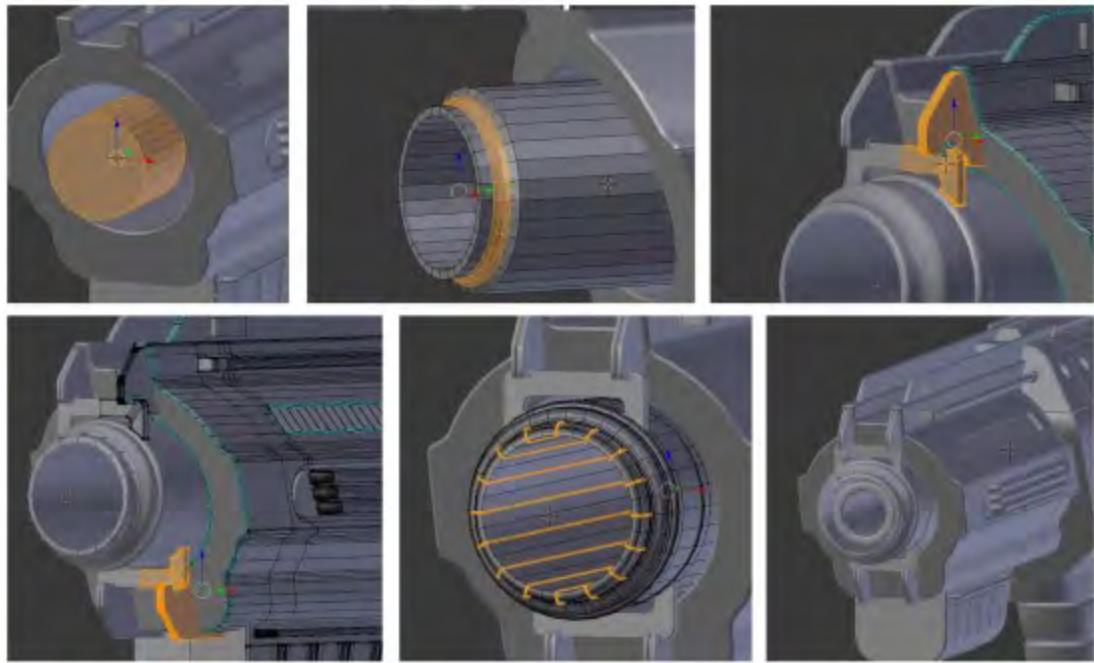


Once you've created one pipe section, don't hesitate to duplicate it, change the diameter by pressing Alt + S, and move it to other the areas of your model:

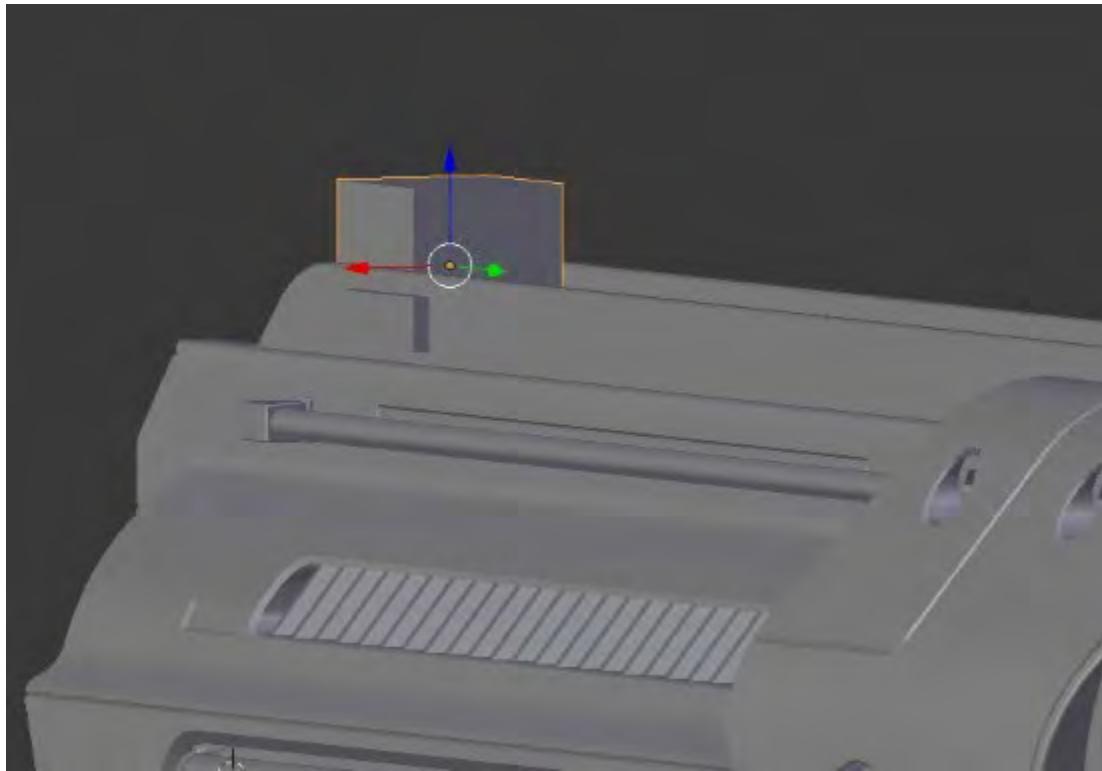


<pagebreak></pagebreak>

Next, we'll add some detail to our gun's barrel. We already have a nice circular hole in the front of the gun, so we'll just fill it in with a cylinder. Then, using techniques previously discussed, we'll add a bit of detail to the front of the gun:

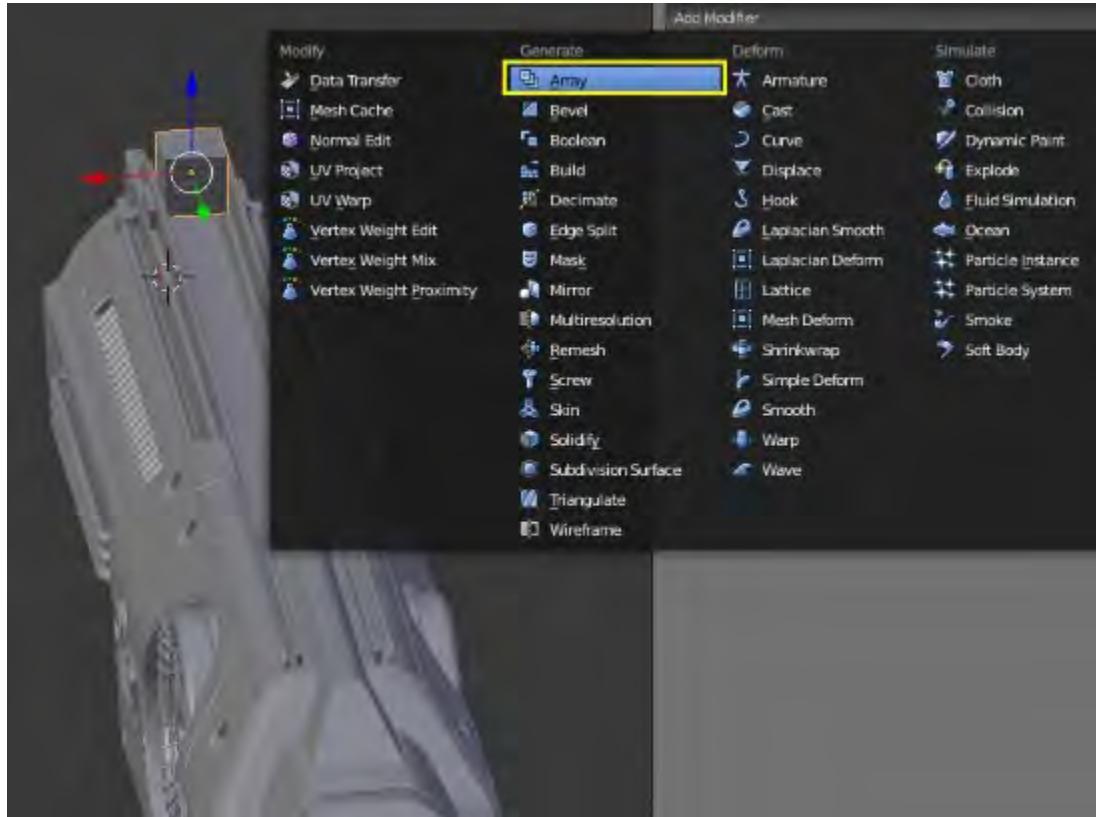


Then next thing we'll do is add some detail to the top with the **Array** modifier. First, we'll add a cube at the top front of the barrel:



<pagebreak></pagebreak>

Then, we'll add the **Array** modifier to it. The **Array** modifier creates multiple copies of an object in a line (or "array") with an offset in position (that you specify):

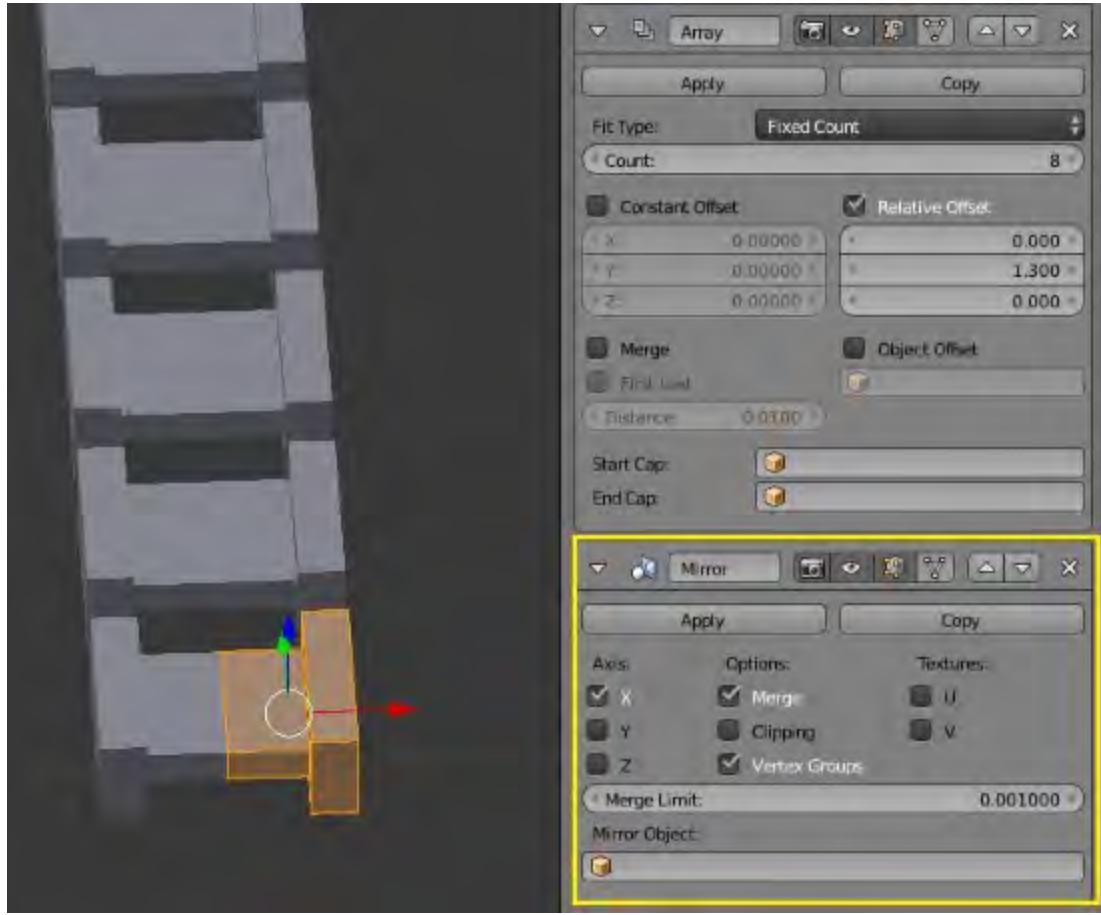


Adjust the **Count** and **Relative Offset** until you're happy with the result. You may need to change these settings as you begin to add detail to the arrayed object:



<pagebreak></pagebreak>

Of course, you can also delete half of your arrayed object and add a **Mirror** modifier to it as well:

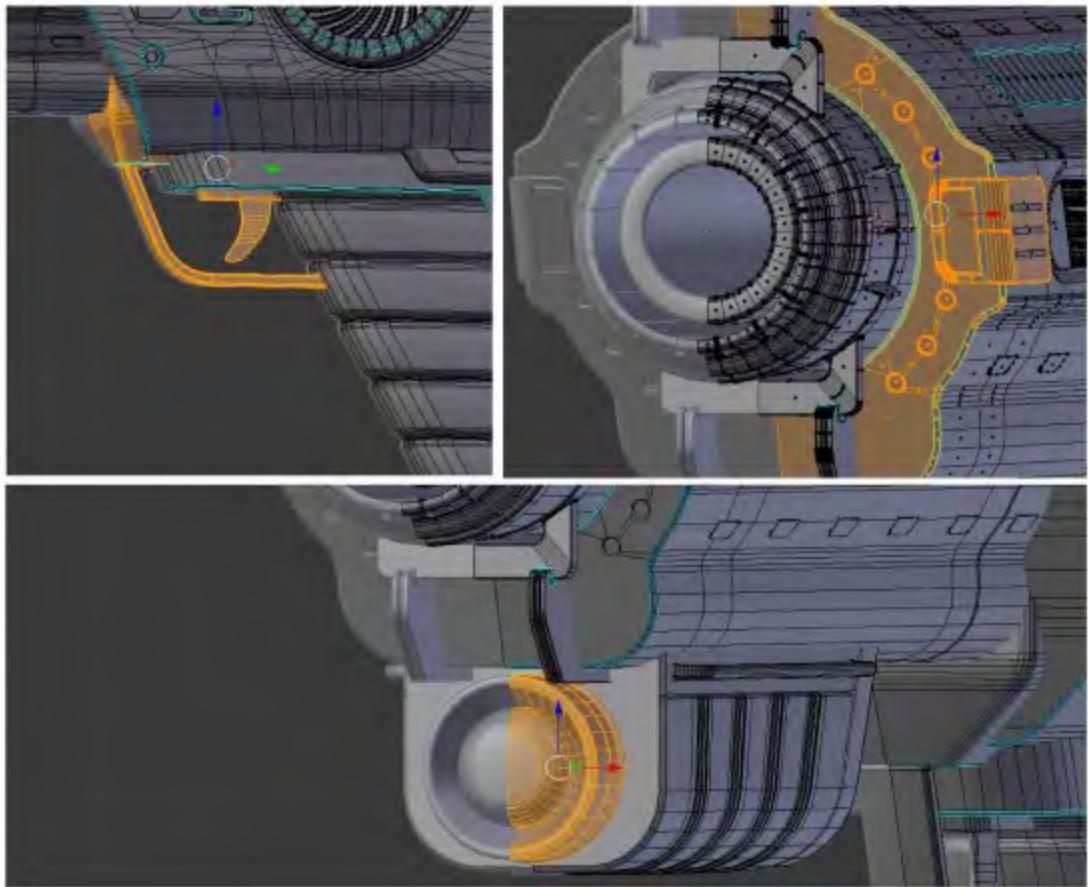


This is a great tool for detailing long, complex components. Use the same techniques that we've covered throughout the chapter to add a bit more detail here. This is where I ended up:



<pagebreak></pagebreak>

We're just about done now, and all of the techniques that we're going to use on this project have already been covered. Use them in various combinations to fill in detail wherever you feel it's appropriate. Don't hesitate to go back and change things if they don't look as good as you imagined. You can always make a copy of the model if you'd like and try something a little different.



<pagebreak></pagebreak>

Here's a quick render of my final gun. We will be creating a render setup in the next chapter, so this is just for demonstration:



Summary

In this chapter, we've gone through and finished modeling our gun. We looked at a number of different modeling tools and techniques to do this. Hopefully, you're starting to get a good understanding of the mechanical modeling process as well. Once you're happy with what you've built, you can bring it to life with materials and textures by adding some basic material slots to the model in the next chapter.

Chapter 3. Texturing and Rendering Your Sci-Fi Pistol

In this chapter, we'll add materials and basic (procedural) textures to the pistol to significantly improve the look of our model. We'll set up a basic scene with the Cycles Render engine and look at some basic materials. You will learn how to apply them to your gun. These techniques will also be useful in our future projects; they are as follows:

- Preparing our scene
- Enabling GPU rendering
- Adding materials and textures

We'll be adding Cycles-based materials and a basic render setup. When we're done, our gun will look like this:



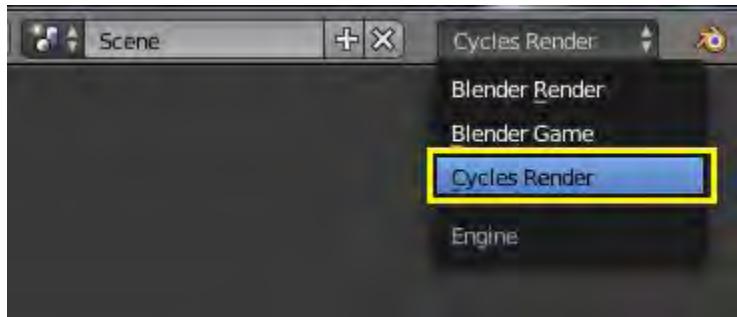
Of course, this is just what my gun looks like. Once we're done with this chapter, we will have no trouble adjusting various materials and settings to make it look however we'd like!

Preparing our scene

The first thing we have to do is set up our scene. Right now, the gun is probably the only object that we have. If it isn't, go ahead and delete any other objects, lights, or the background stuff that you may have.



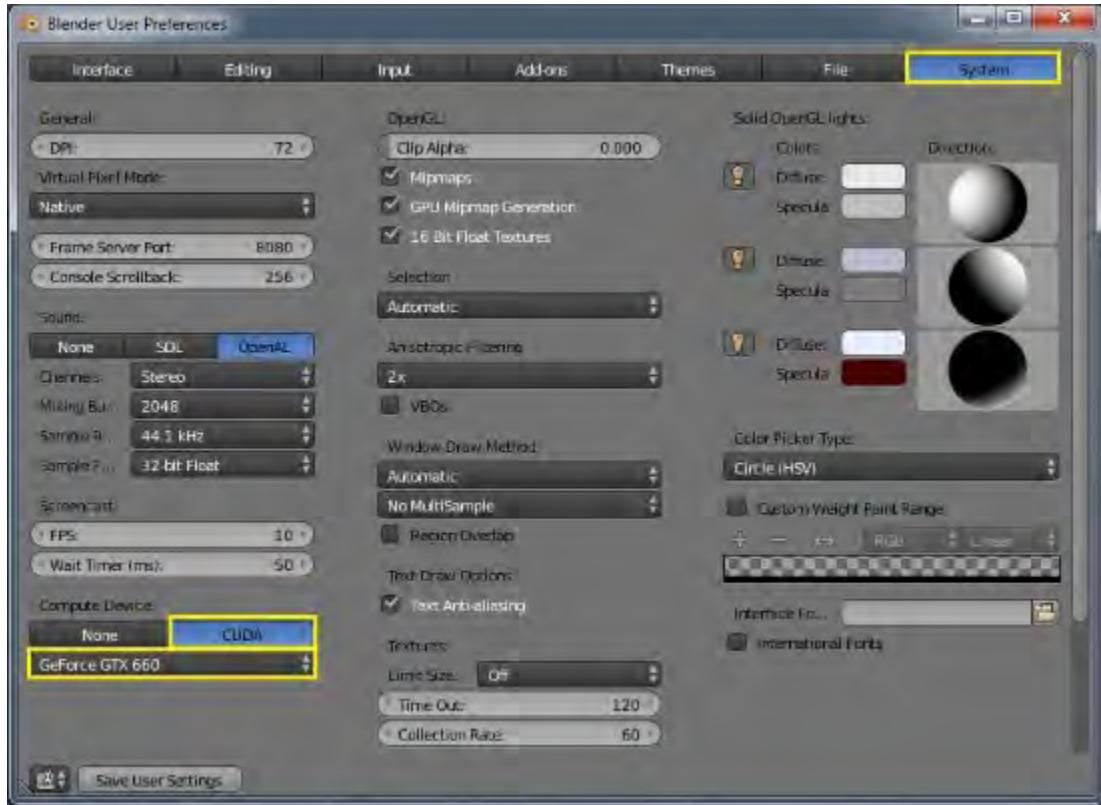
The first thing we need to do is make sure that we're using the Cycles Render engine. This can be selected from the drop-down menu at the top of your screen:



Next, we'll move over to the **Render** tab of our **Properties** panel and check a few settings. We want to make sure that our **Device** is set to **GPU Compute**. This enables Blender to use the GPU on your graphics card, rather than the CPU in your computer. By selecting this, you'll drastically improve your render timings.

Enabling GPU rendering

If the **GPU Compute** option is not available to you, you may have to enable it in **Blender User Preferences** under the **System** tab.



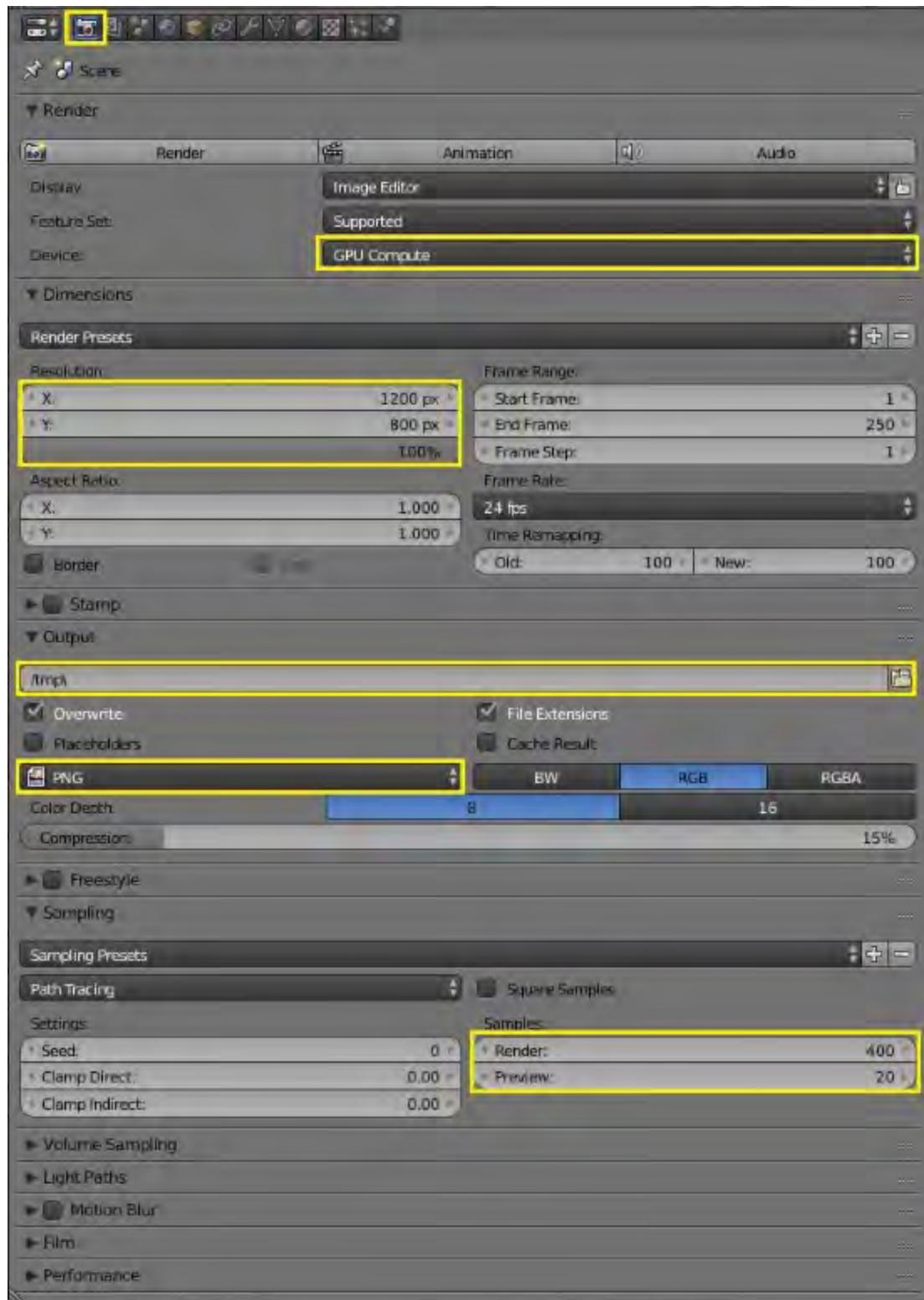
You can also set your **Resolution** at this point. This is how big the rendered image will be. The bigger the image, the longer it will take to render.

Under your **Output** tab, you can select the default directory where Blender will store rendered images. In general, this is mostly used to render videos or a series of still images. If you only render a single image, it's very easy to manually save it wherever you want.

Also, under the **Output** tab, you'll be able to select what type of image or video format you want to render. In this case, we'll use PNG.

Finally, we'll set the number of **Samples** under the **Sampling** tab. The **Render** setting tells Blender how many samples to use when producing a final render.

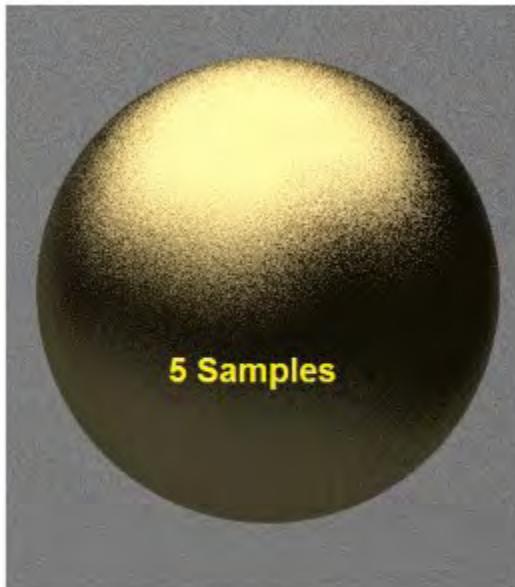
The **Preview** setting tells Blender how many samples to use in the 3D window when you're in the **Render Preview** mode (more on that in a minute). For now, we'll change these settings to **400** and **20** respectively.



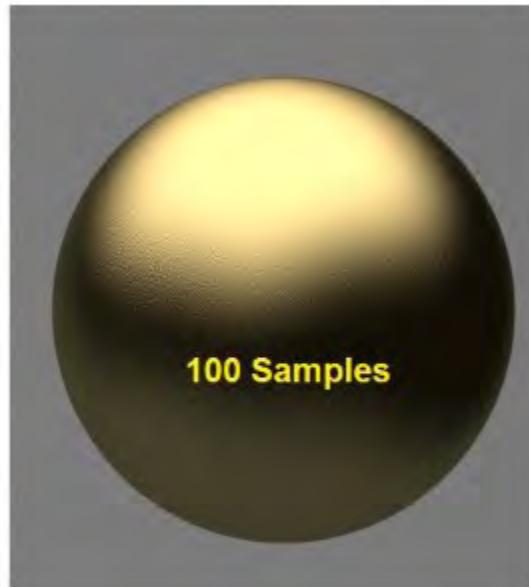
<pagebreak></pagebreak>

Note

Sampling in Cycles The more samples you use, the less grainy your image will be. However, using more samples will also increase your render times. There's a point of diminishing returns when it comes to samples - 5,000 doesn't look twice as good as 2,500. The trick is to find a balance:

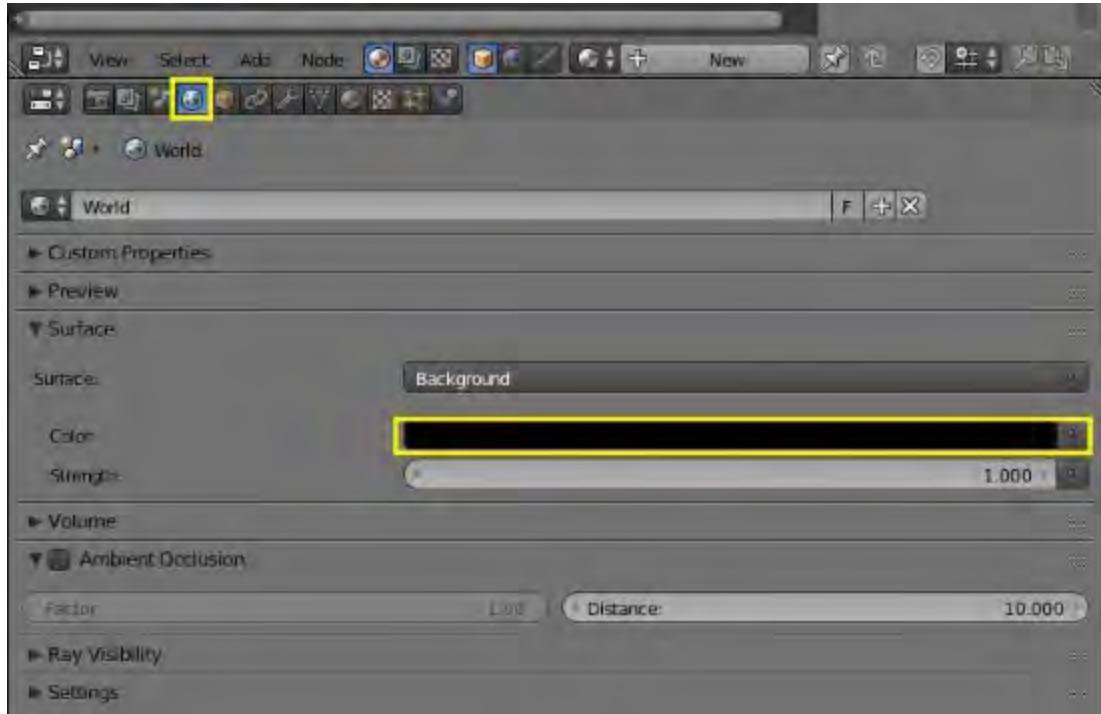


5 Samples

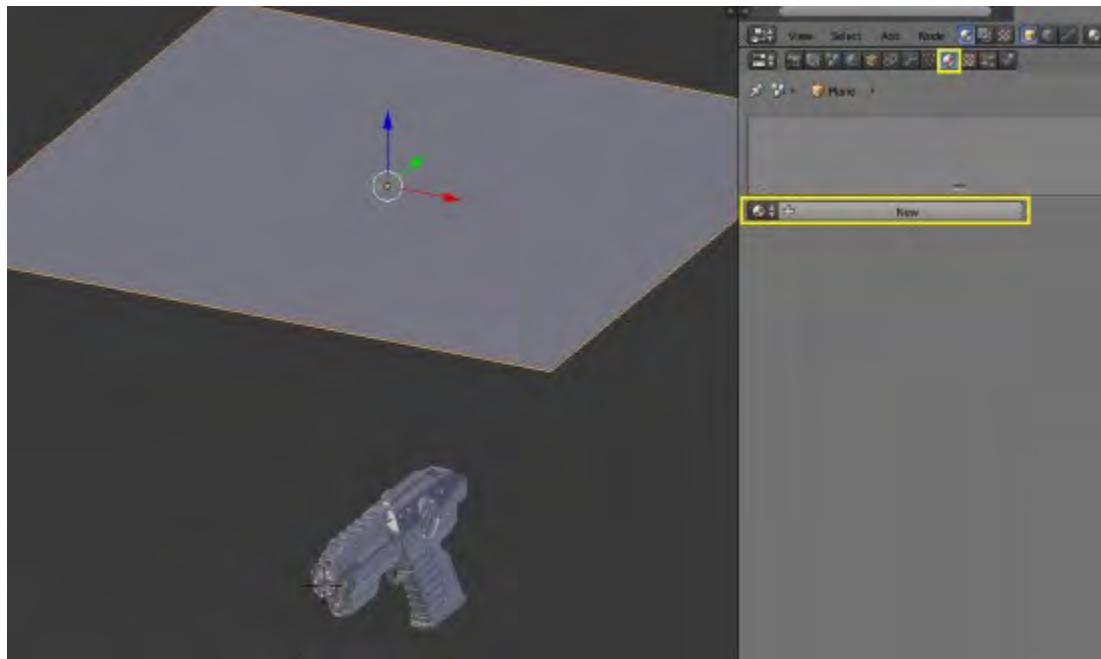


100 Samples

Next, let's go into our **World** tab and turn our ambient light all the way down. Do this by either changing the **Color** to black or the **Strength** to 0.



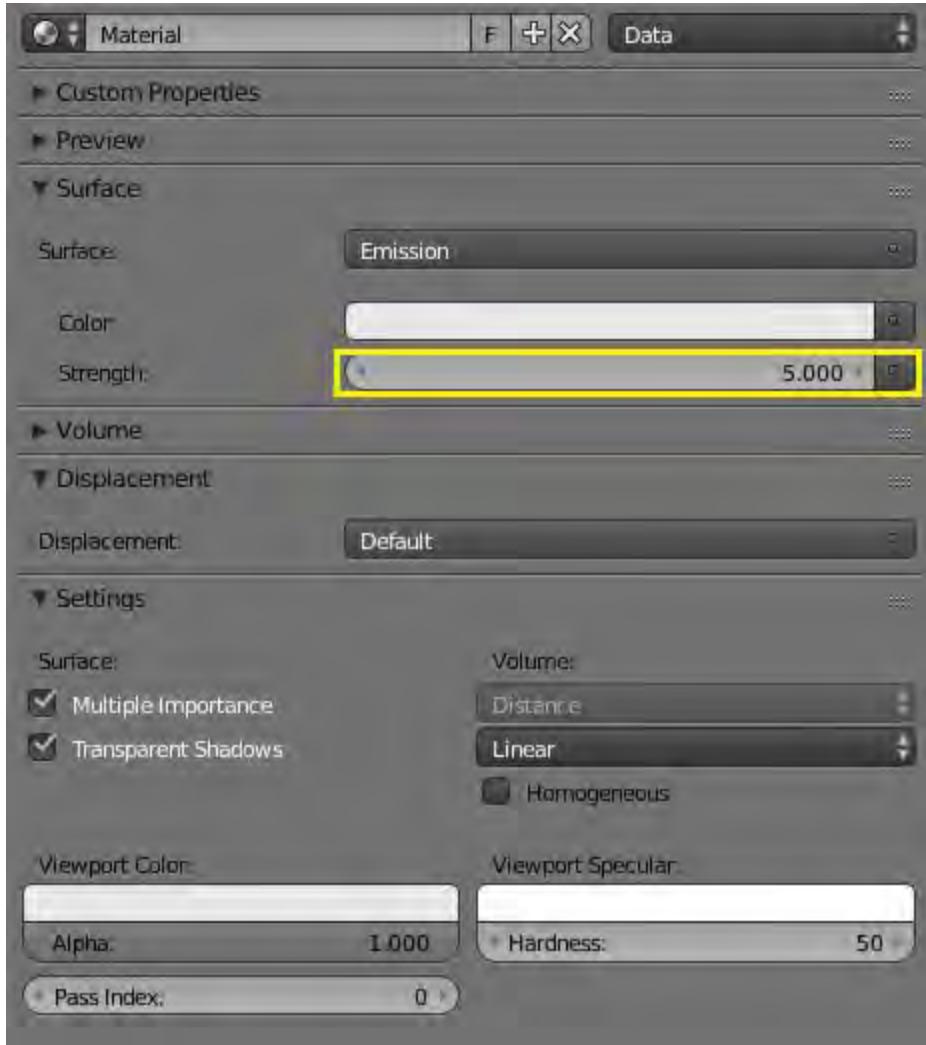
Next, we'll add a basic plane in our scene to function as the overhead light. Then we'll add a new material to it under the **Materials** tab:



We'll choose **Emission** for the material.



Setting the emission **Strength** to 5 is usually a good place to start. You can adjust it later as needed.



At the bottom of your screen, you can switch **Viewport Shading** to **Rendered** (you can also do this by pressing Shift + Z). This will give you an idea of what your scene will look like when you render it. It's also a great way to check materials and textures as you go along.

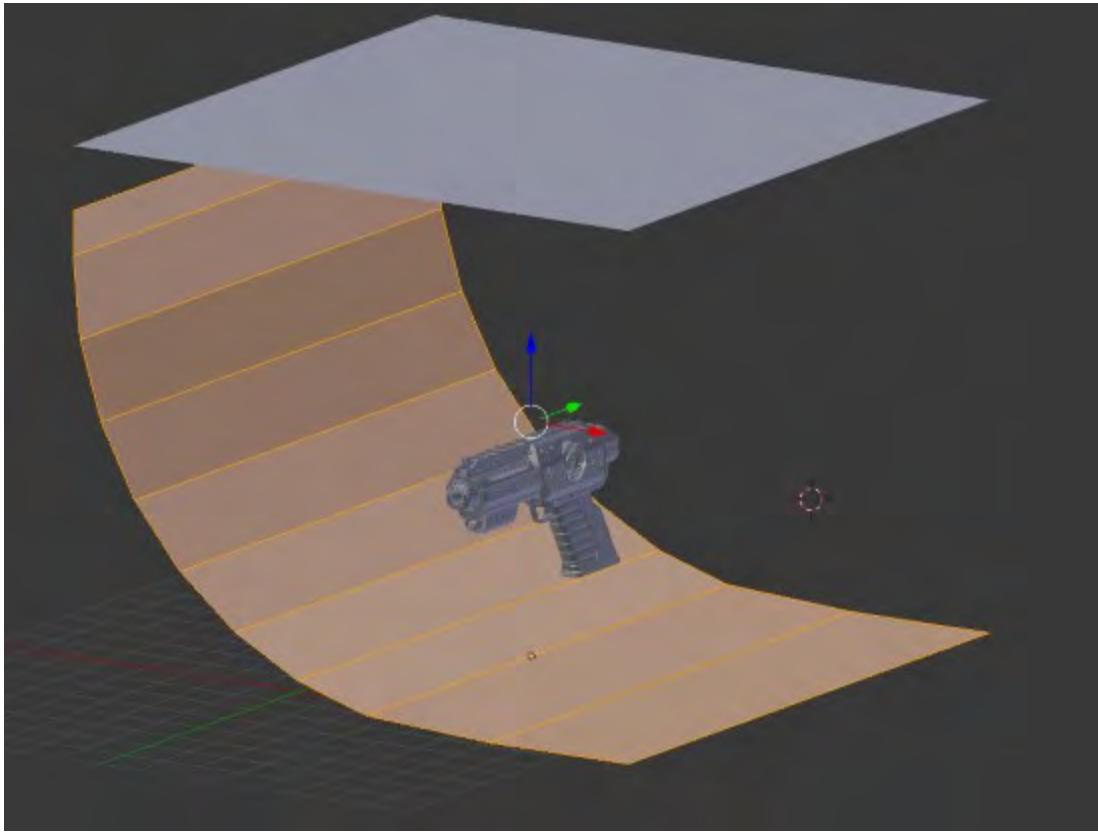
If you'd like to quickly switch back to **Solid Shading**, which is the default view, you can press the Z key twice or Alt + Z.

Note

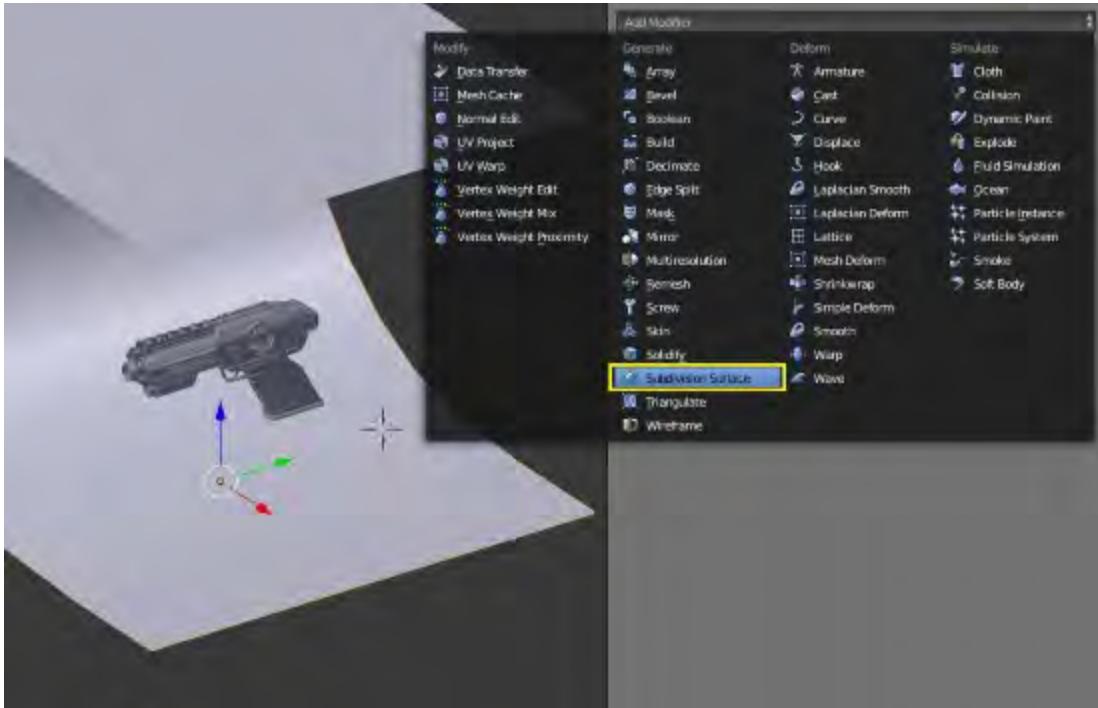
This only works to switch back to solid shading from something else.



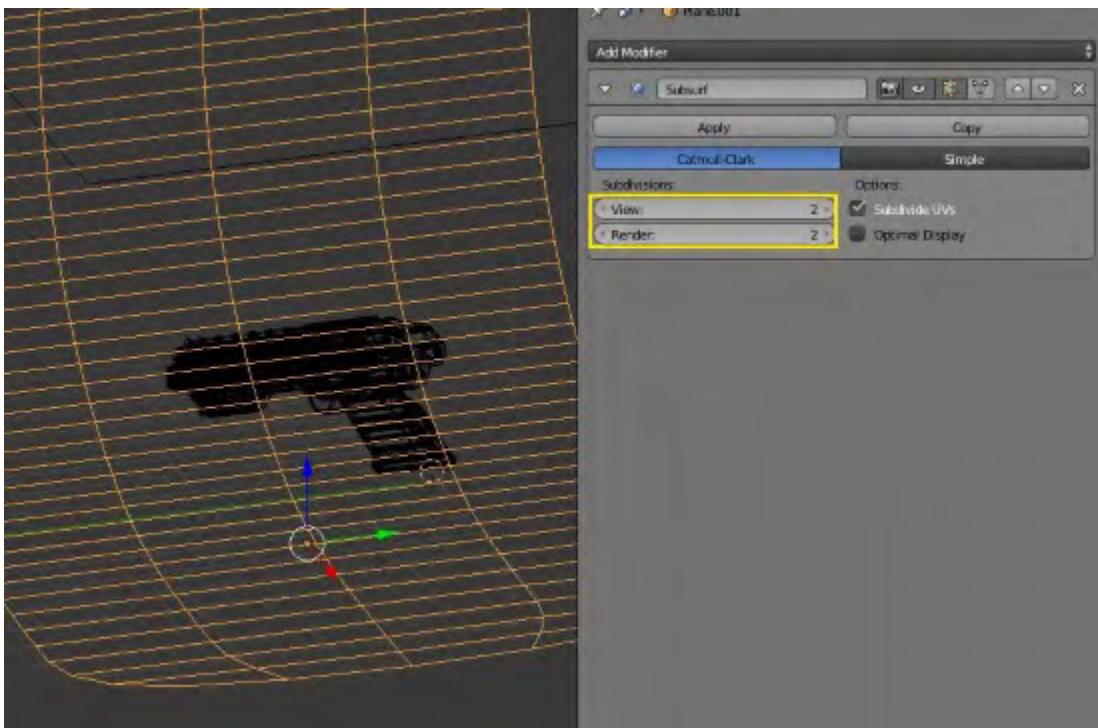
Next, we'll add a curved background plane. You can add this using any of the techniques that we covered in the previous chapter. It can be a portion of a cylinder, a part of a beveled cube, or so on.



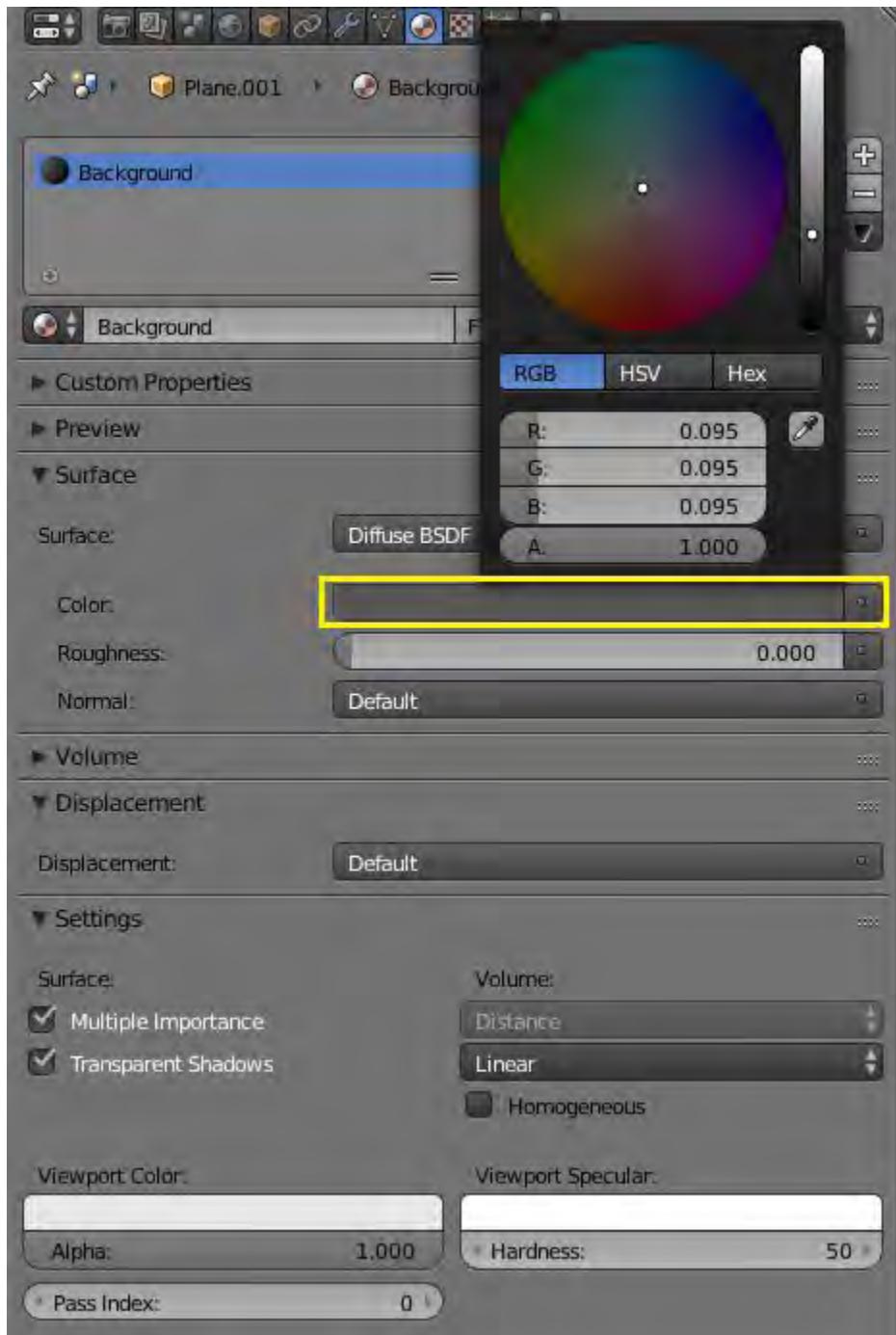
Let's quickly add a **Subdivision Surface** modifier to our background plane. We'll discuss the **Subdivision Surface** modifier in detail later, but it essentially adds geometry and smoothens out the shape of your object.



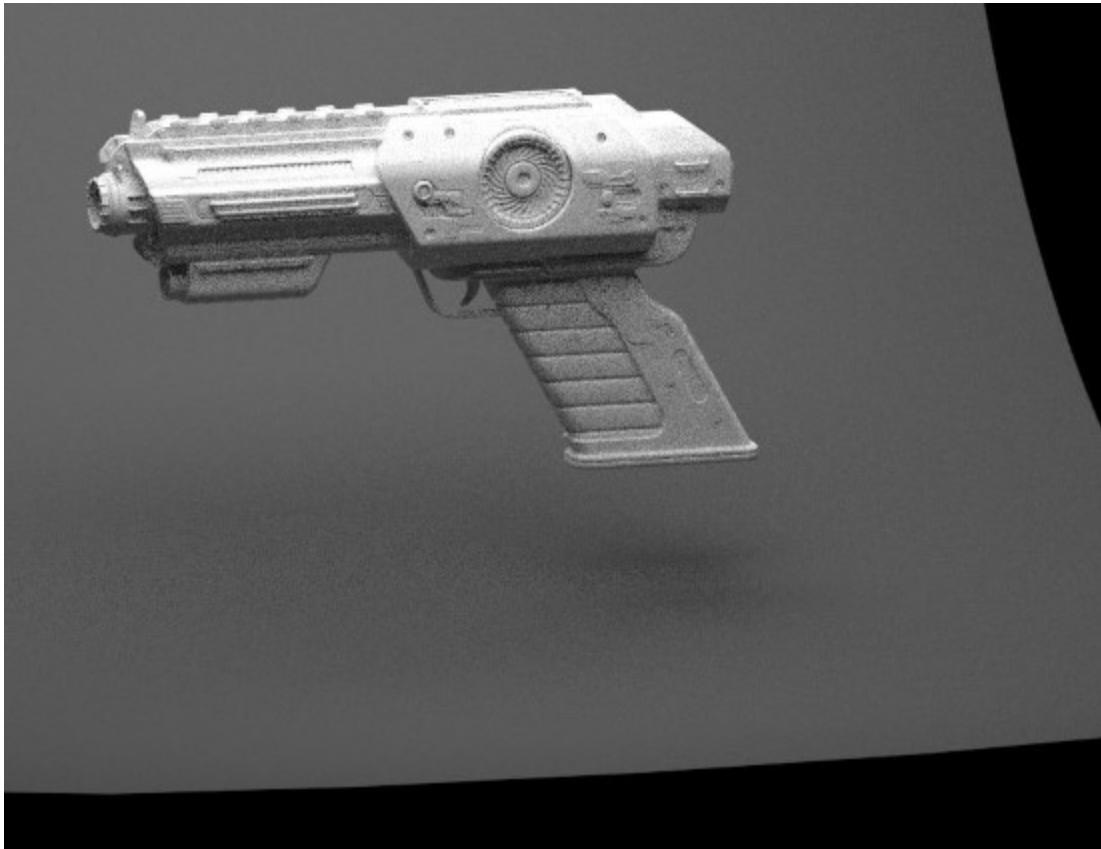
When you toggle into Wireframe mode (press the Z key once), you can see what the **Subdivision Surface** modifier has done for you.



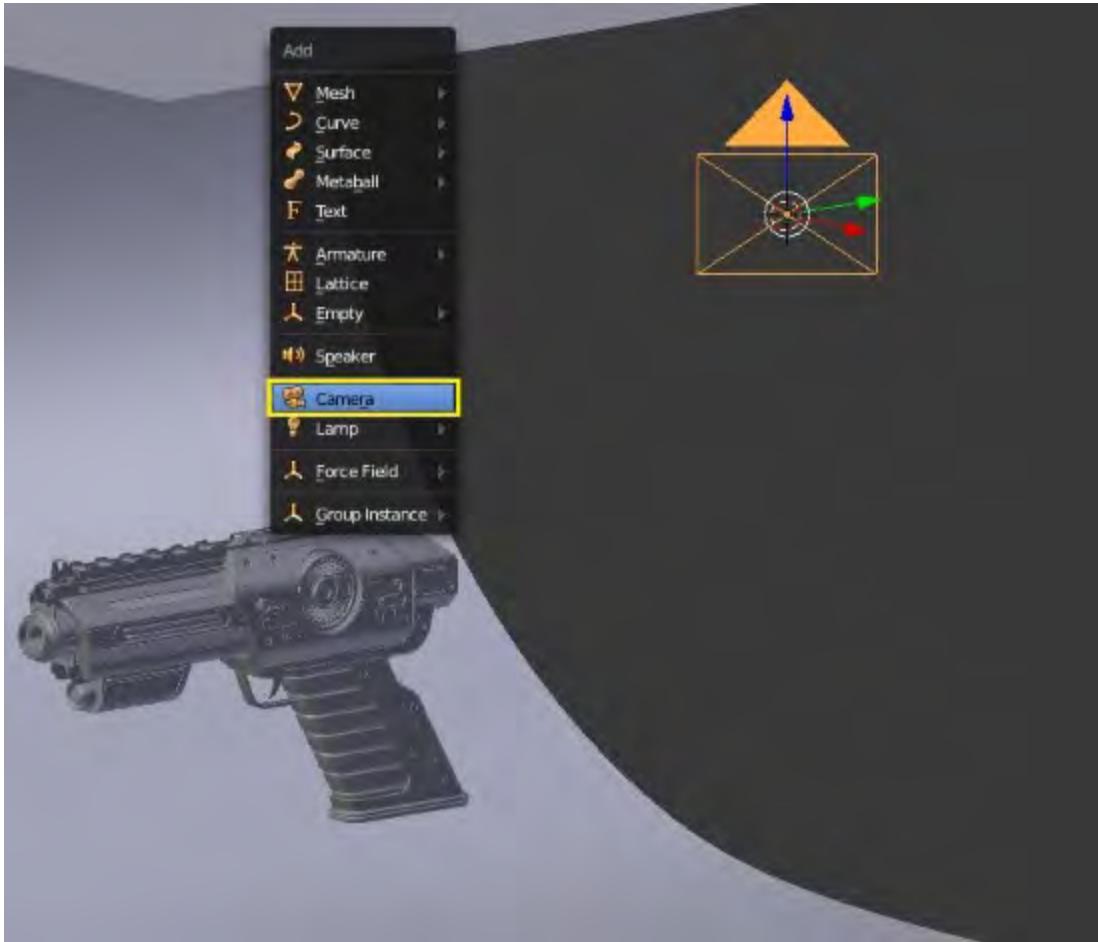
Let's add a basic **Diffuse** material to our background plane. A **Diffuse** material is essentially a flat (not shiny) material that gets its name by diffusing light. We'll take a look at this in a minute.



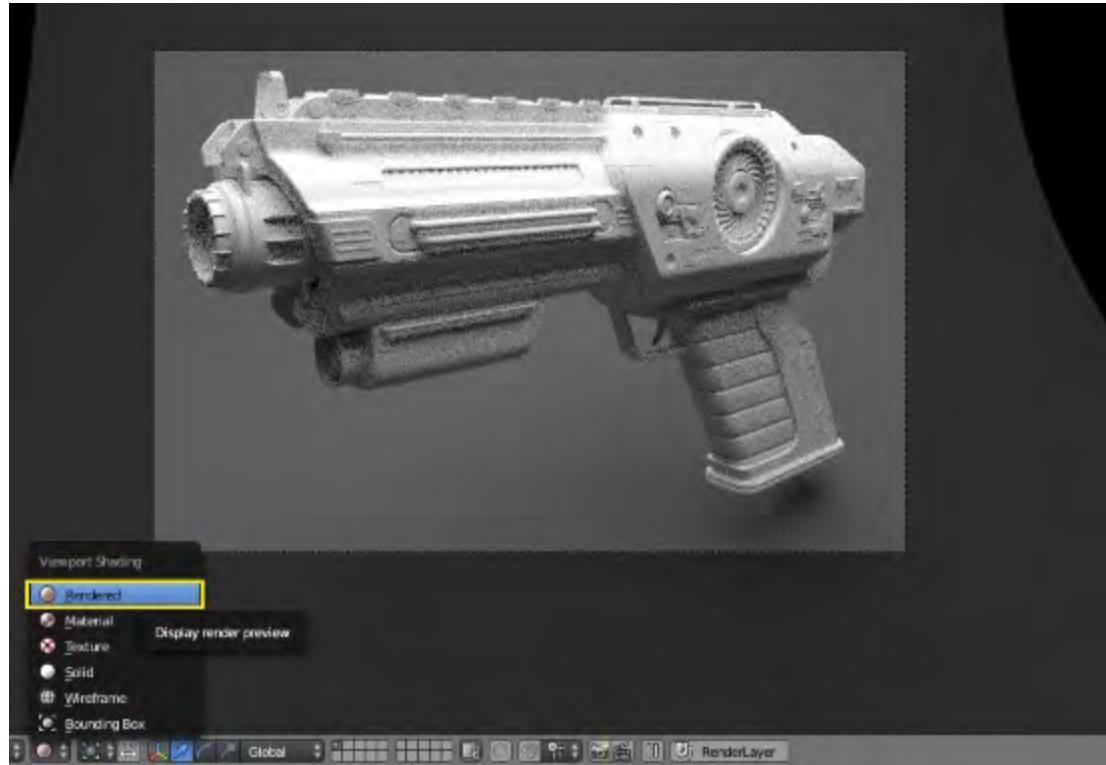
Take another look at our rendered preview to see how the background plane looks.



We'll also want to add a camera to our scene. Even though we don't want to render yet, it's a good idea to add this early. In this way, you can make sure that any object you add will be in the frame and in the right size/shape for later use.



Position the camera however you'd like. You can switch to **Camera View** by pressing numpad 0. Switch to **Rendered** view one more time, and you will get a good idea of what the render will look like now:



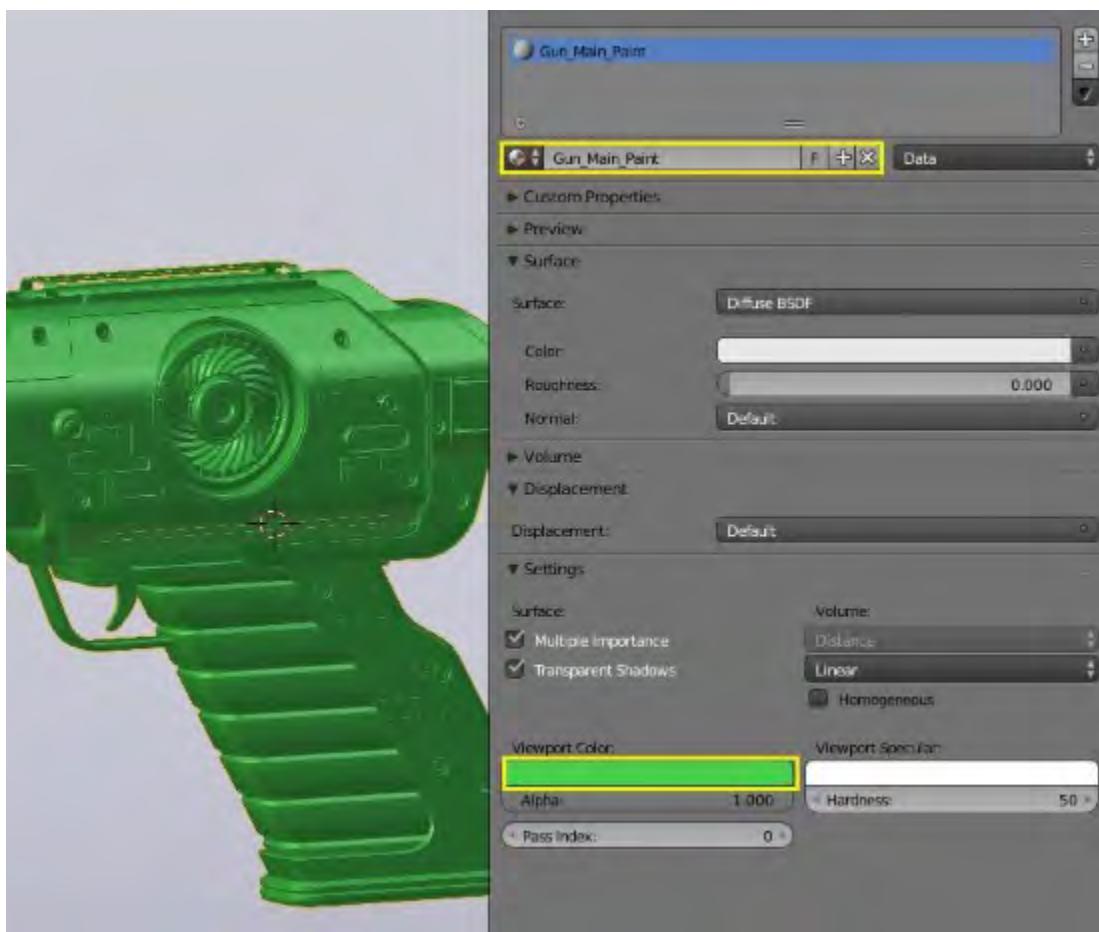
Adding materials and textures

Now, it's time to add some materials to our gun!

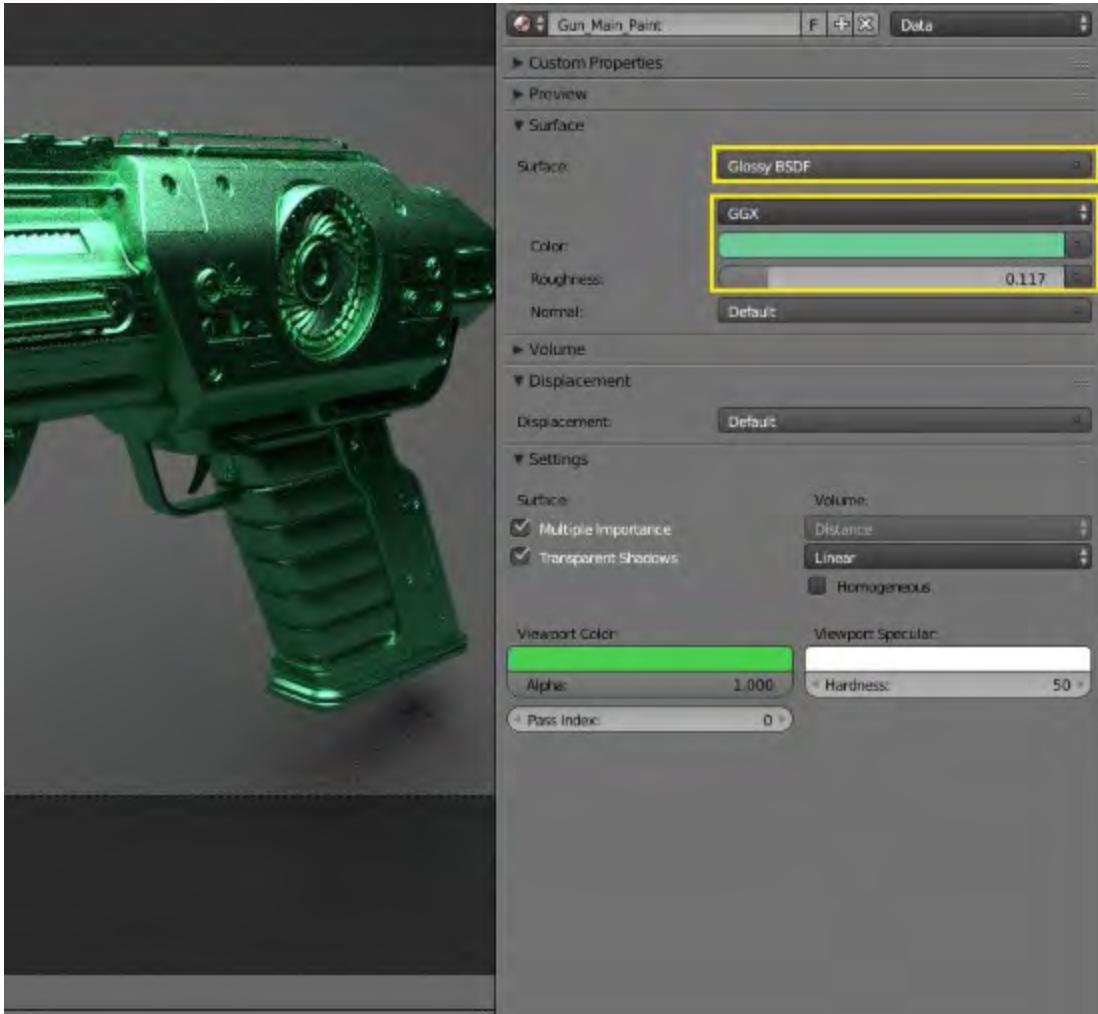
We'll add a basic one first and call it **Gun_Main_Paint**. One thing we want to do is adjust the **Viewport Color** setting so that we can see which materials are assigned to the various parts of our model. Because it's the only material on our gun right now, the entire gun will change its color to match what you've selected.

Note

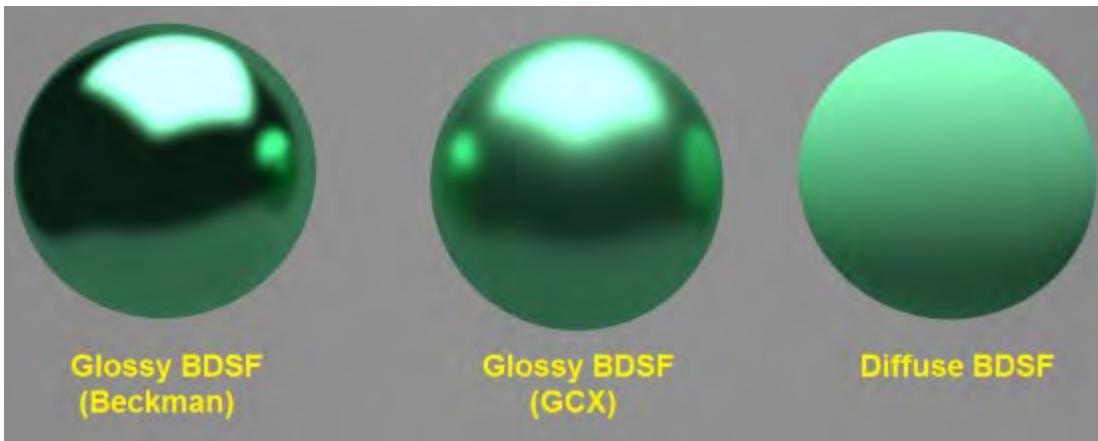
Changing the **Viewport Color** does not affect the actual color of the material (at the render time). It simply changes what shows up in your 3D window.



We'd like our gun to be a little shiny, so let's switch from **Diffuse Shading** to **Glossy Shading** in our materials tab. Using the **Rendered** preview, you can adjust the **Roughness** and **Color** of the material until it looks the way you like.



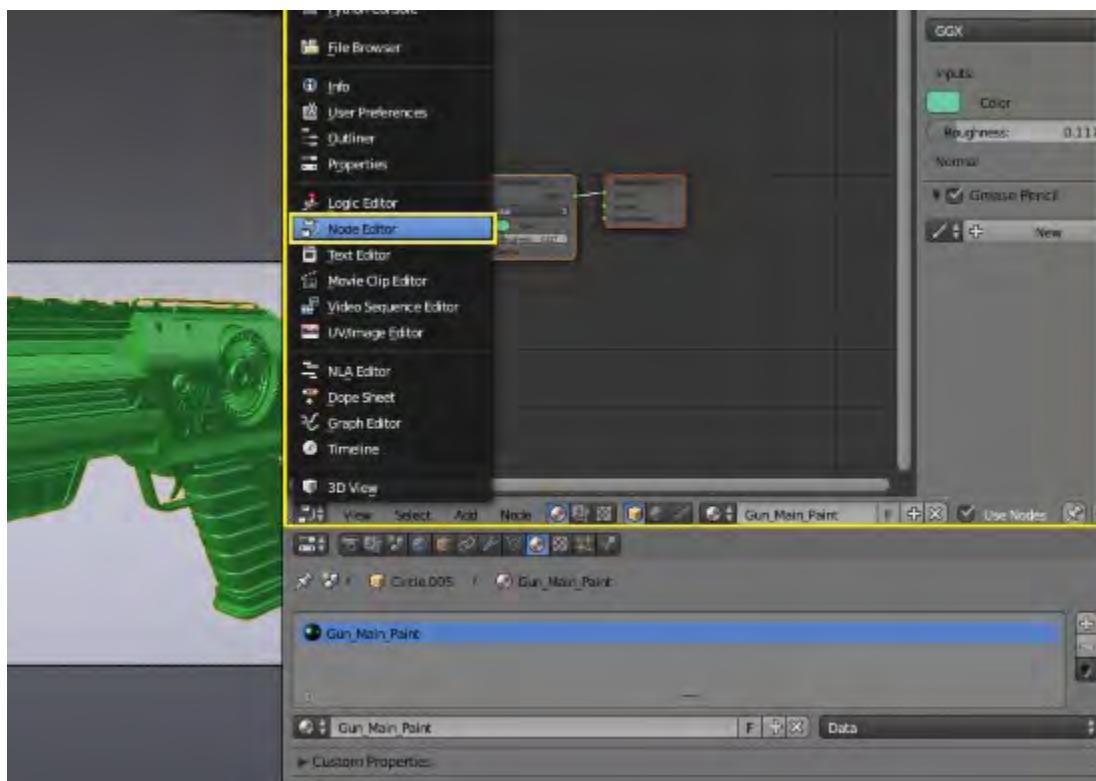
You also have the option to switch between the **Beckman**, **GCX**, **Sharp**, and **Ashikhmin-Shirley** types of shading. Each of these reflects light somewhat differently and you must feel free to experiment. For this model, however, we'll probably only need Beckman and/or GCX shading.



So now we have a glossy material on our gun. However, any real material is going to have more than one type of shading to it. For instance, almost everything in the real world has some **Diffuse** (flat) as well as some **Glossy** shading to it.

Even the shiniest object has a bit of flat shading to it and even the flattest object has a tiny bit of gloss to it. So, let's adjust our gun's material to reflect this.

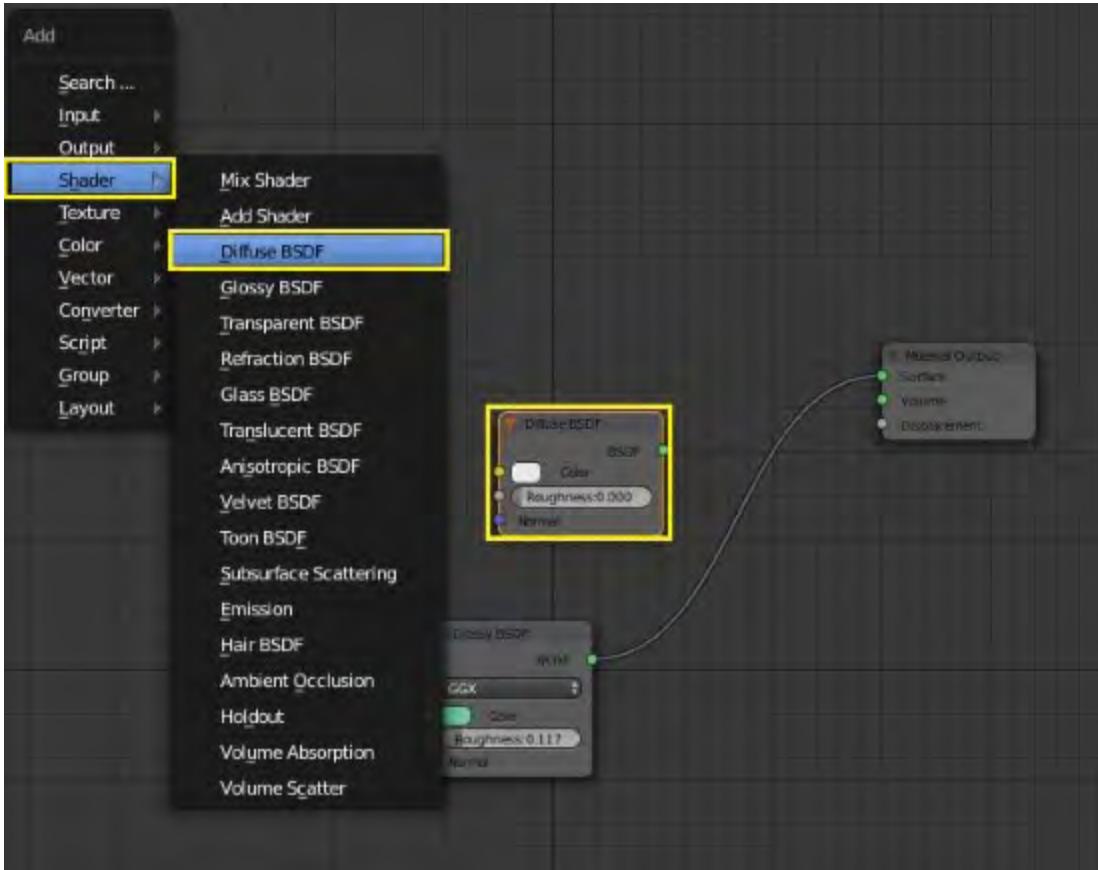
We'll open up **Node Editor** in Blender. It's easiest if you have it set up right above your **Properties panel** and next to your main 3D window. Of course, this is really a personal preference.



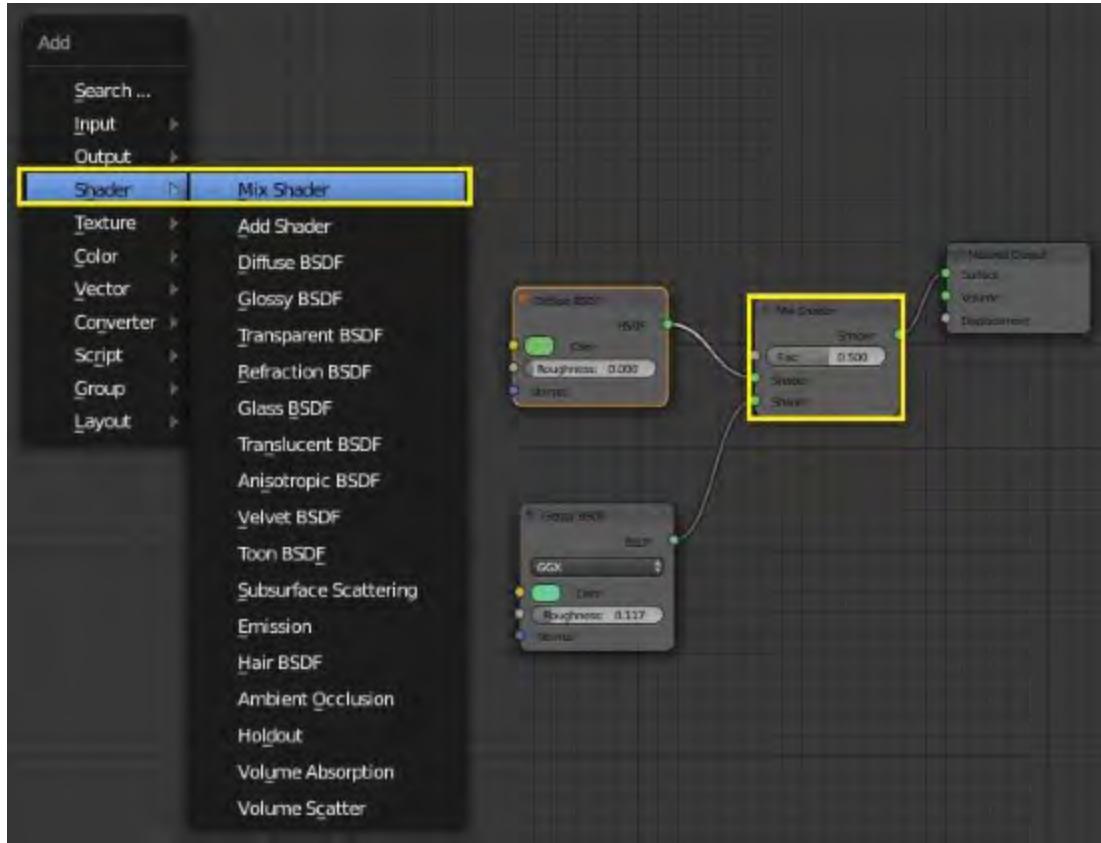
By looking at our **Node Editor** window, we can see that the **Glossy** shader that we selected is connected directly to the material's **Surface Output**.

This is about as simple as materials get in Blender and it doesn't actually look too bad. However, we'll want to improve it a bit.

First, let's add a **Diffuse** shader.



You can adjust the color of that shader to match the glossy shader that we already have. In order to mix these two, we'll next need to add a **Mix Shader**.



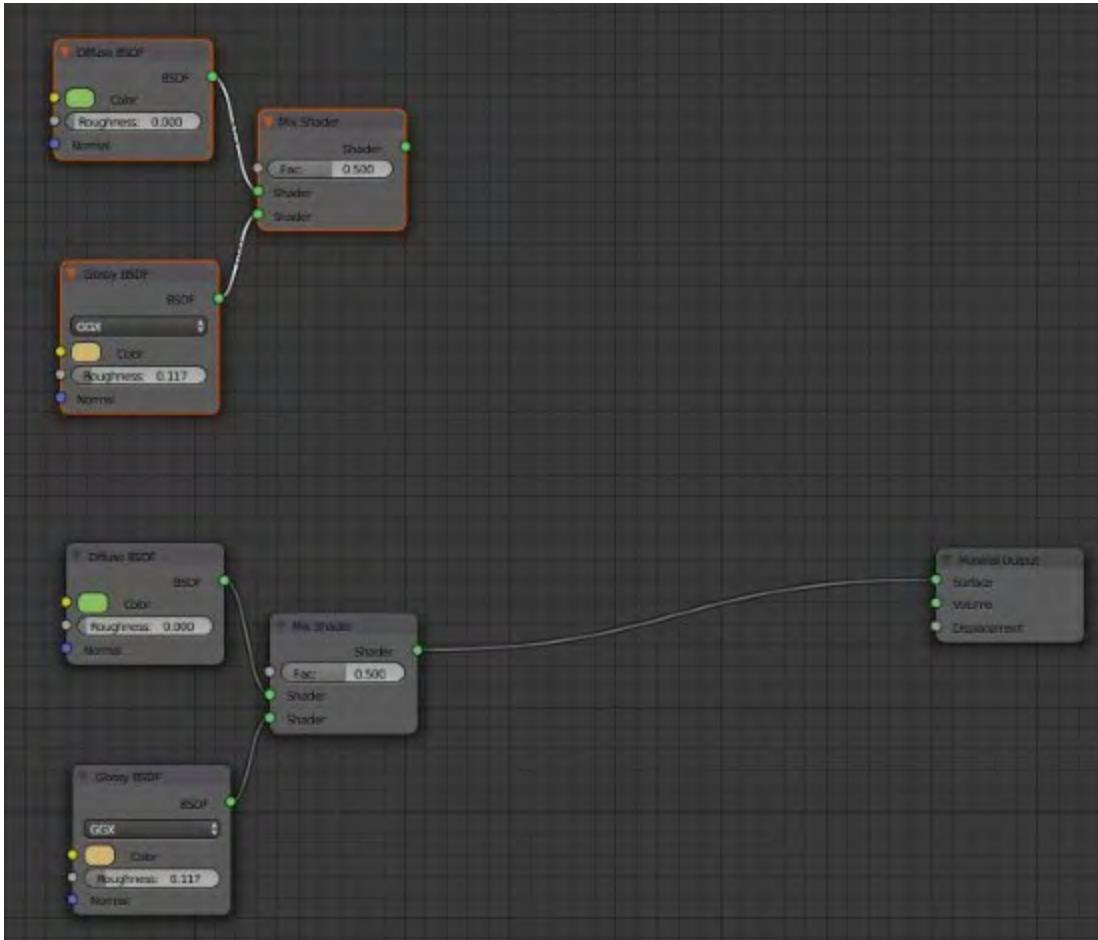
Connect the outputs of both the **Glossy** and **Diffuse** shaders to the **Mix Shader**. Then, connect the **Mix Shader** output to the **Surface Slot** of the **Material Output**.

When you go back to the rendered view now, you can see that the gun's material is a combination of both diffuse and glossy shading. You can change all kinds of settings here (color, roughness, type of shading, and mixing factor) until you get a material that you like.

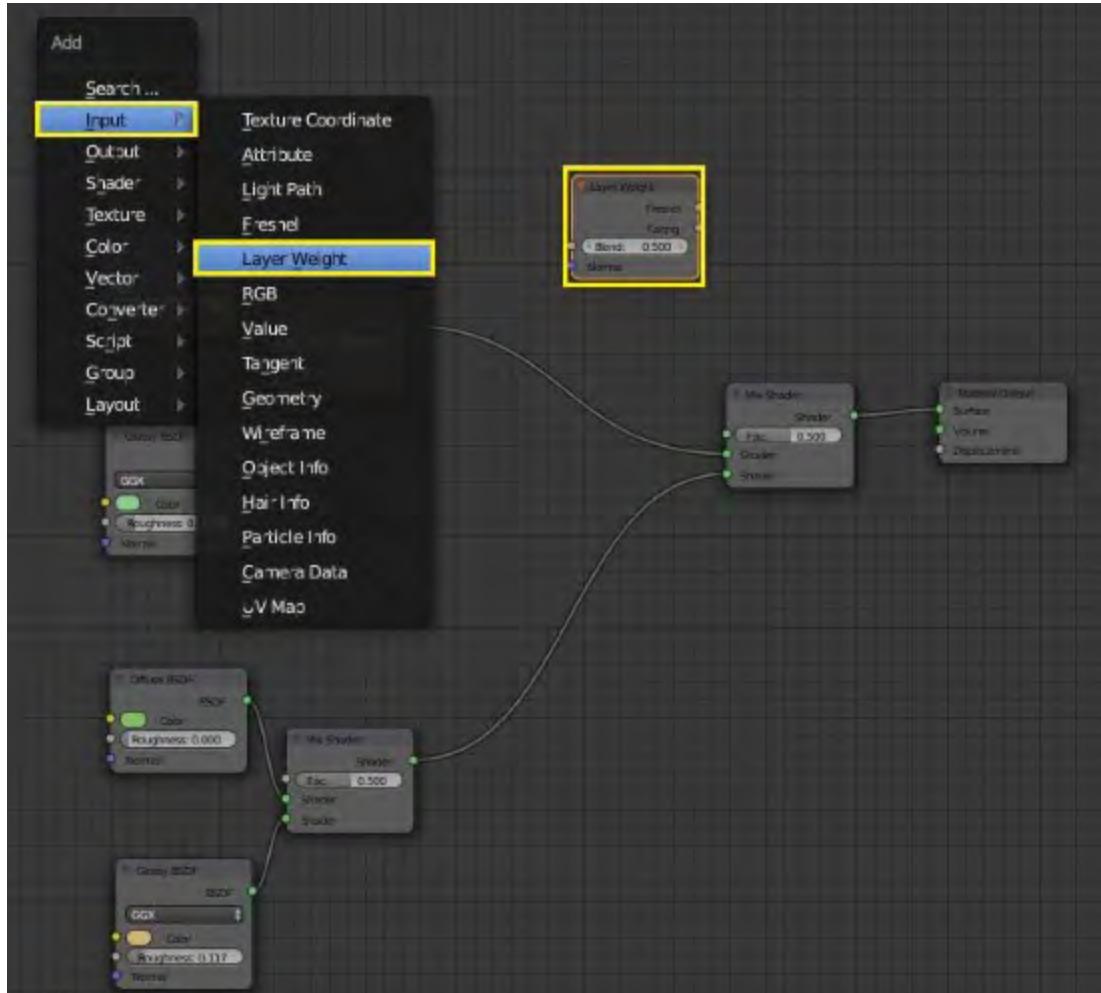


Another cool thing we can do is to create a color-change effect as light moves across the object. I think this will look pretty cool, especially for a sci-fi pistol. In order to do this, we'll first need to duplicate the setup that we already had. Just like in the 3D view, you can grab a set of nodes and hit Shift + D to duplicate them.

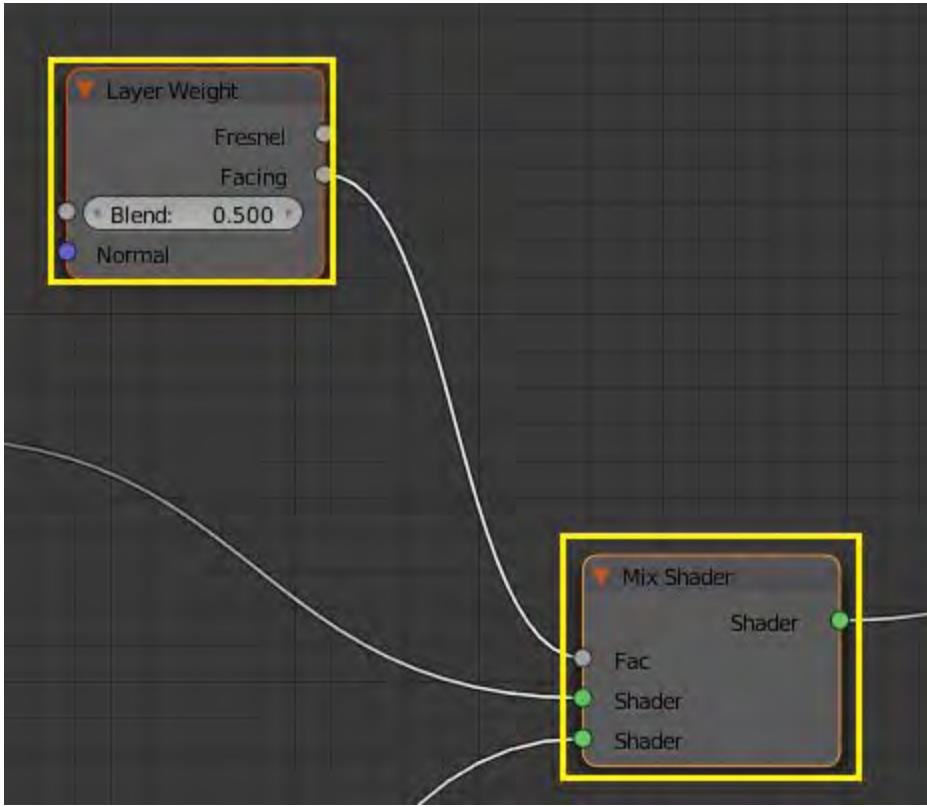
Go ahead and make a few changes to your duplicate set of nodes. Change the color or roughness a bit. We just want to make it a bit different from the first set.



We'll need another **Mix Shader** here to combine our two sets of nodes. Then, we'll add a **Layer Weight** input.



Connect the **Facing** tab of the **Layer Weight** input to the **Fac** (Factor) input of the **Mix Shader**.

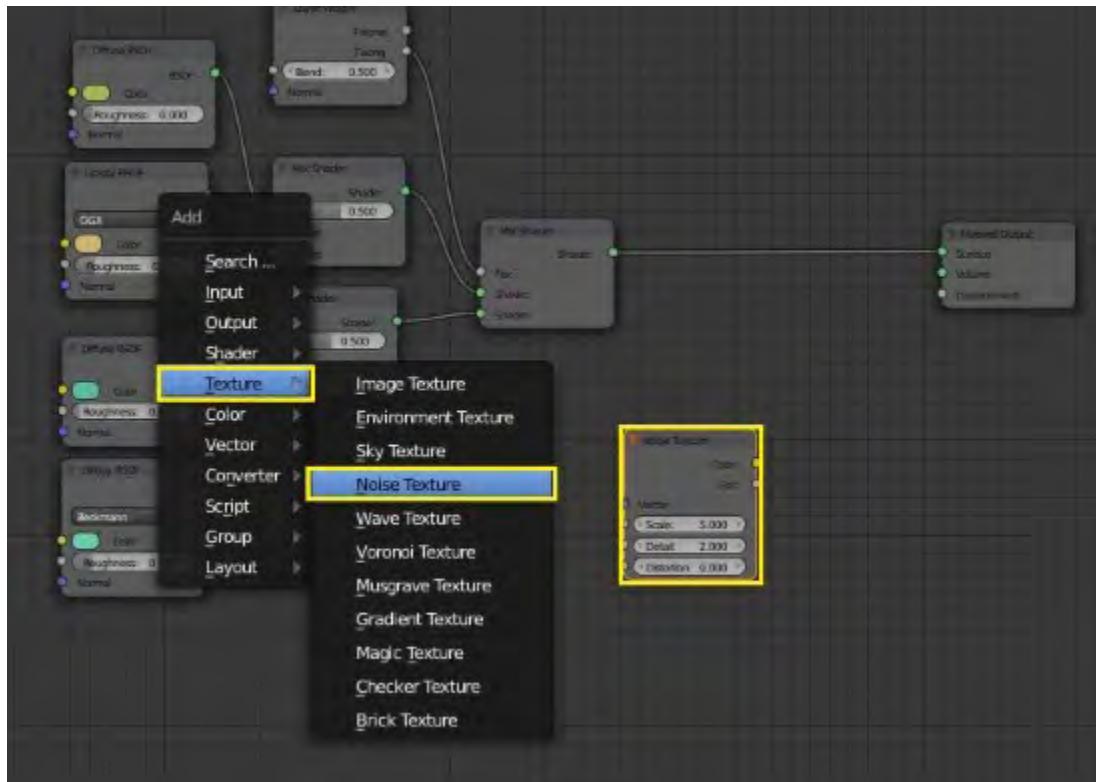


Since we're using solid (metal/ plastic) types of materials, it's best to use the **Facing** output of the Layer Weight shader. If you have a material where light penetrates the surface (such as glass or translucent plastic), you can also experiment with the Fresnel tab. For now, we'll stick with **Facing**.

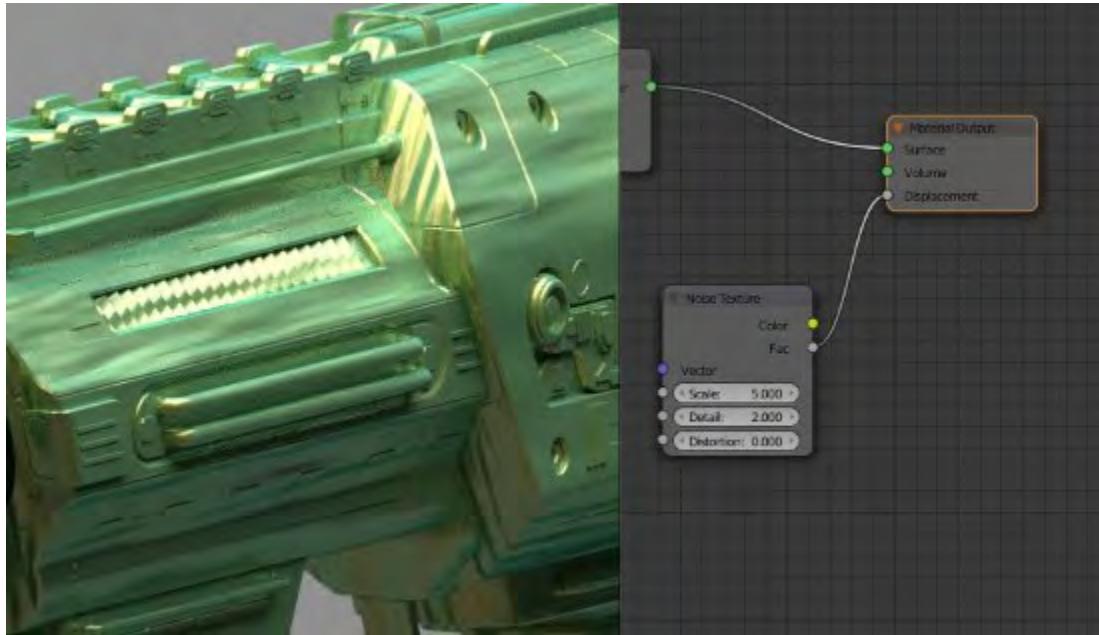
Now our gun has a nice fade between its different colors.



Right now, the surface of the gun appears perfectly flat. However, it'd be nice to add a rough metal texture to it. In order to do this, we're going to add a **Noise Texture** node.

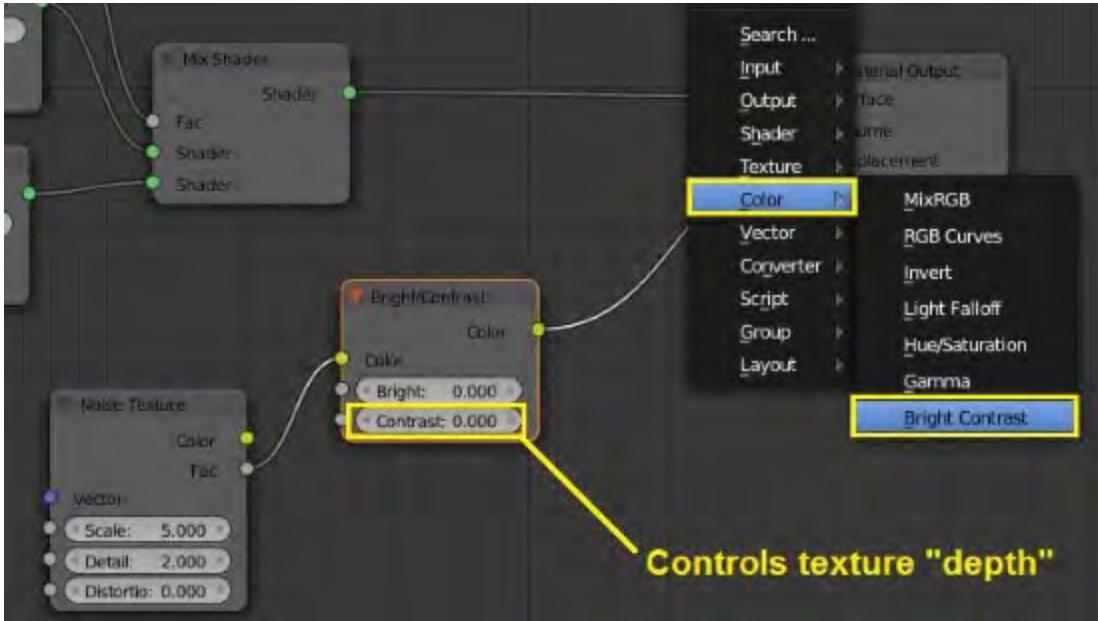


When you plug this node into the **Displacement** input of our **Material Output** node, you can see that the surface of the model is affected:



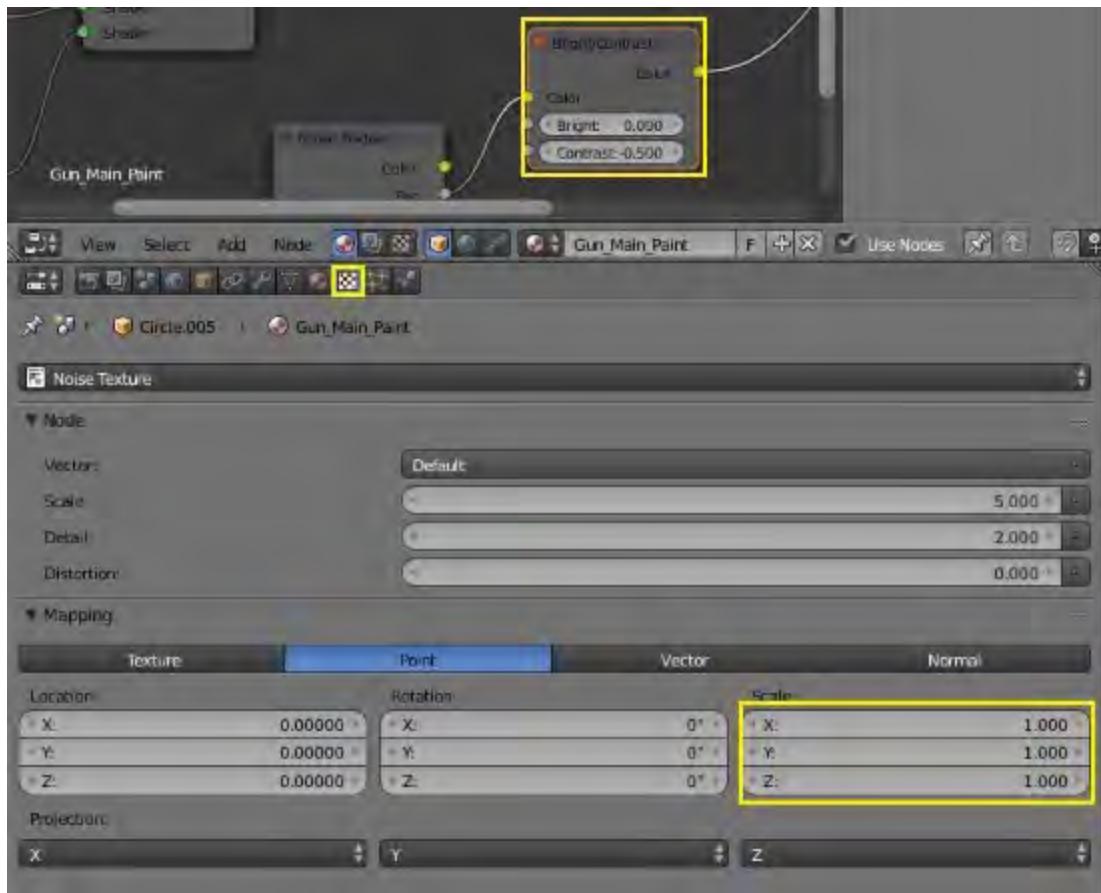
It appears to have some hills and valleys to it. Note that this does not change the mesh at all; it's just a material's effect.

If you feel that the effect is too strong, you can add a **Bright/Contrast** node between the **Noise Texture** and **Material Output** node. By decreasing the contrast, you change the apparent height/depth of the texture.

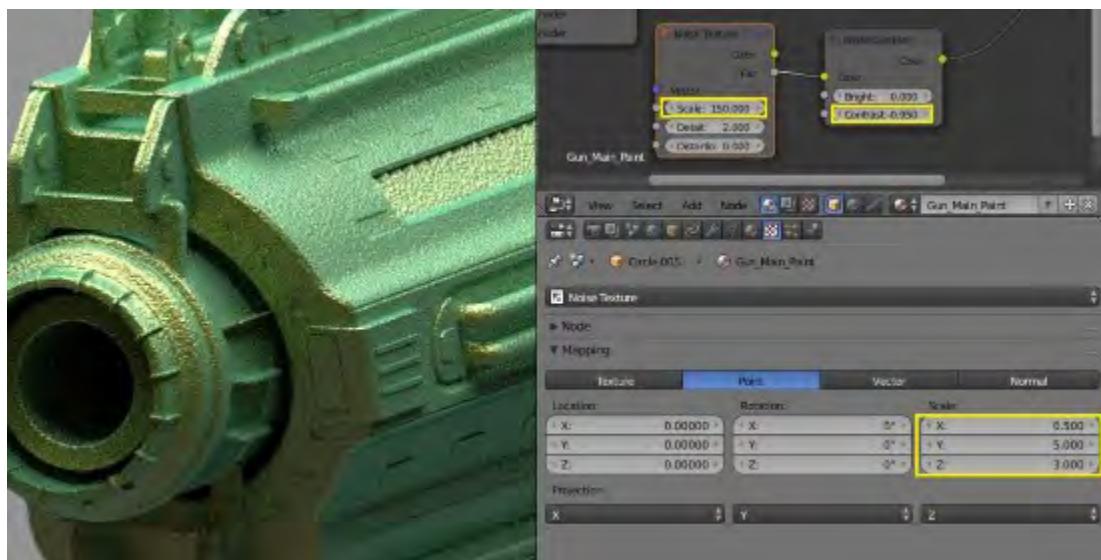


We also need to change the **Mapping** of the texture. Because our gun is longer than it is tall, (and taller than it is wide), the texture appears stretched. In order to fix this, we'll select the **Texture** node and move it to the **Textures** tab of our **Properties** panel.

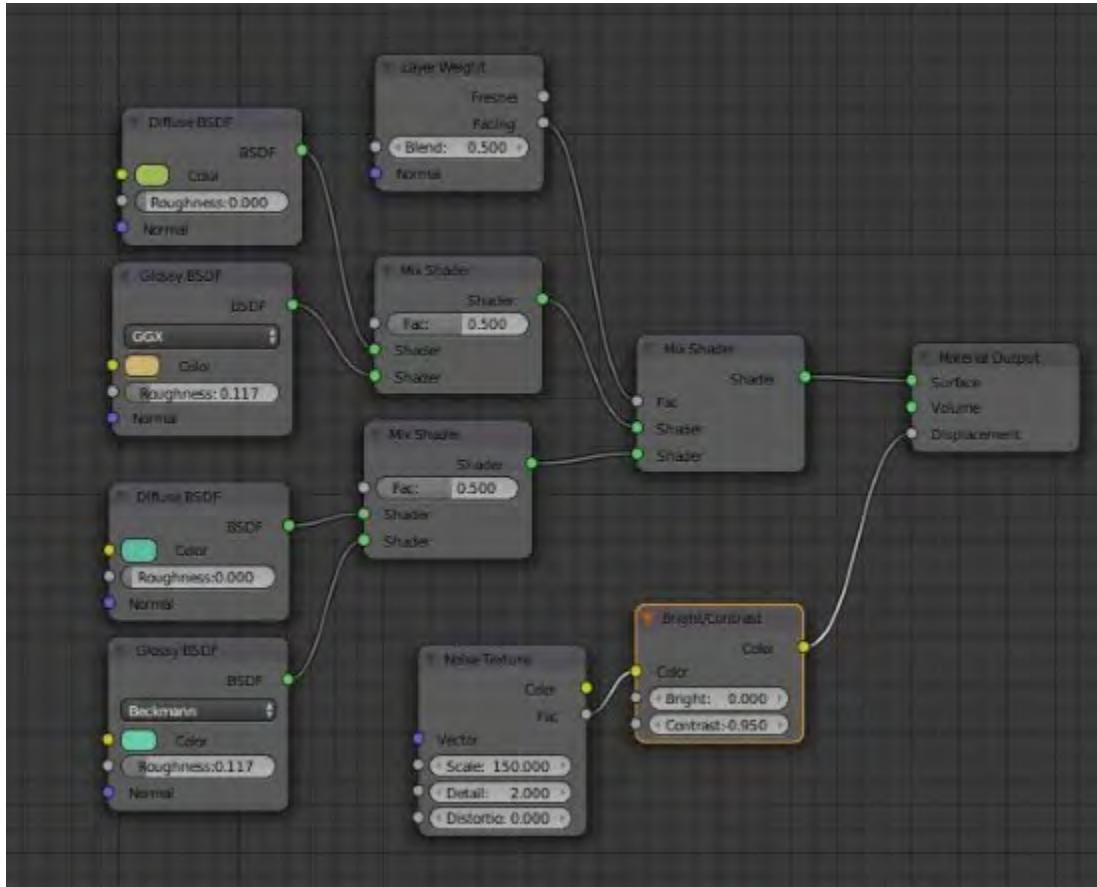
Here, you can see that the texture is being mapped to the gun evenly on all three axes. Let's adjust these numbers to reflect the different length, width, and height of our gun.



You can also adjust the overall scale of the texture. This is done directly in the **Noise Texture** node. By changing it, you can make the texture appear larger or smaller. You can adjust this as needed until you get the desired effect.

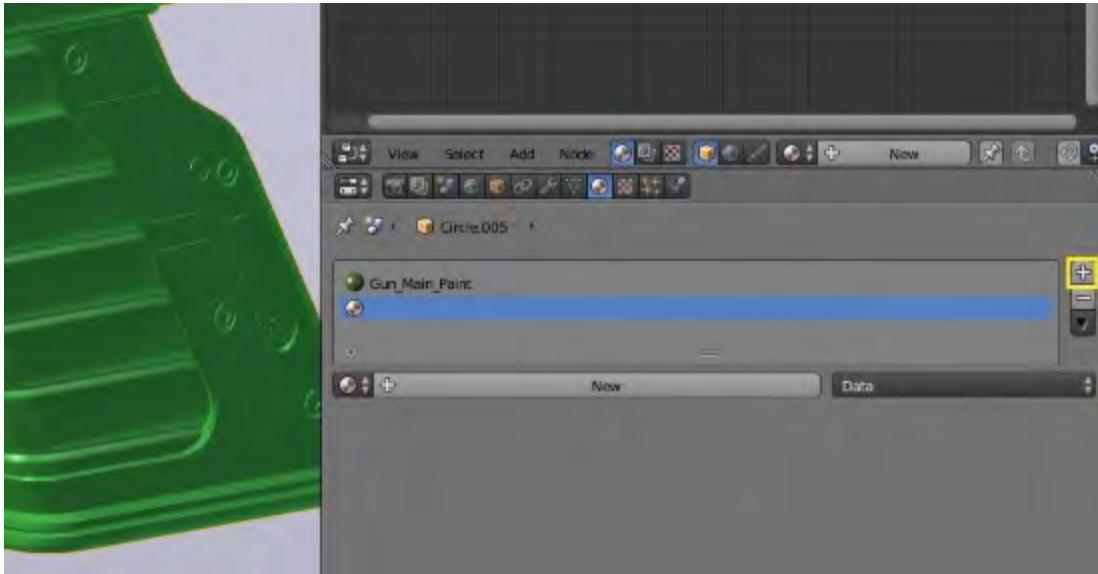


This looks pretty good for our basic paint material. Before we move on, here's what my material looks like in the **Node** editor:

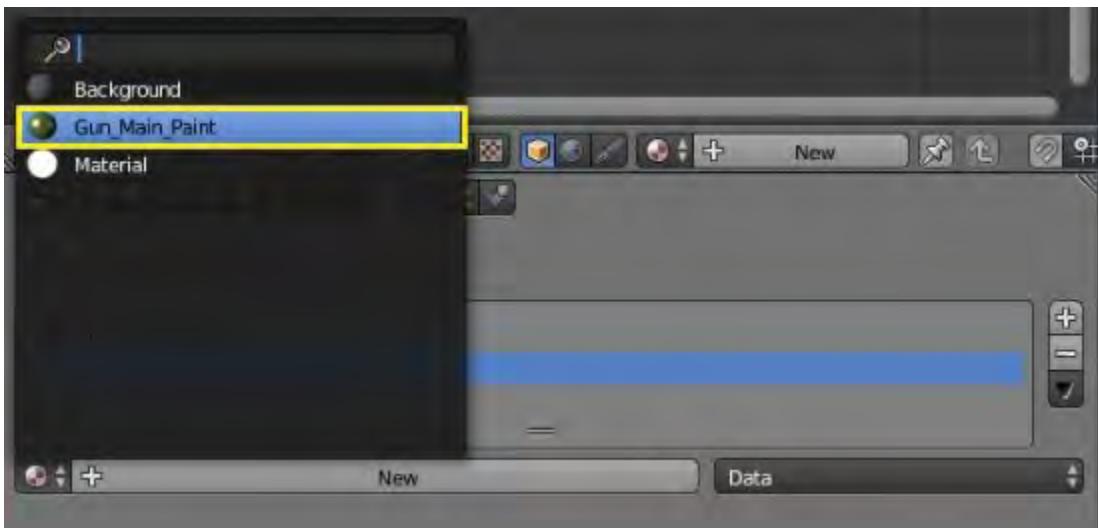


You can feel free to use these exact settings, or you can adjust them until they look good to you.

Next, we'll add a second material to our gun. This will look very similar to the first material, so we'll duplicate it and make a few adjustments. In order to do so, first add a new material to the gun with the small plus tab next to the materials list.



Then, select the **Gun_Main_Paint** material that we already have in place.

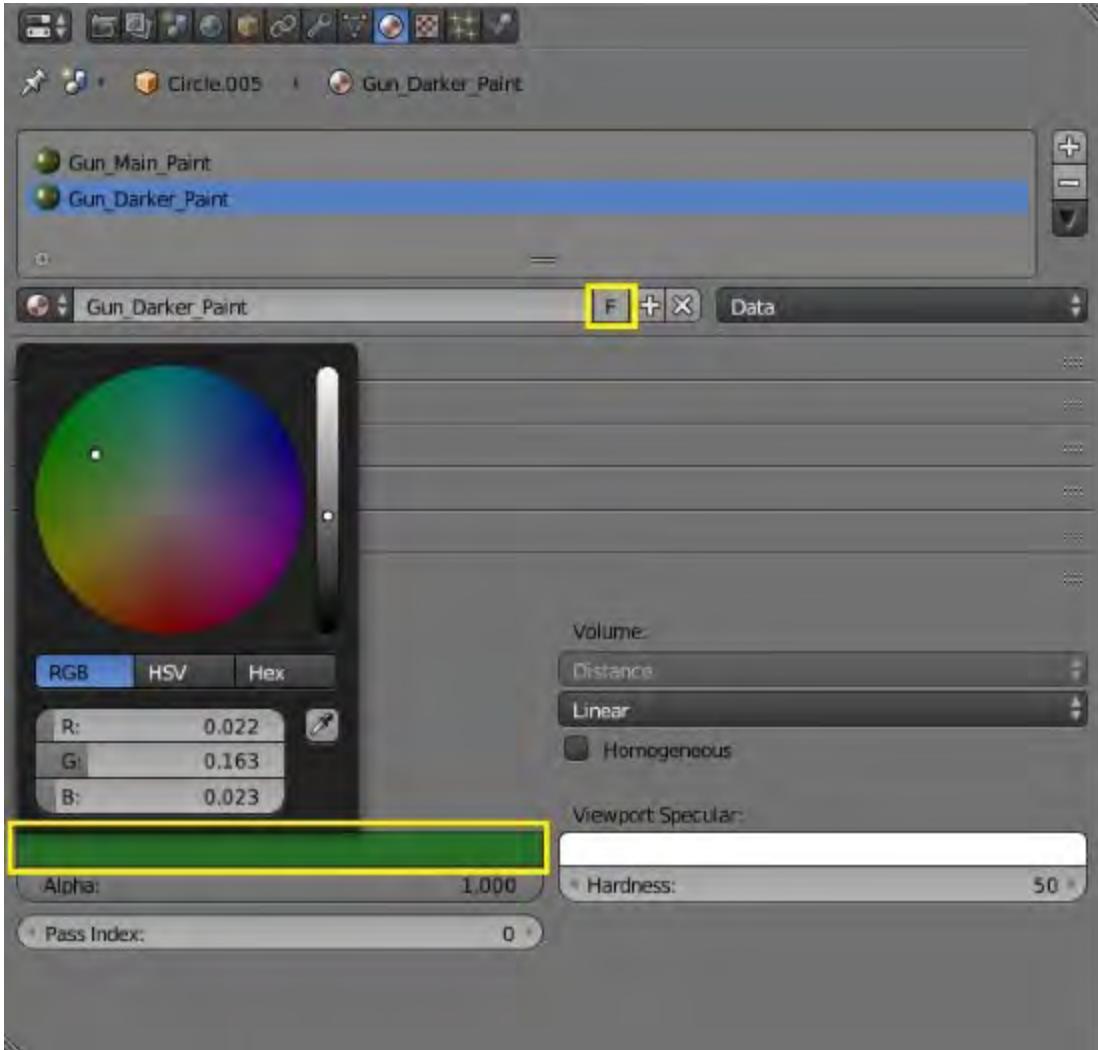


Because it's being used twice, you can see that the number **2** appears just below the materials list. We need to click on this to make it a **single-user**. In this way, we can adjust our second material without affecting the first one (right now, we just have two copies of the same one).

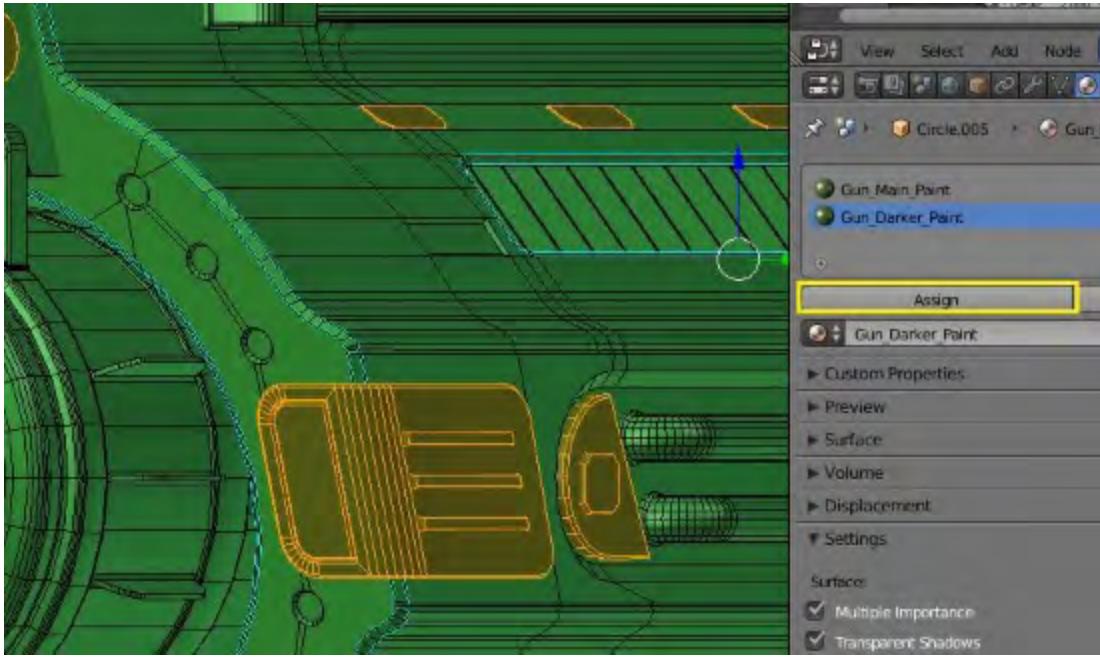


When you click this button, the letter **F** appears. You've now created a second material that's an exact copy of the first one.

In order to distinguish between the two materials in our 3D window, let's change the appearance of the second material. You can also change its name to anything you'd like.



Next, it's time to assign our second material to the various parts of the gun. Tab into the **Edit** mode on your gun and select the pieces/parts that you'd like to use the new material on. This is really an artistic decision, so I'll leave it to you!



Once you're done assigning the second material, you can go back to your **Node Editor** and make changes in the material. You can adjust the colors, brightness, roughness, mix settings, and whatever else you'd like. Personally, I prefer to keep my materials pretty close to each other.

You can see here that some parts of the gun have a slightly different color than others.



You can repeat the process as many times as you'd like, adding a variety of different materials to your weapon. Don't forget to make something different for the handgrip—perhaps a largely diffuse material that simulates rubber... it's your choice.

When you've got all of your materials the way you'd like, simply press F12 to render your scene.



Feel free to go back and tweak your different materials, adding more, fewer, or different ones as needed.

There are plenty of other things that we could do here. We haven't added any damage or wear to the metal, nor have we added any paint discoloration. Our render setup is very basic, and we haven't touched the compositing function of Blender at all. As we move on to other projects, we'll explore more and more options for materials, texturing, and rendering within Blender. Feel free to come back later and apply some of these techniques to this project.

Summary

This pretty much wraps up our sci-fi pistol! This was a pretty basic project, but we've actually covered a number of different topics while creating it. Now that we've covered these basic skills, we can move on to something a bit more complex in the next chapter —a sci-fi spaceship.

Chapter 4. Spacecraft – Creating the Basic Shapes

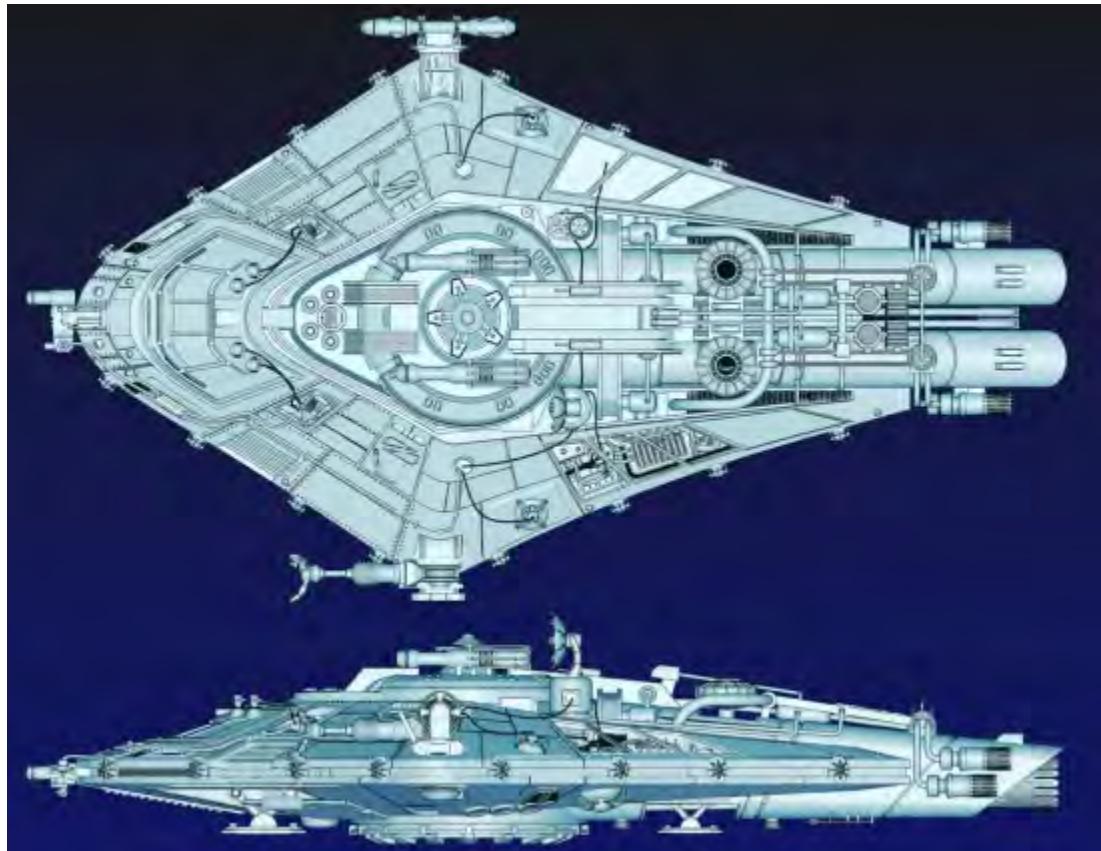
In this chapter, we'll get started on a detailed spacecraft. We'll lay a good foundation here by reviewing our goal, and then creating the basic shapes. That way, we can take it further in subsequent chapters. The following topics are covered in this chapter:

- Shaping the body
- Creating the accessories

Shaping the body

For our next project, we're going to build a futuristic shuttlecraft. The ship is fairly complex, but becomes easier when you break it down into distinct subsections. In this chapter, we'll work on getting the major shapes and components blocked out. It will start to look a lot less daunting by the time we're done!

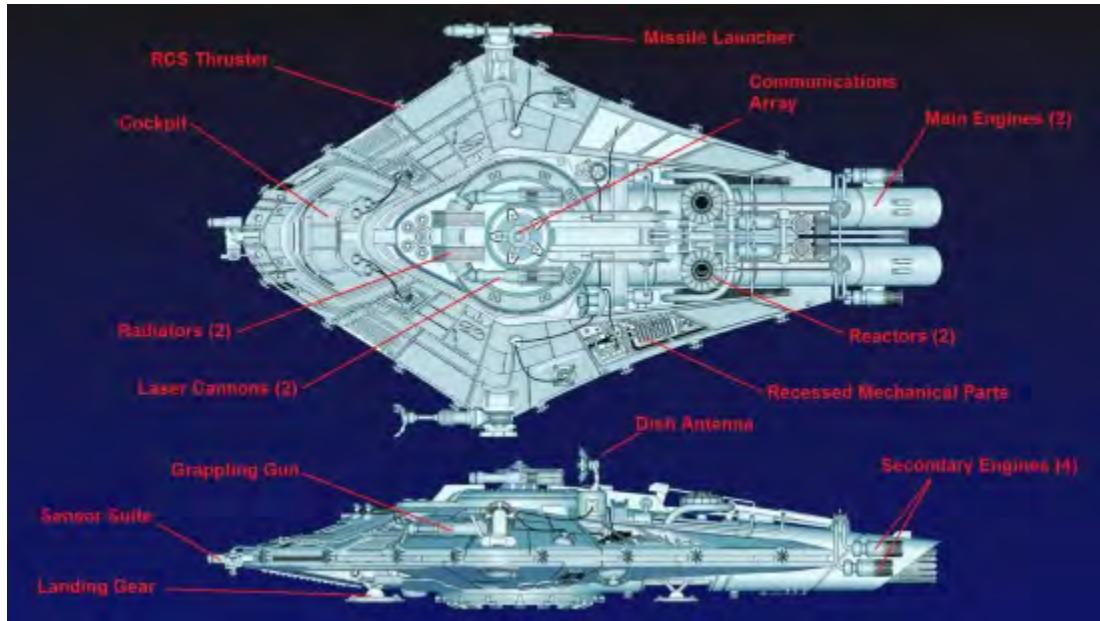
First, let's take a look at our design:



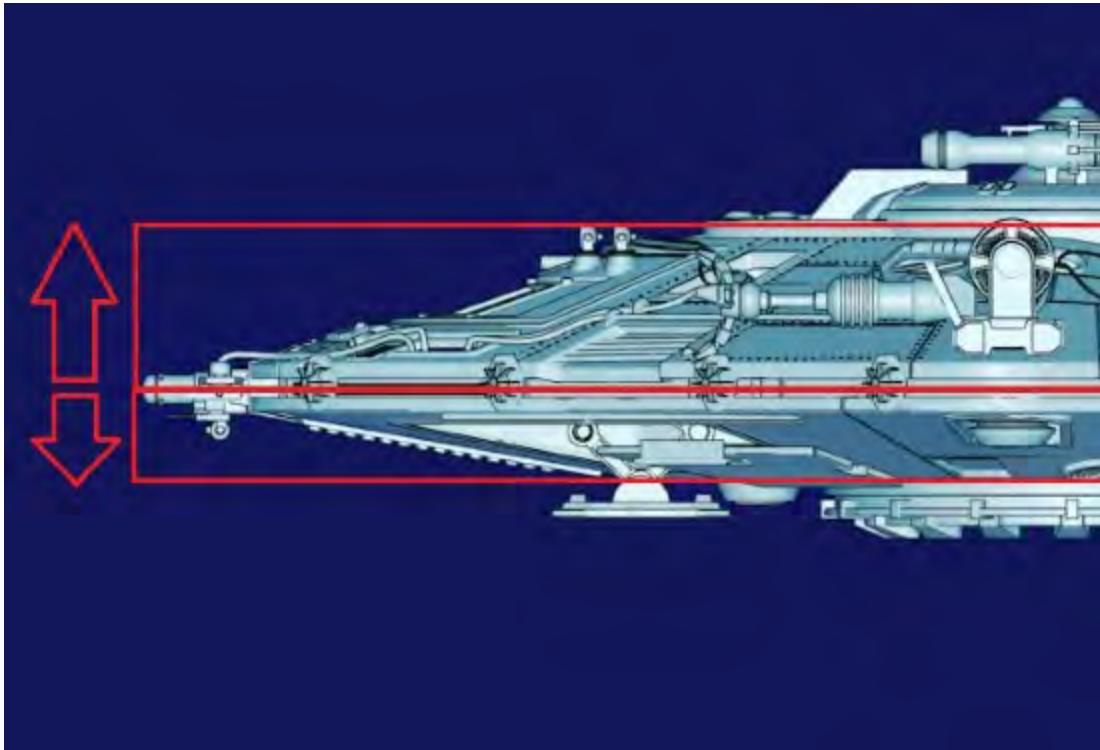
If this looks familiar, that's because it was one of my first major Blender projects. I released it under the Creative Commons Non-Attribution license, which means anyone can use the model (or the design) for whatever they want.

We don't need to replicate this ship exactly, but we'll use it as a guide. I've made several versions over the years, and no two are exactly alike. We'll just think of it as concept art.

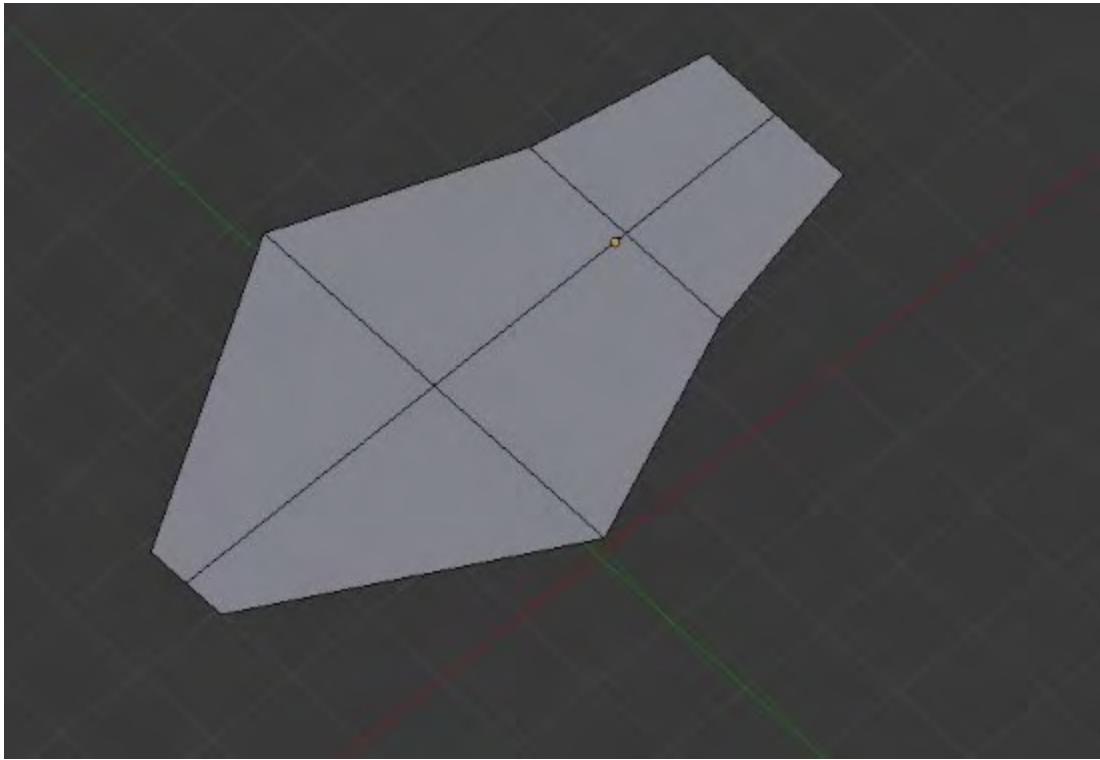
From the image, we can identify several key components and details that we'll want to pay attention to:



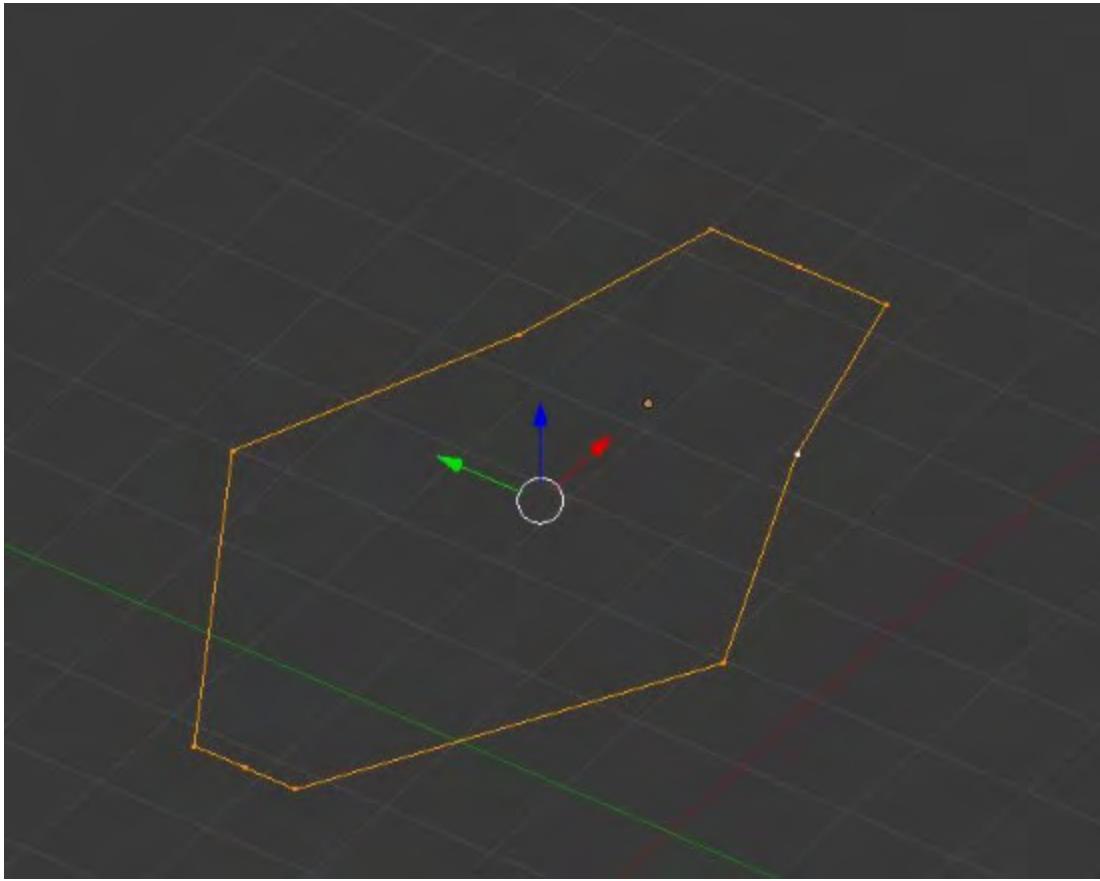
The first thing to focus on is the basic shape of the hull. You can see that the top and bottom sections are very similar, but not identical. The bottom section is significantly shorter than the top:



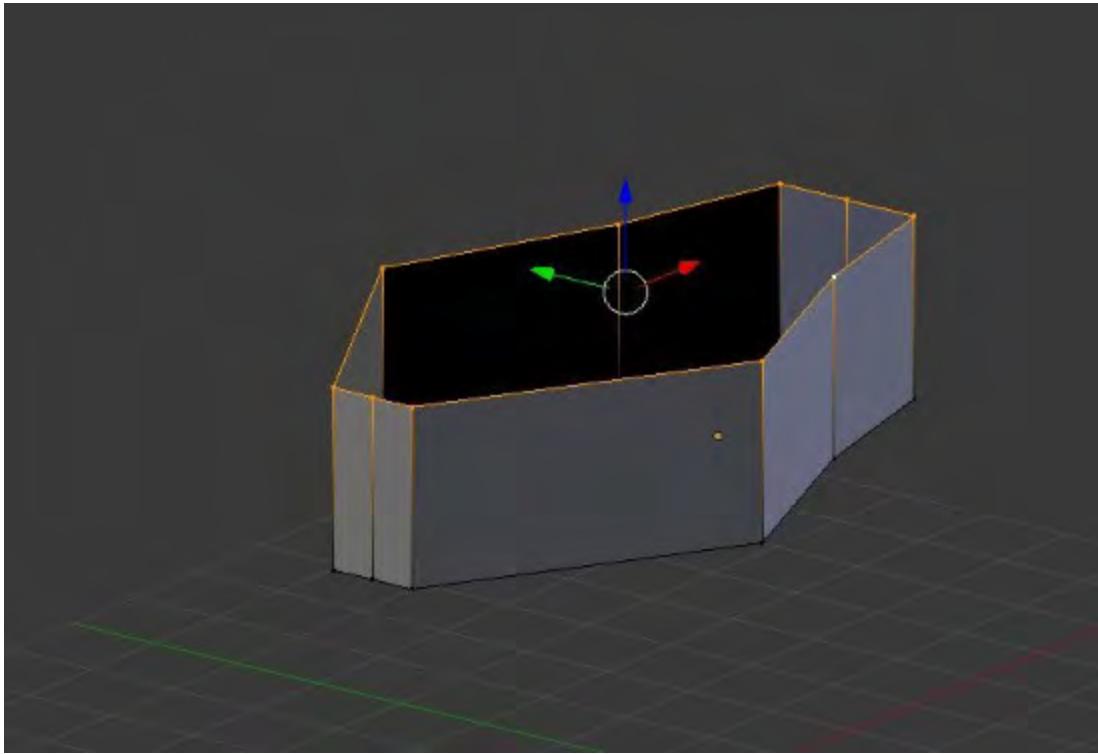
Rather than modeling them separately, it's probably easier to just create one section (the top, for example), and then copy and modify it as required. So, let's tackle that first. We'll open up a new scene in Blender and start with a basic plane. We'll then subdivide it twice with loop cuts, and move the vertices around to the approximate position that we want:



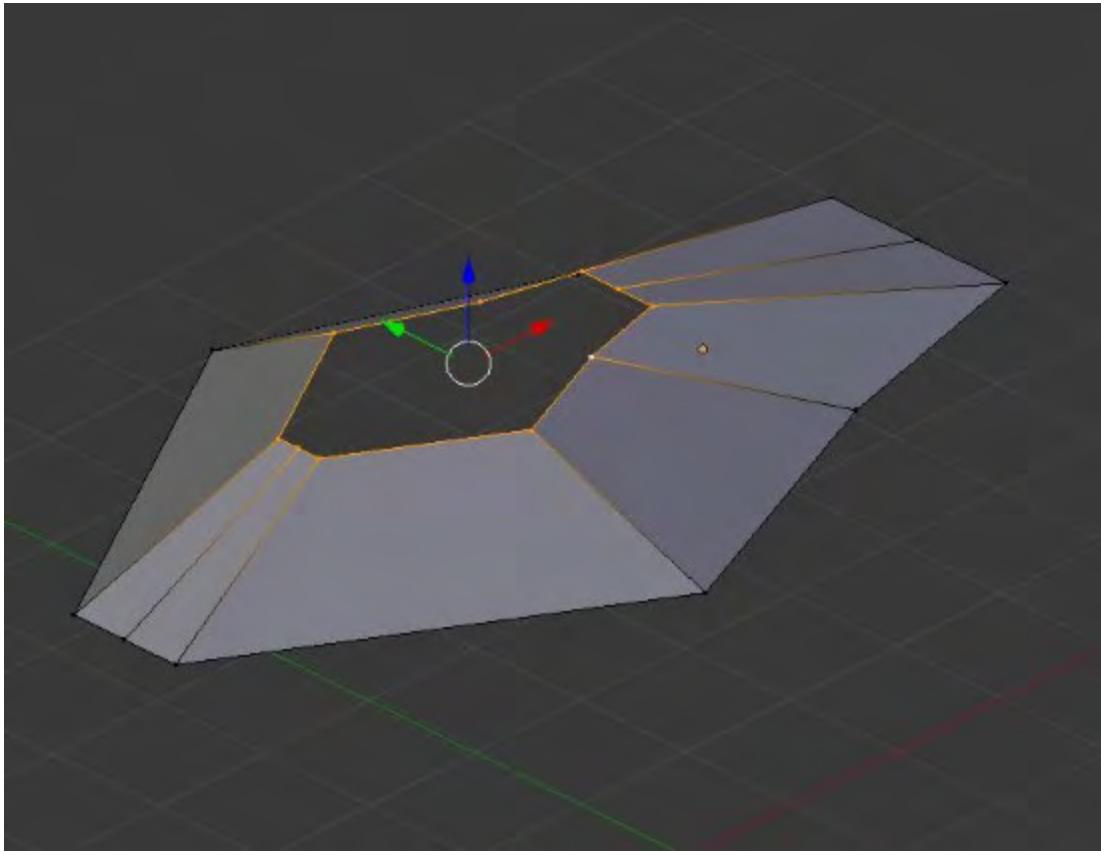
Next, we'll delete the inner vertices, leaving just a ring:



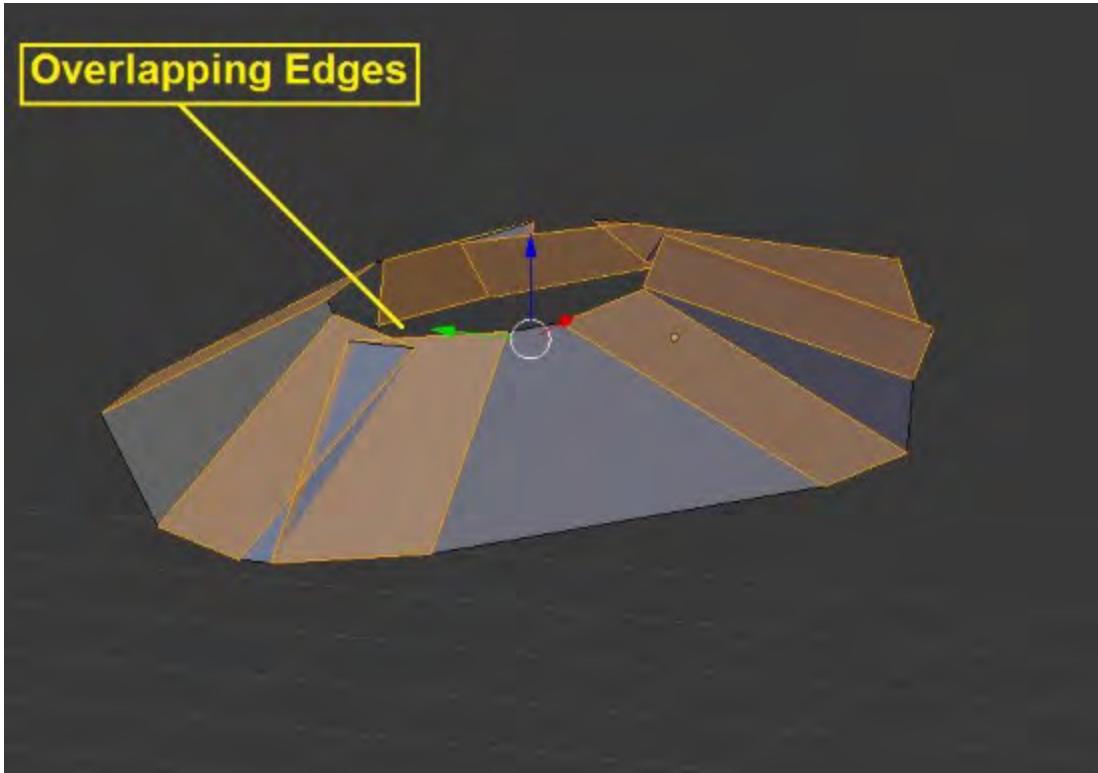
Then, we'll extrude those edges straight upwards:



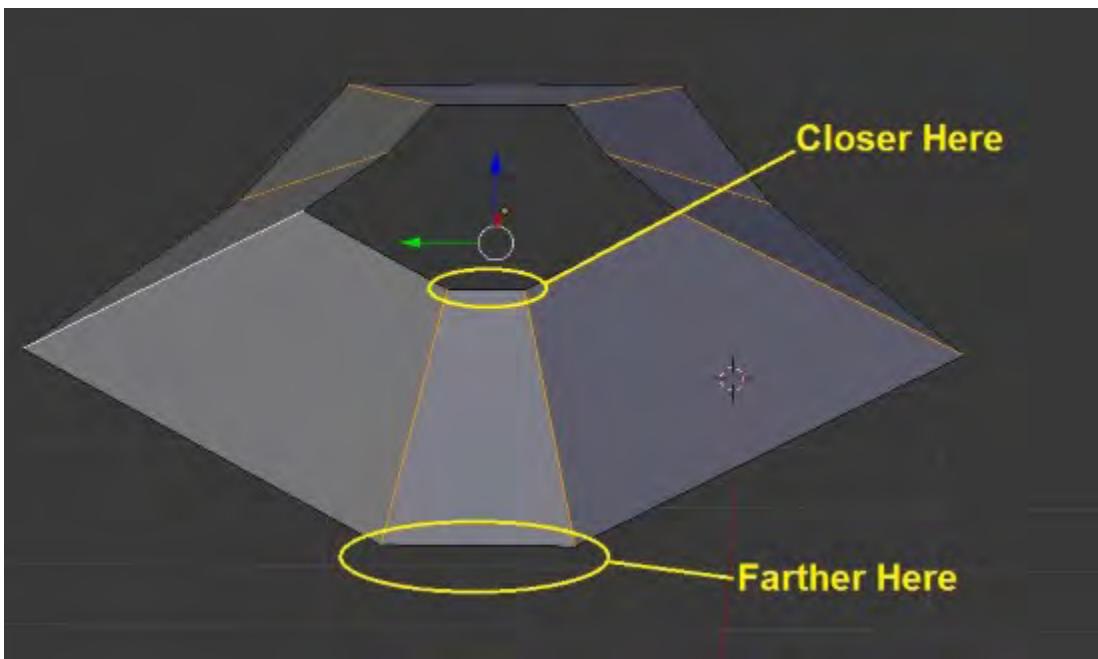
Obviously that's not how it looks in the design, but we're going to be beveling this to round out the shapes. If you'll recall from , Sci-Fi Pistol - Creating the Basic Shapes, that can cause problems sometimes. Let's say that we scaled down the top vertices to the approximate position that we wanted them to be:



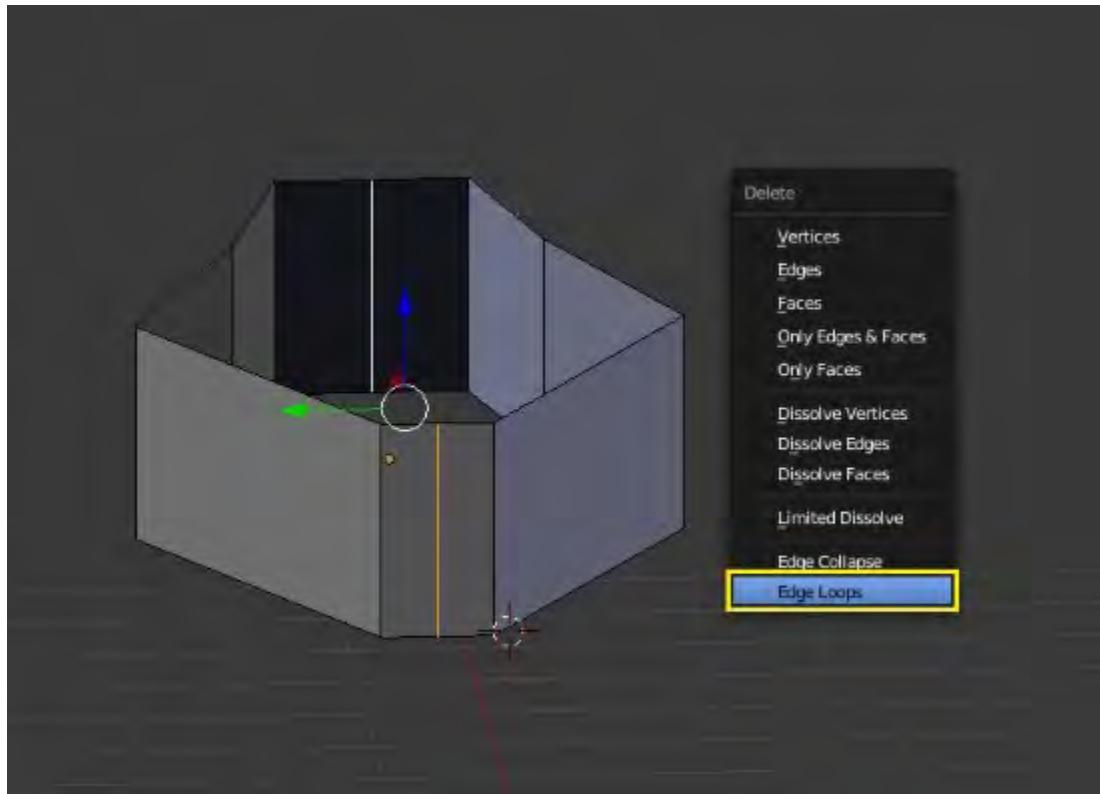
Now if we tried to bevel the edges, we'd end up with something like this:



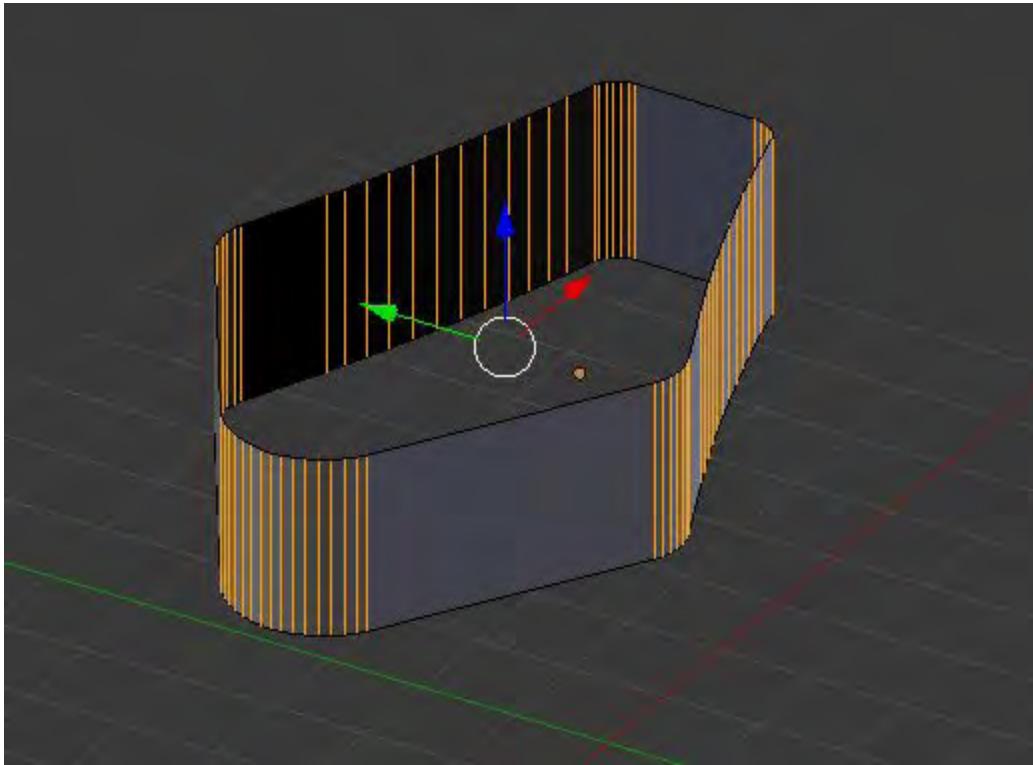
That's kind of a mess, isn't it? This type of overlap tends to occur when the vertices are closer together at one end than at the other:



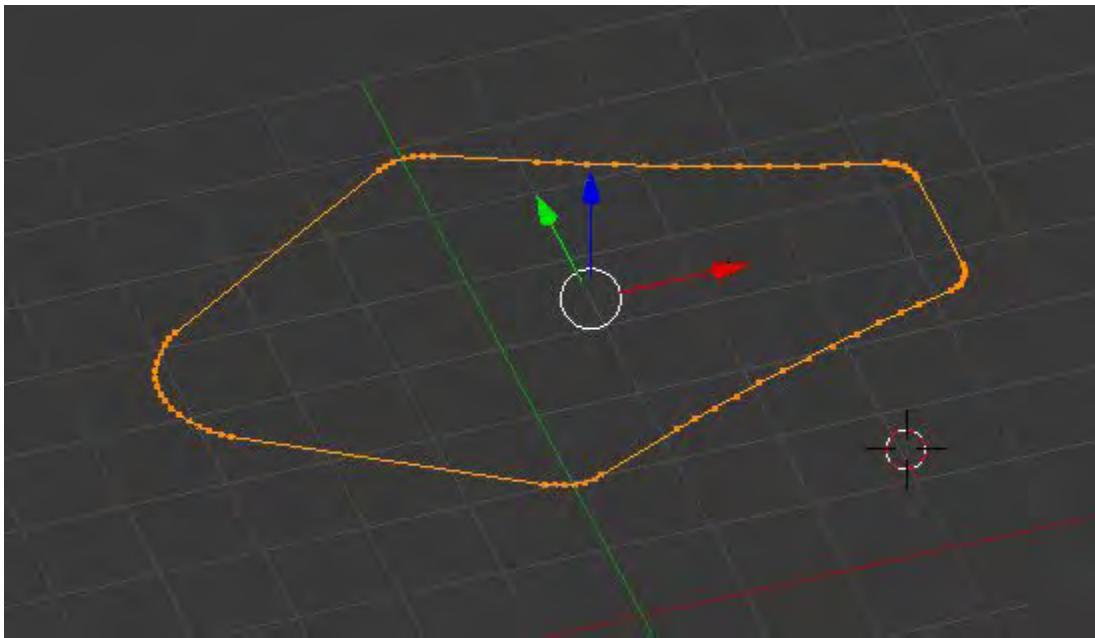
So to avoid this problem, we'll first extrude our edges straight up. Then we'll delete the two-edge loops on the ends so that we can bevel further:



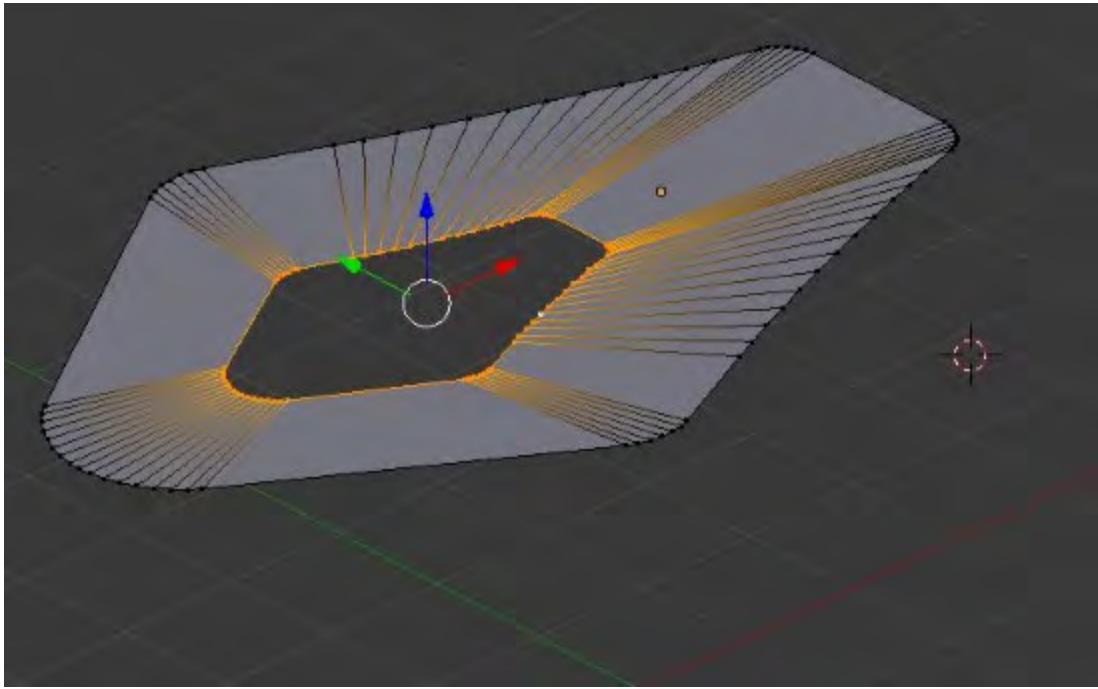
Then we can bevel it and get a nice smooth shape:



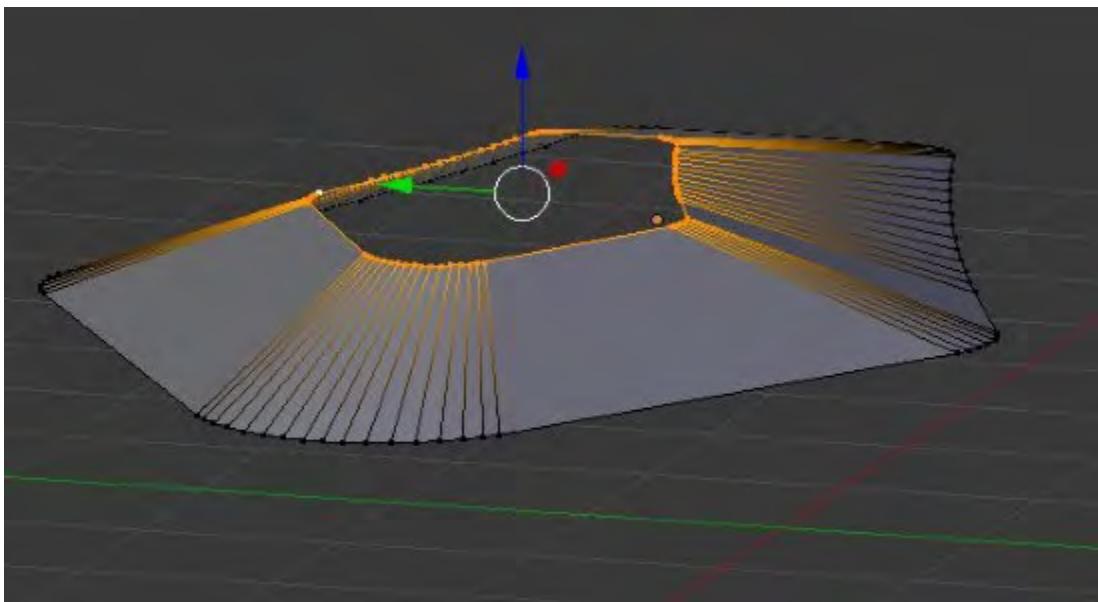
Once that's done, we can delete those top vertices:



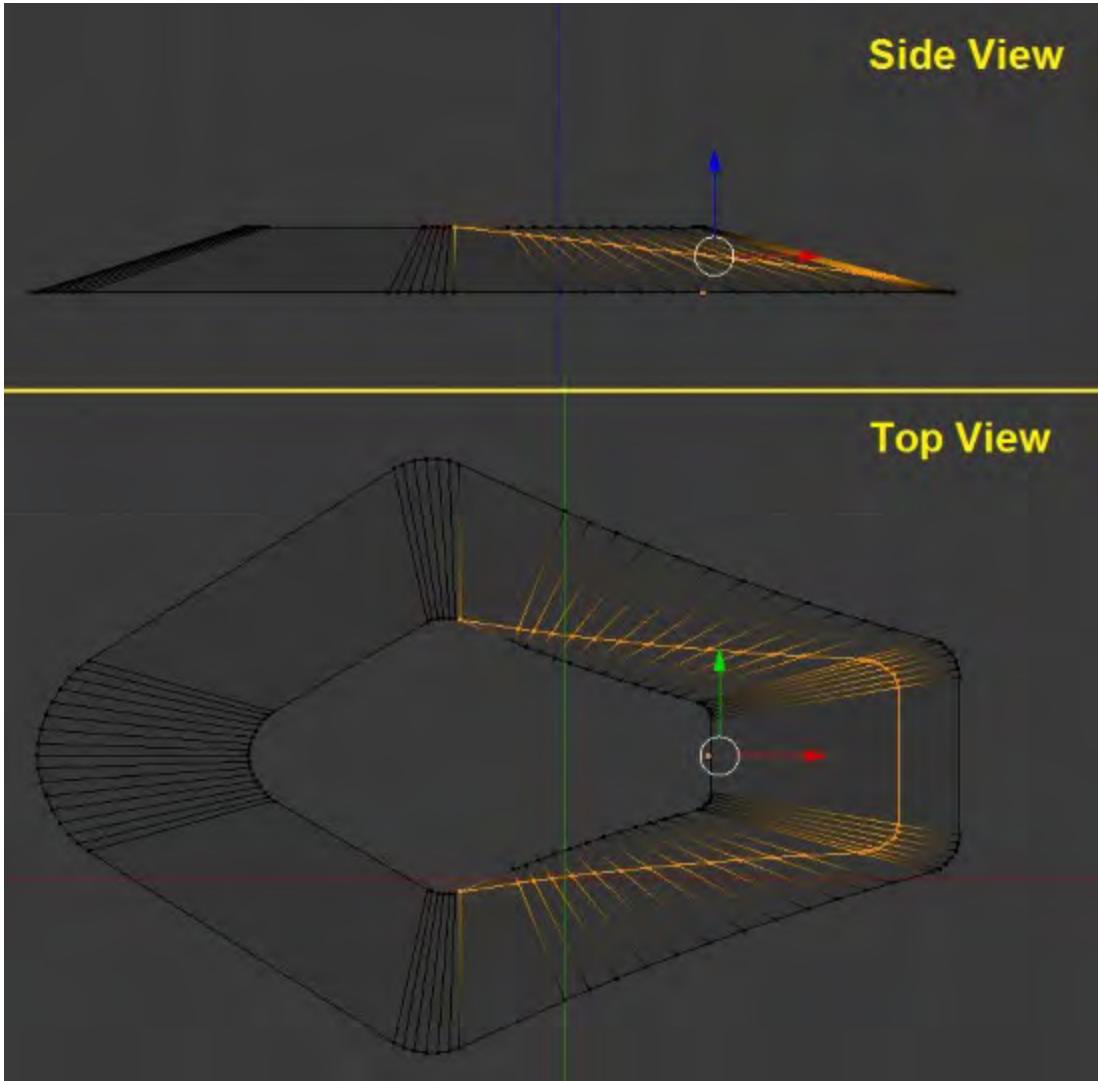
Then we can extrude the bottom ones in and move our inner ring to approximately the correct position:



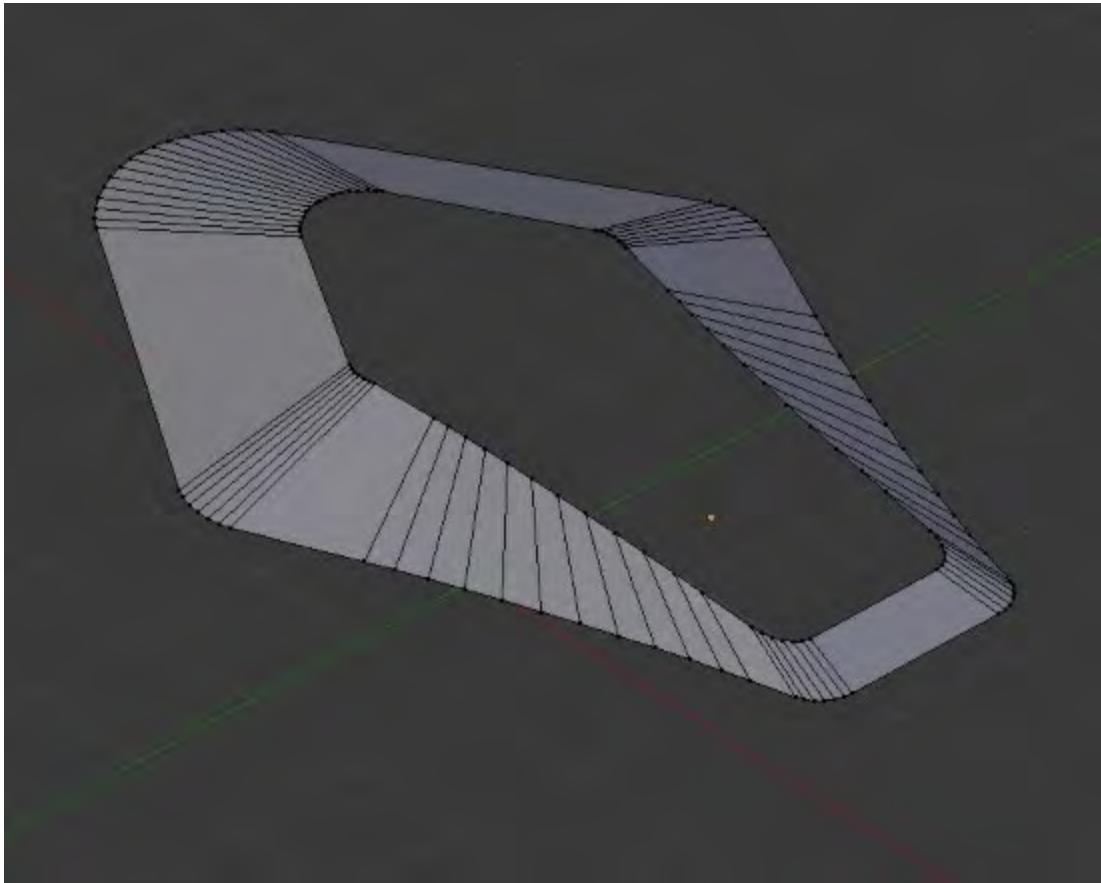
Now, we can pull the whole thing up without running into any trouble:



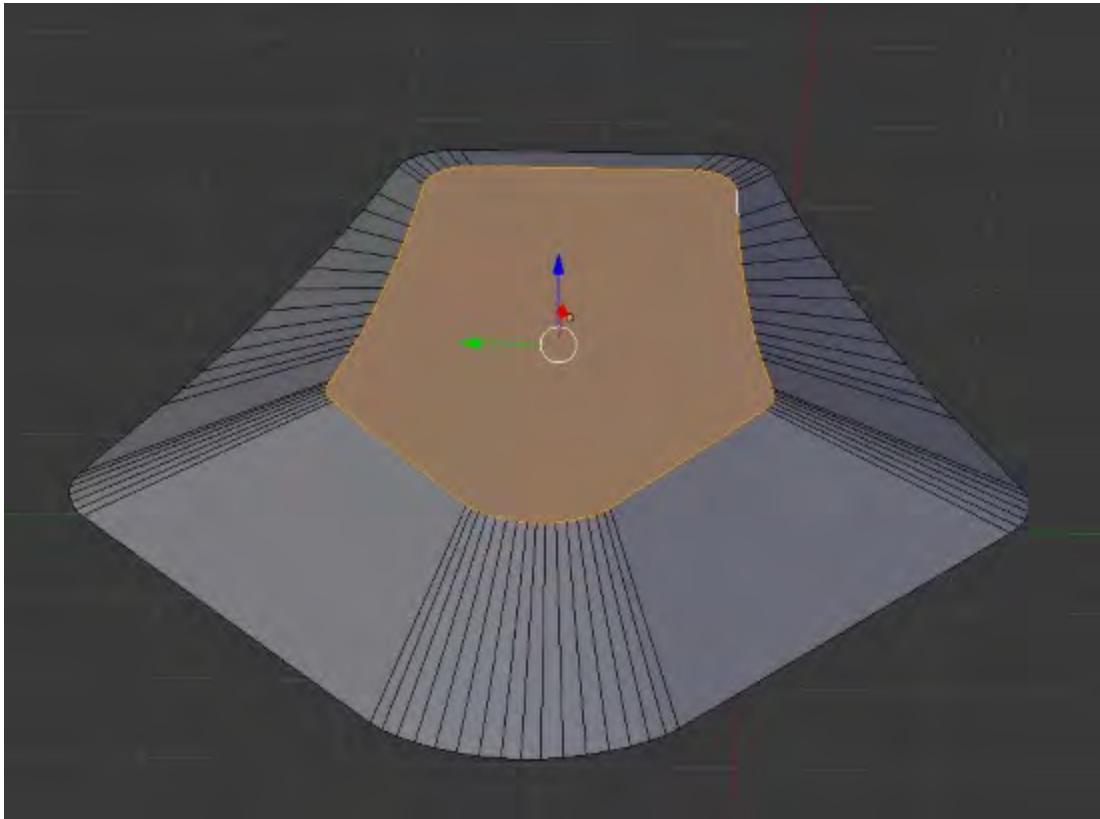
We'll then use our Knife tool from the side to cut the shape at an angle:



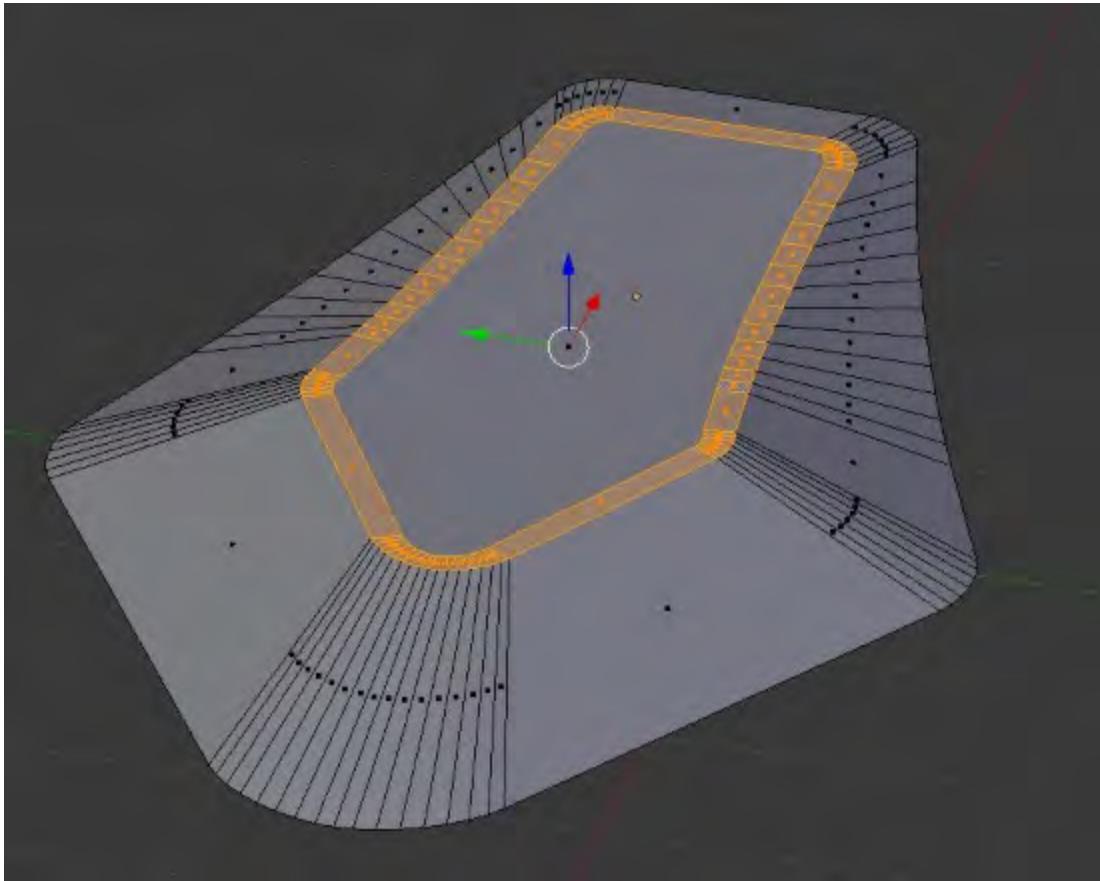
Delete those extra faces that we don't need, giving us the basic shape of the hull:



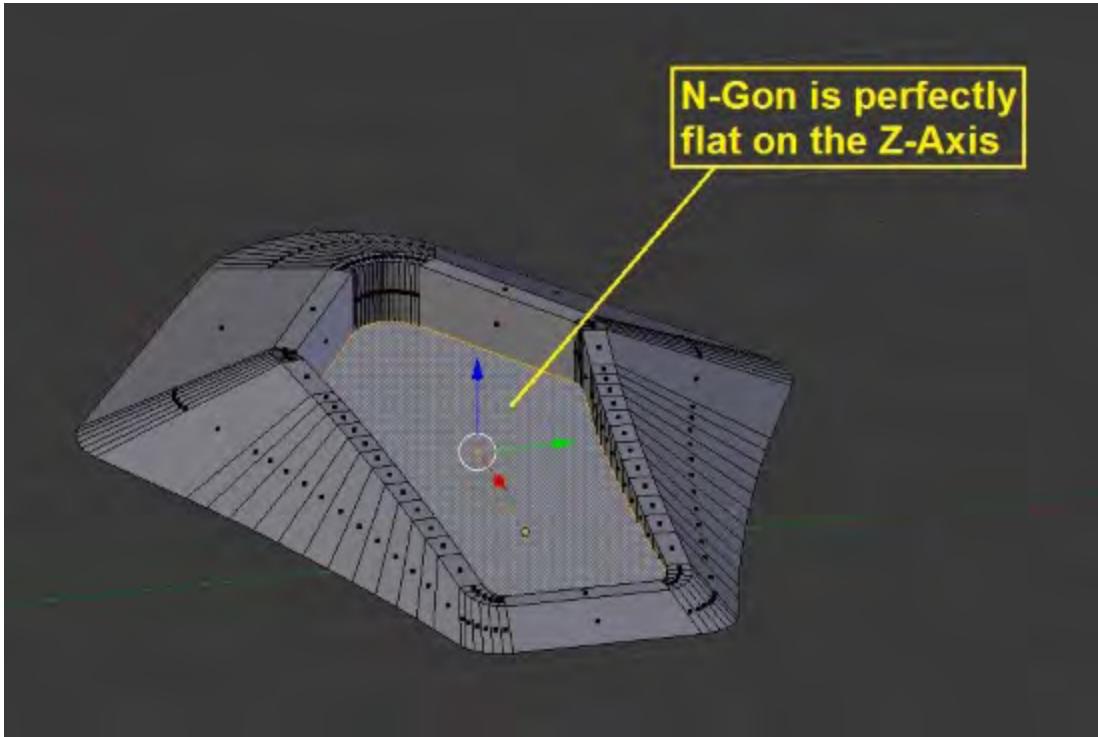
Let's create a temporary N-Gon here at the top. It looks pretty ugly, but that's okay; we'll fix it in a moment.



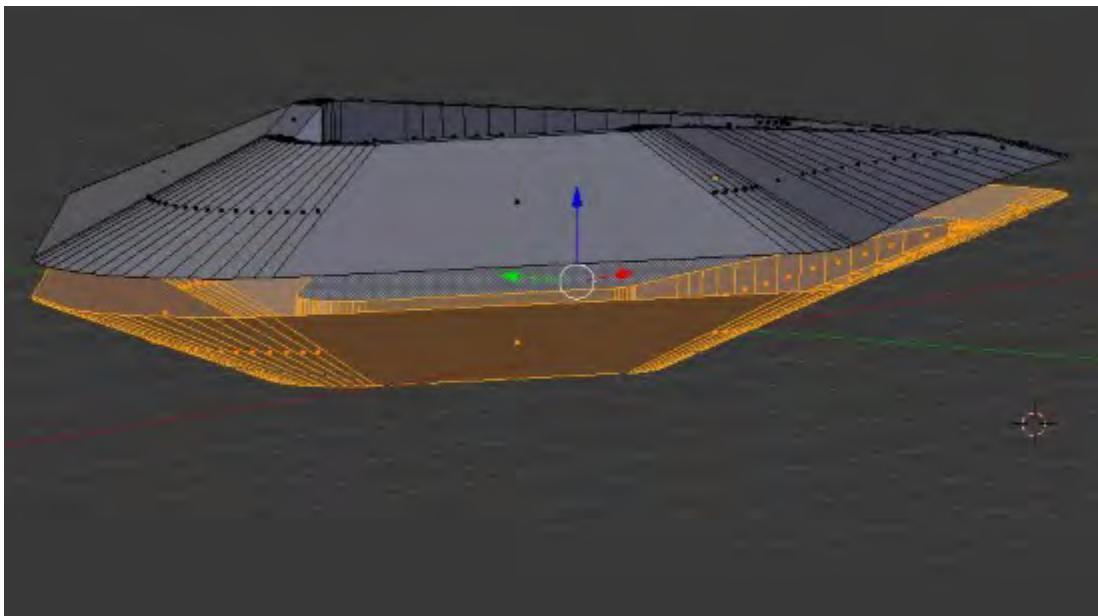
Next, we'll use Inset tool to create a nice even border around the top:



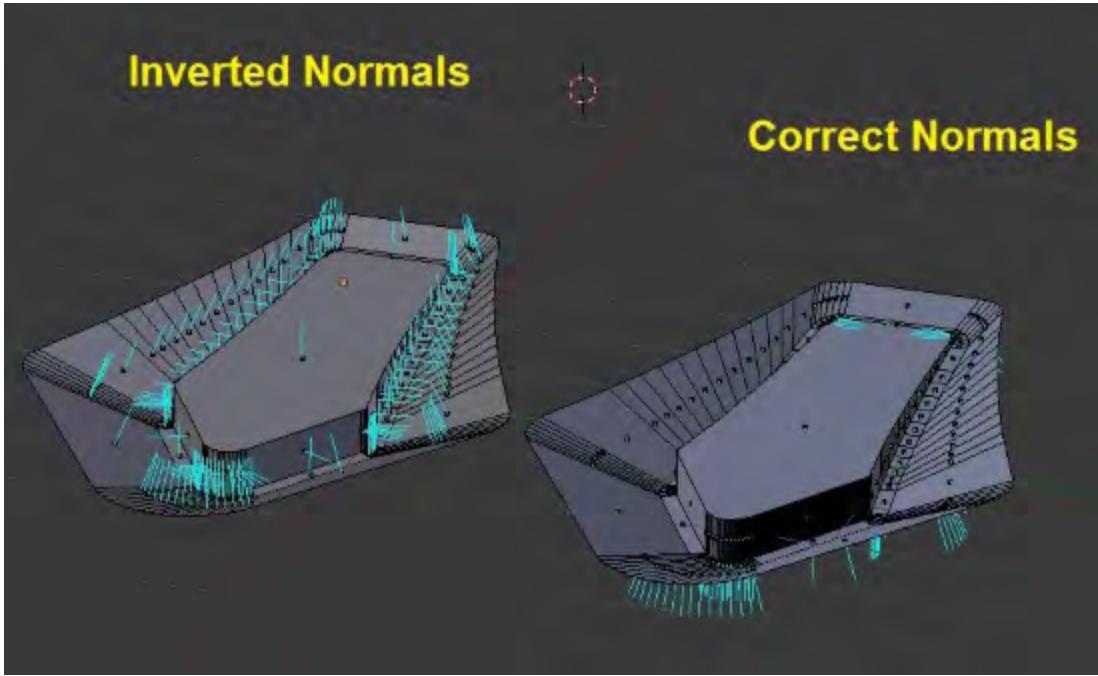
Then, we'll extrude that N-Gon down slightly into the hull and scale it to zero on the Z axis (S, Z, 0):



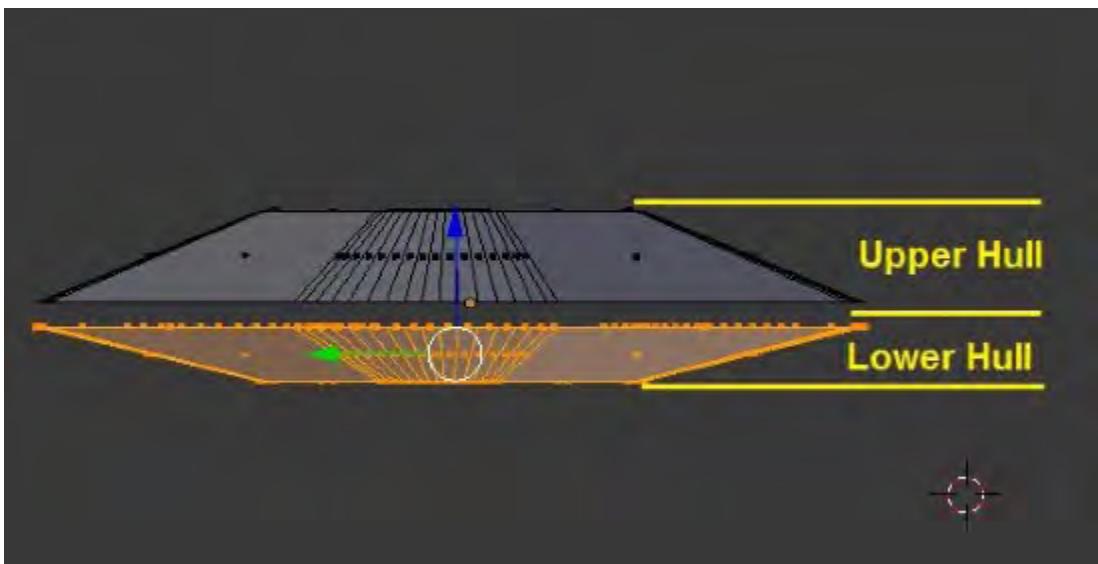
Before we go further, let's duplicate the upper section and flip it down across the Z axis (using Ctrl + M):



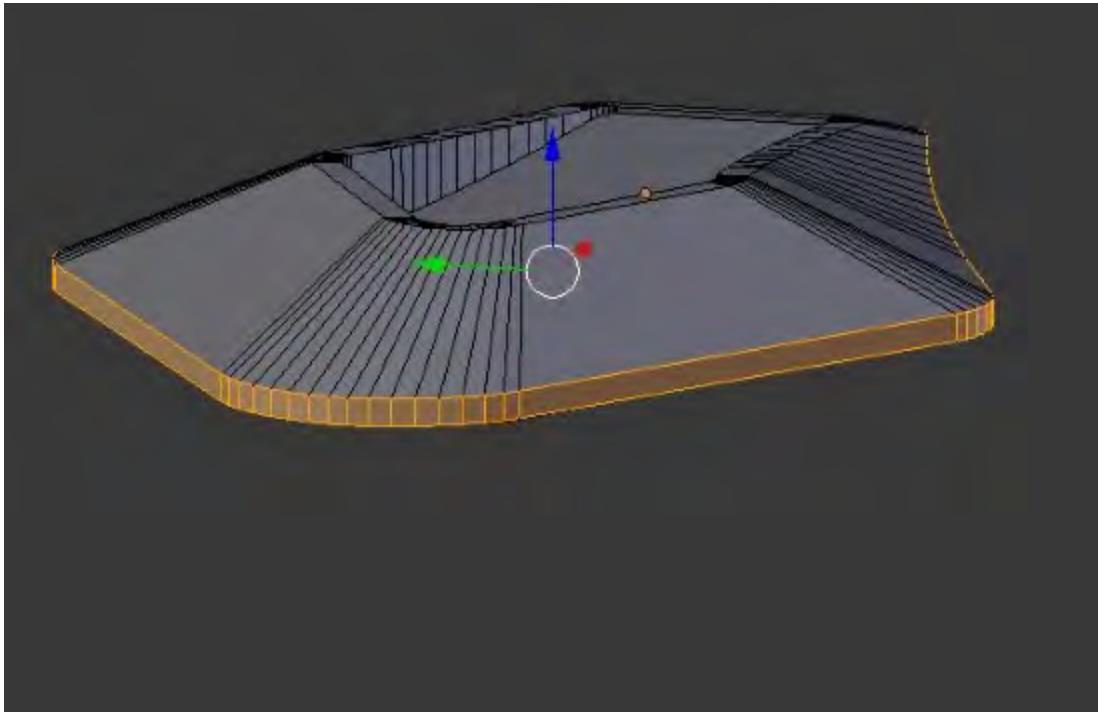
Don't forget to fix the **Normals**!



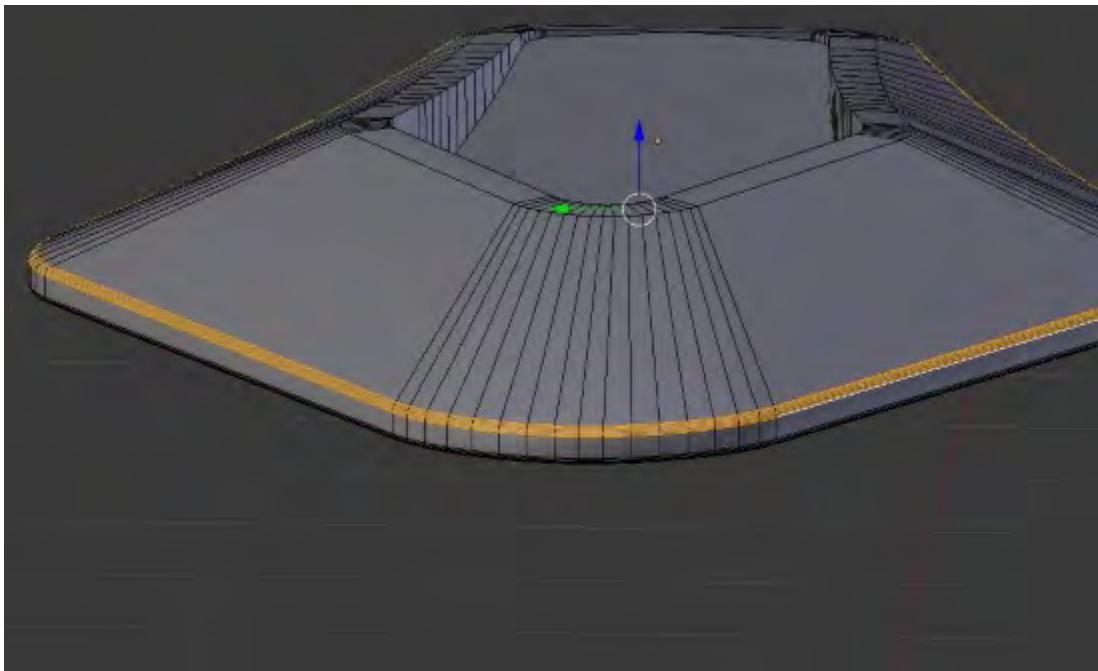
Let's scale the lower section down a bit to match the design:



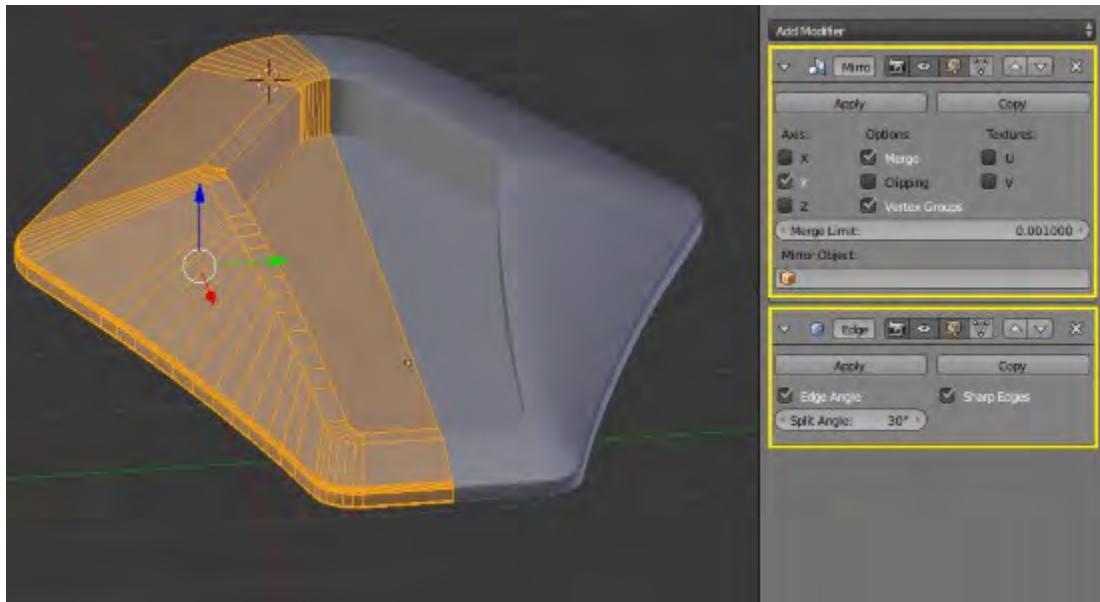
We can now select both the upper and lower edges and bridge the edge loops:



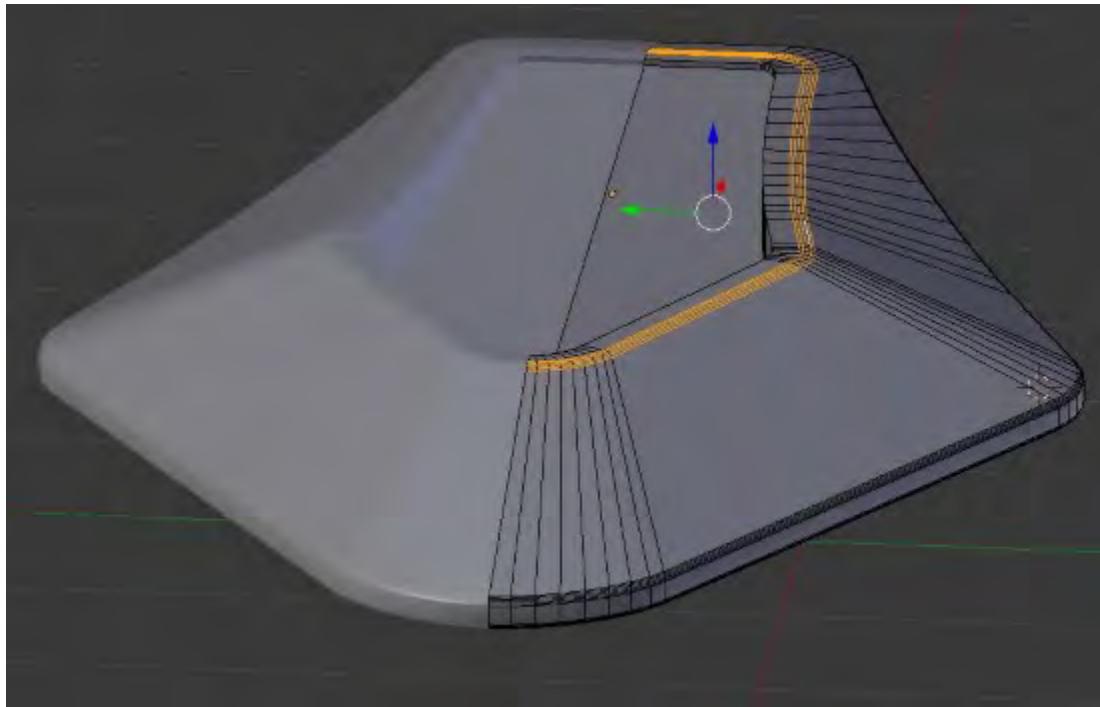
Now, we can bevel these edges at the top and bottom:



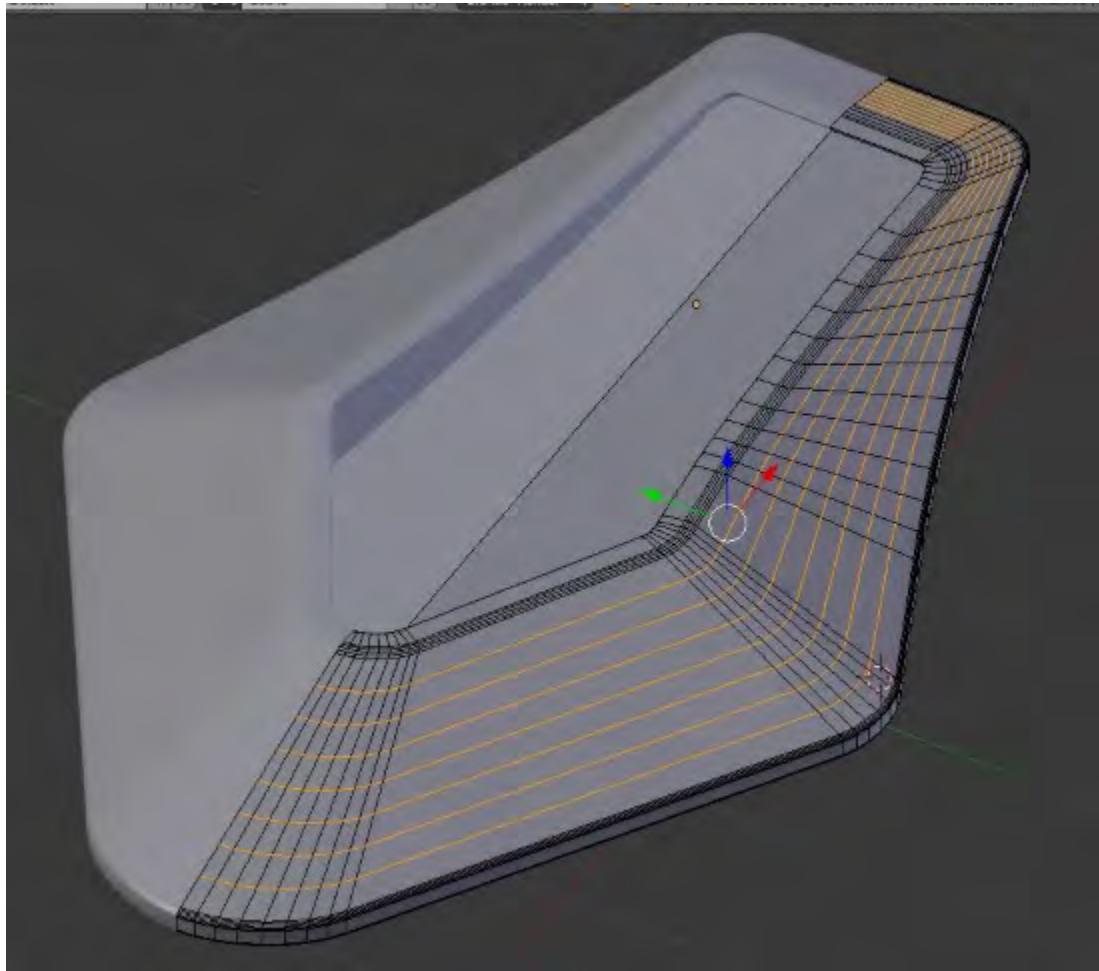
That gives us the basic shape of the body. Before we go further, let's delete half of it and add both a **Mirror** and an **Edge Split** modifier:



Then, we'll add a slight bevel to both the top and the bottom:



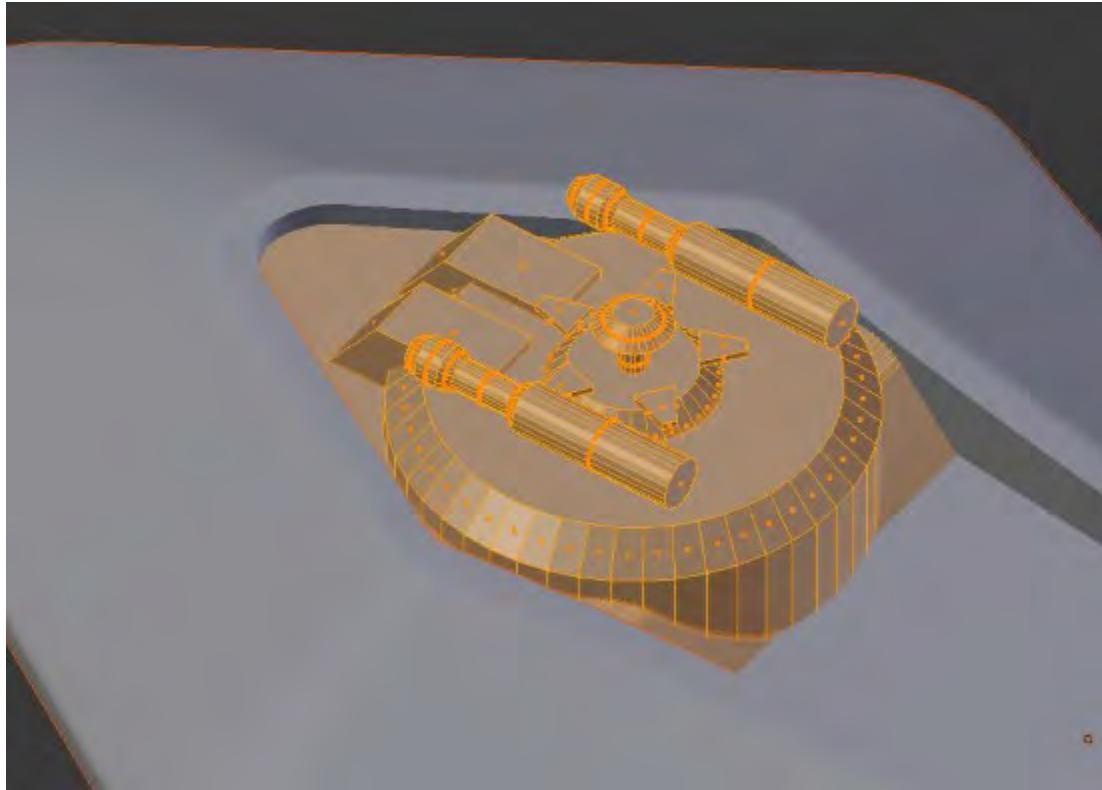
Now, we'll add a few more loop cuts around the side, just to give us more options when we make cuts in the mesh later on:



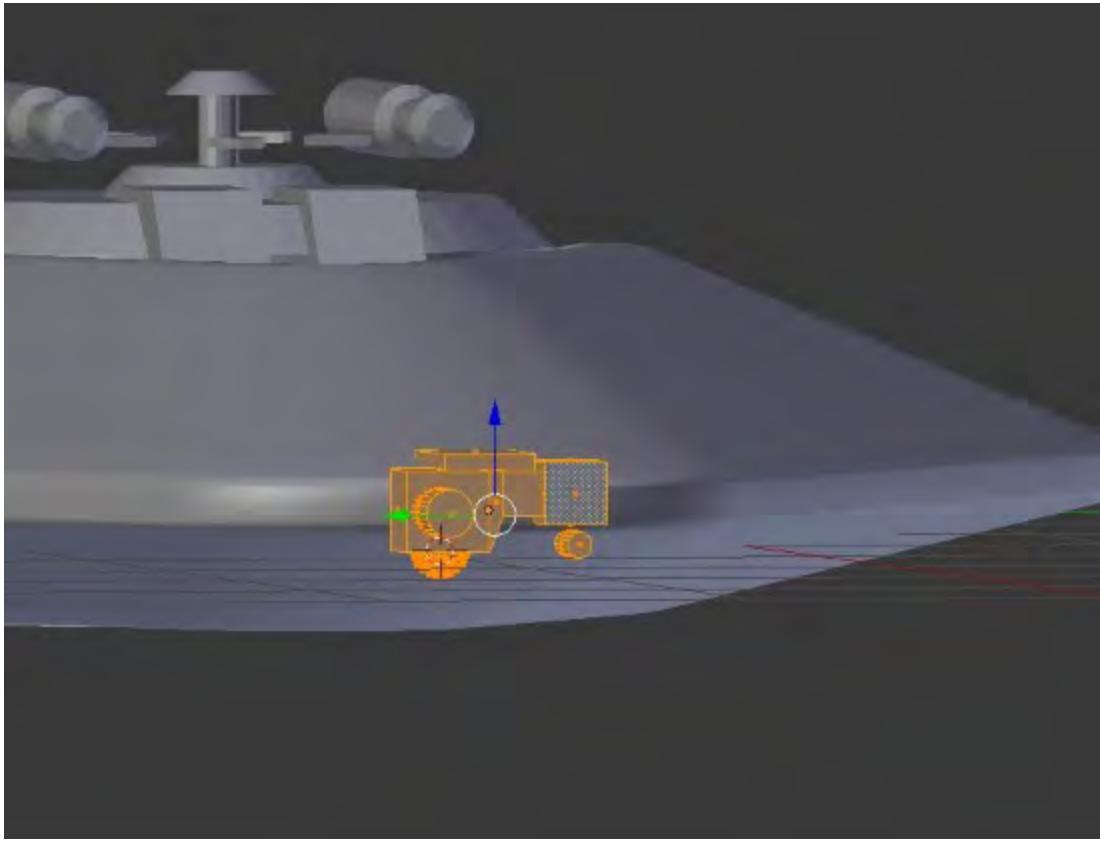
That wraps up the basic shape of the body. Wasn't too difficult, was it?

Creating the accessories

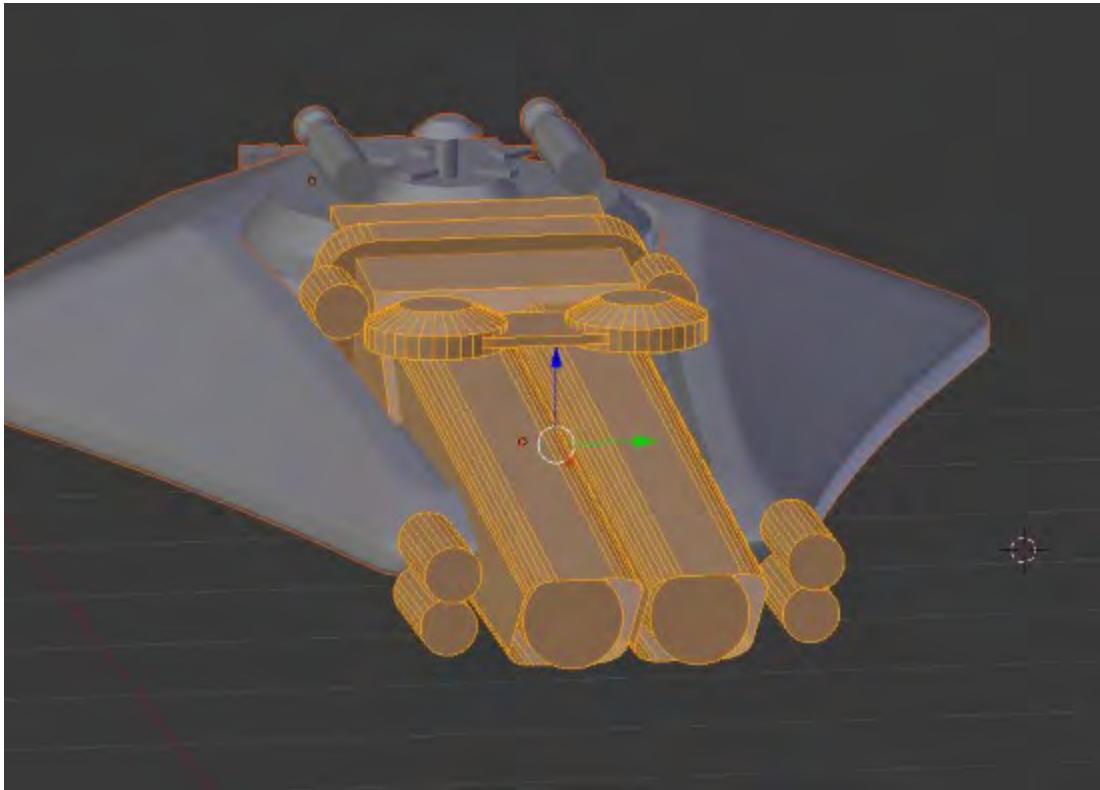
Next, I'll add in a couple basic cylinders and cubes at the top and represent the equipment up there:



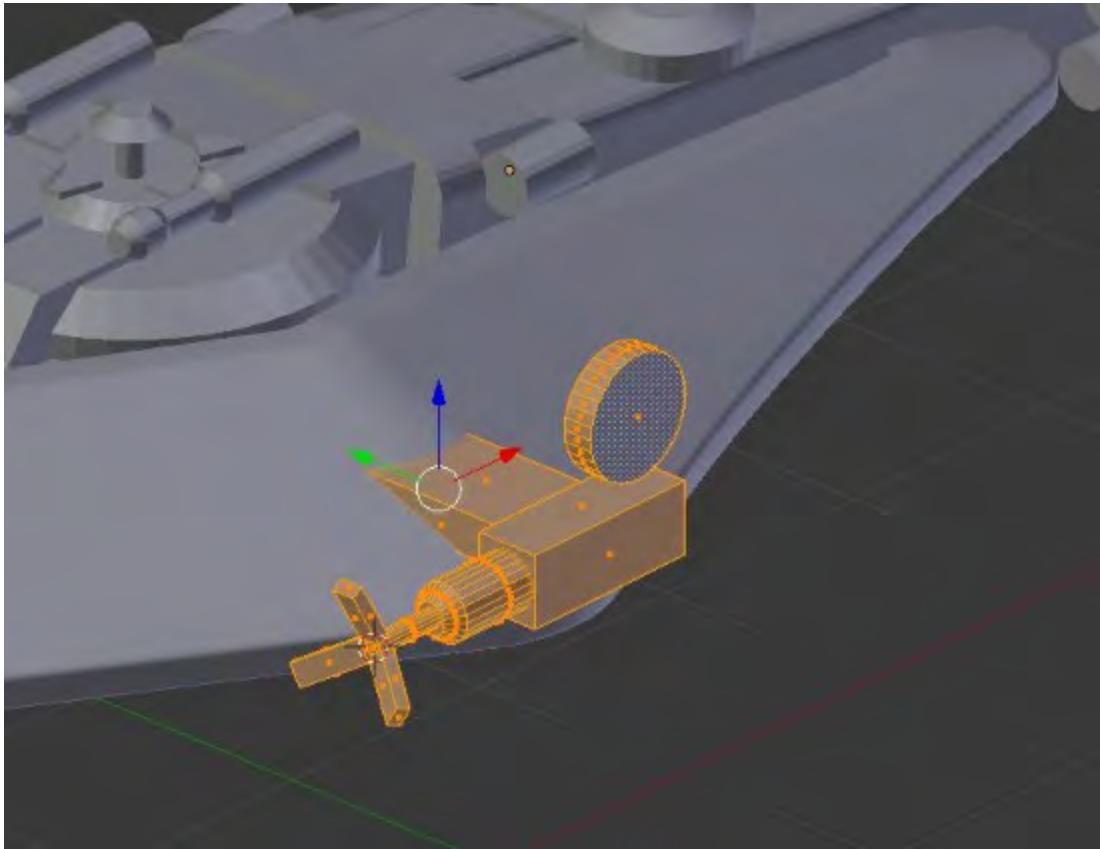
Then, I'll do the same for the navigation array at the front:



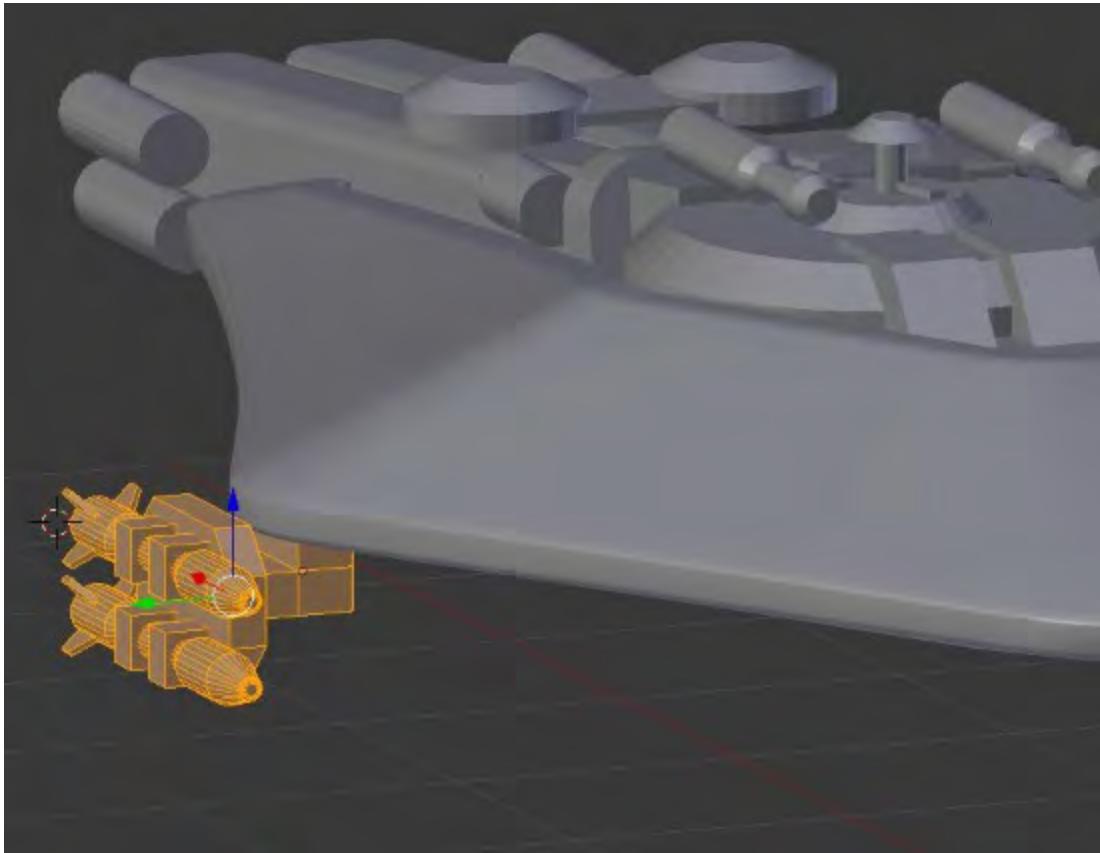
At the back, I'll add a couple of beveled cubes and cylinders to represent the engine area:



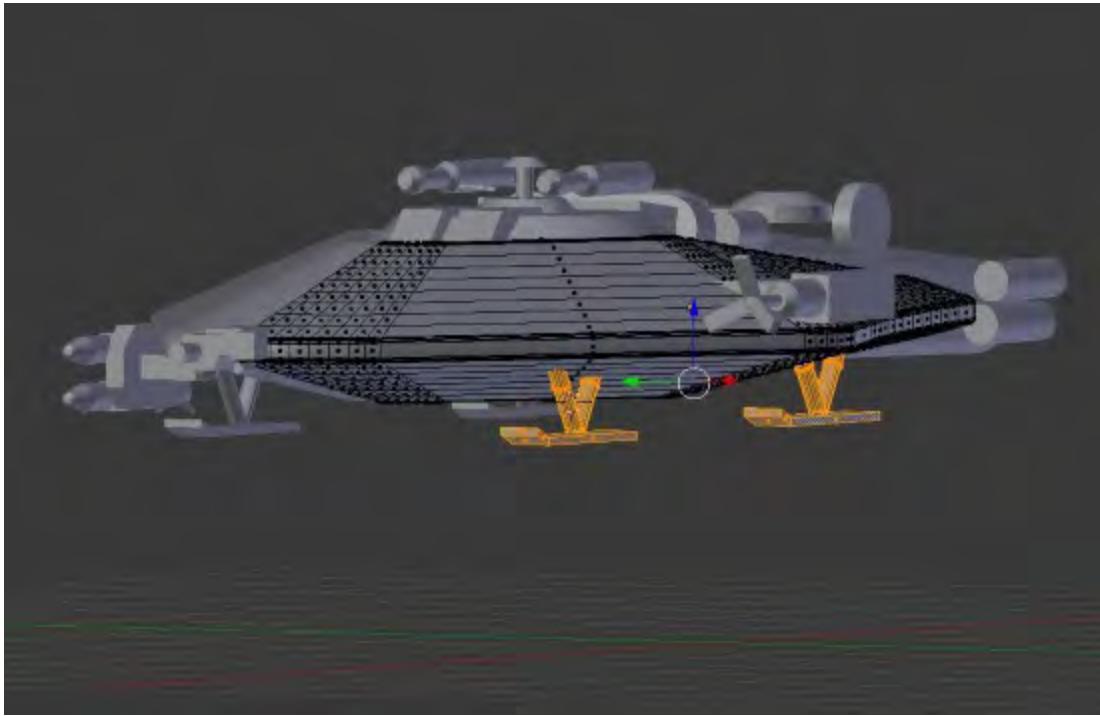
I'll also add some basic shapes in for the grappling gun on the port side:



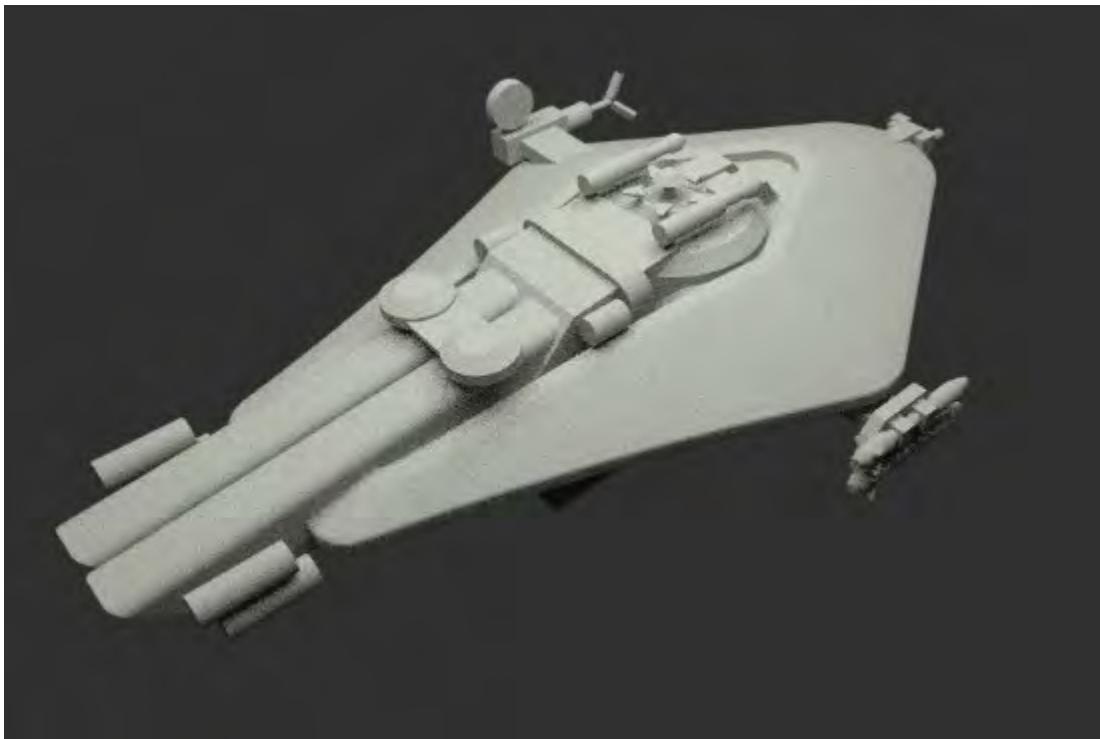
And the missile launcher on the starboard side:



I'll block out some basic landing gear (skids) on the bottom:



That really wraps up the basic shapes of our ship:



It's still pretty far from the concept, but you can start to see it taking shape. Now is an excellent time to experiment with the basic layout of the parts. If you want to move things around or make them different, this is a great opportunity.

Summary

In this section, we got the basic shapes of our ship modeled. In the next chapter, we'll go through and fill in all the details of our spacecraft. As we do, we'll discuss the overall workflow of detailing a complex object like this.

When you feel like your ship is ready, we'll get started!

Chapter 5. Spacecraft - Adding Details

In this chapter, we'll finish modeling our spacecraft. As we do, we'll cover a few new tools and techniques. Mostly, however, we'll be expanding on what we have already learned, and applying things in different ways to create a final, complex model:

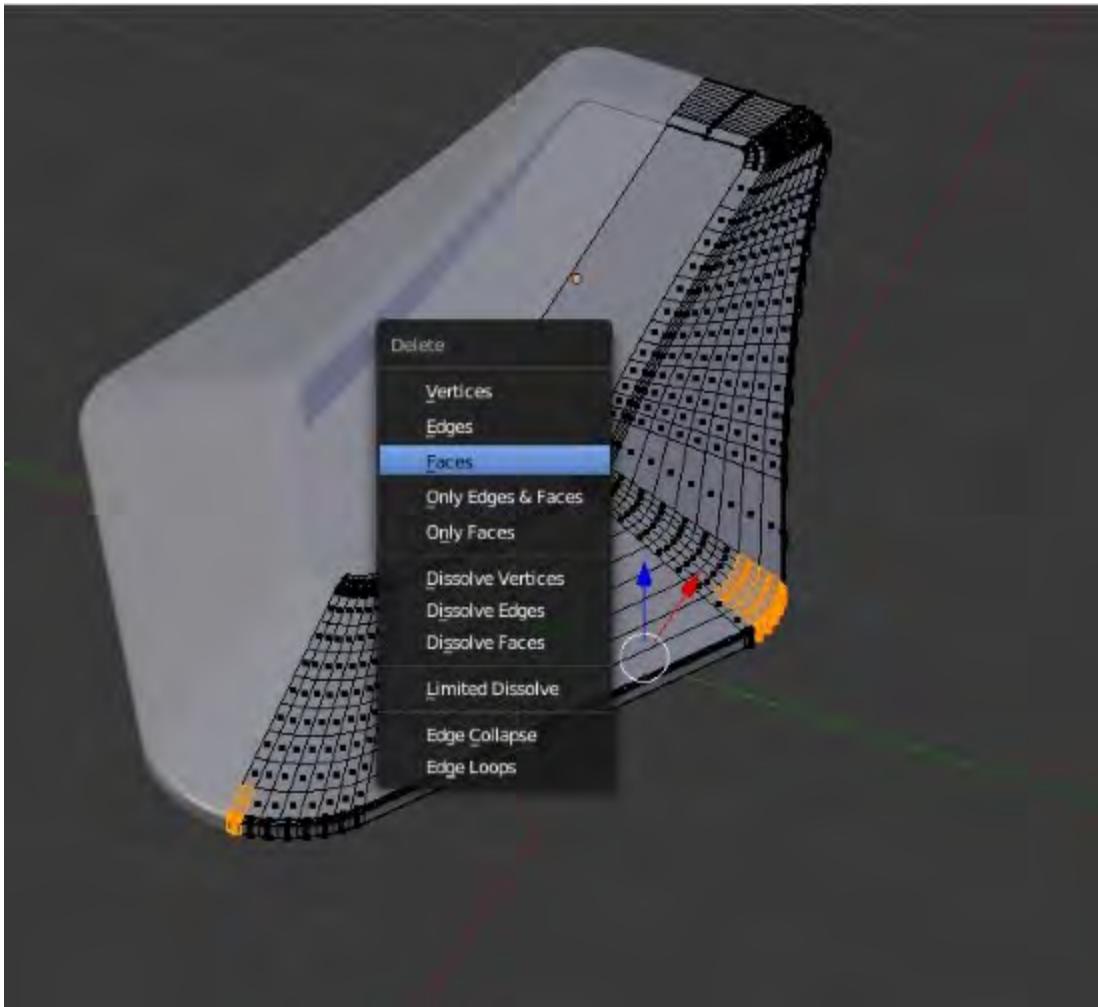
- Refining our ship
- Adding detail with pipes
- Finishing our main objects
- Do it yourself - completing the main body
- Building the landing gear
- Adding the cockpit
- Creating small details
- Working with Path objects
- Finishing touches

Refining our ship

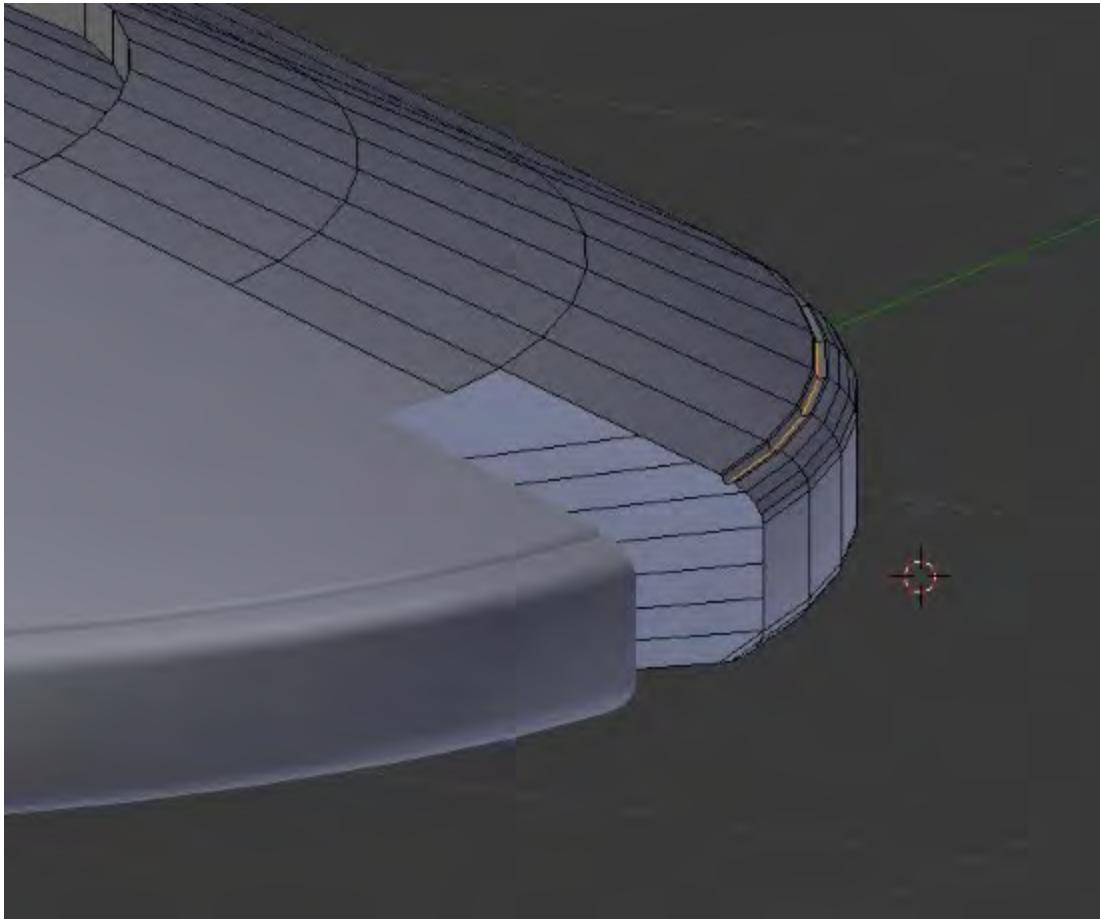
We'll work though the spacecraft one section at a time, adding detail until it's finished.

<pagebreak></pagebreak>

Let's start by moving everything but the main body to another layer. Then, we'll select the portions of the body that we want to remove (to create gaps in the hull) and delete them:

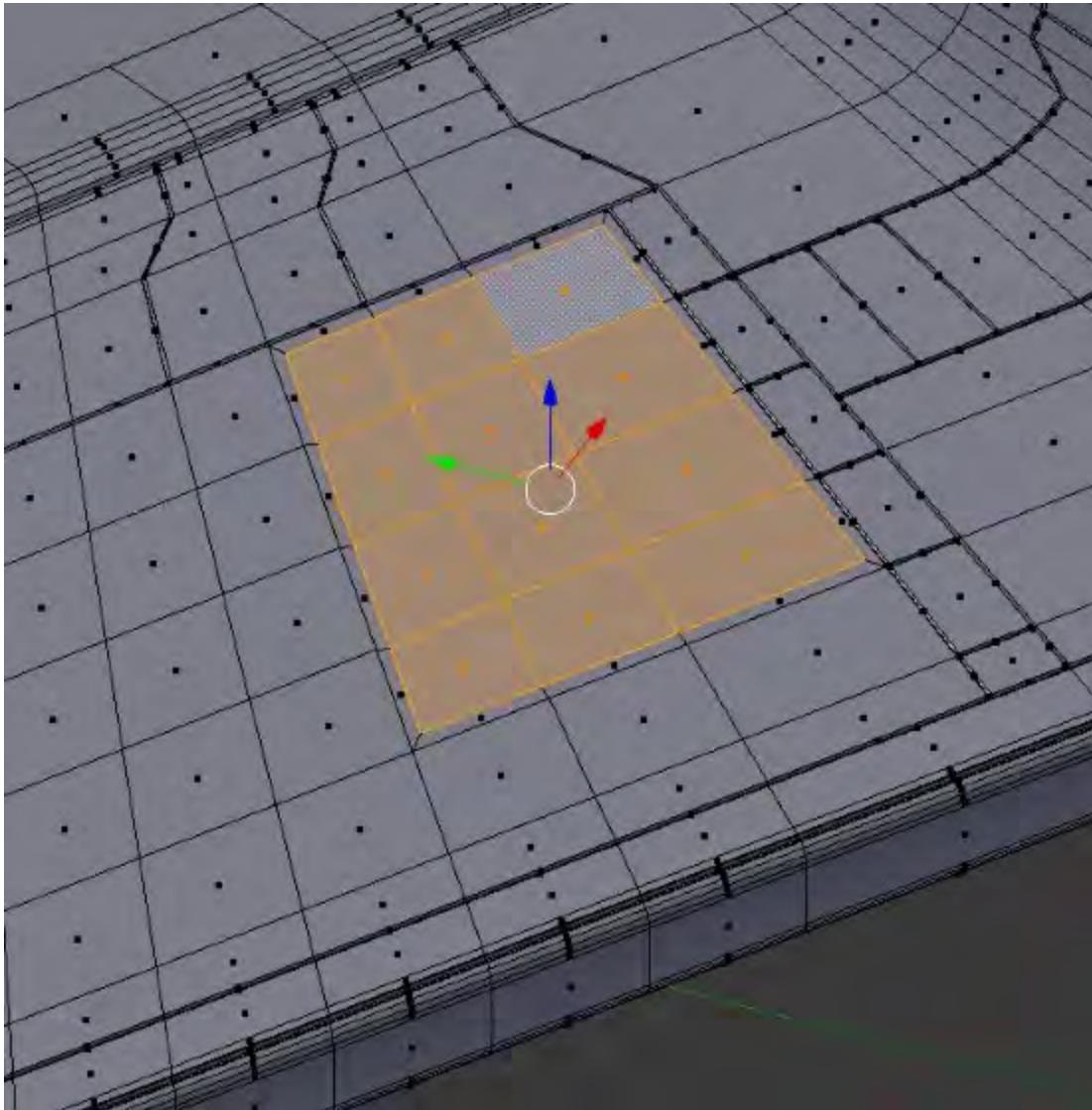


We can use Alt + S to scale in some edges, then rip them with the V key:

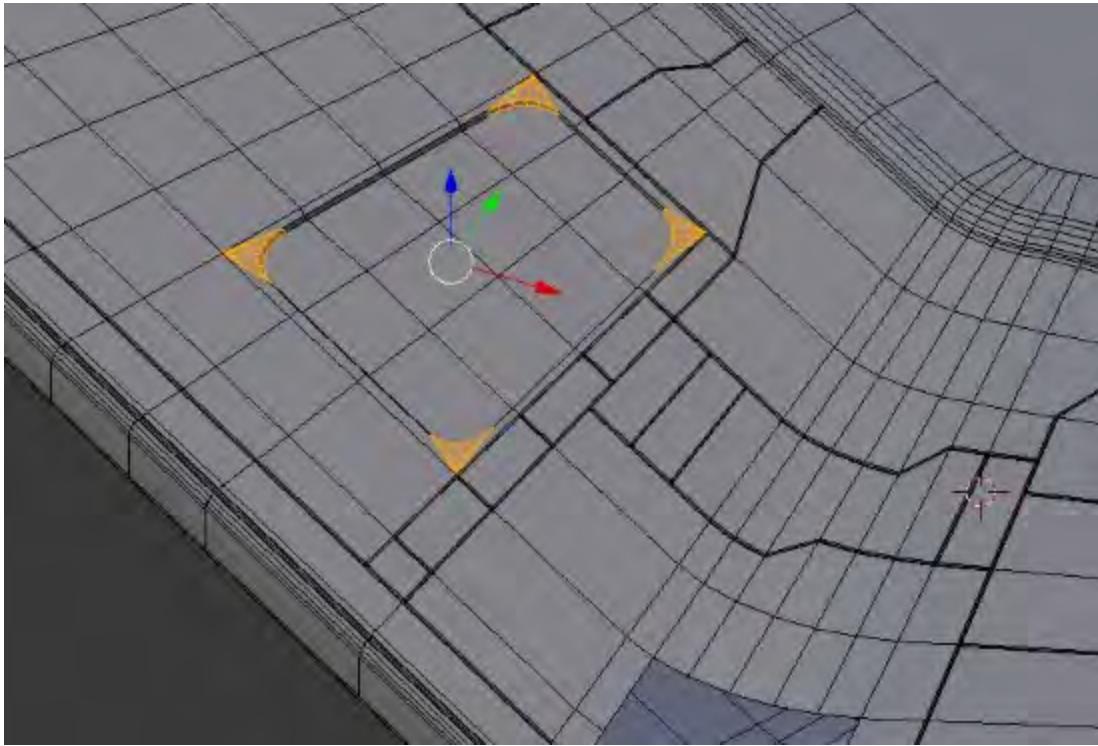


<pagebreak></pagebreak>

I'm also going to use the Inset tool to create a door (or hatch) on the side of the ship:

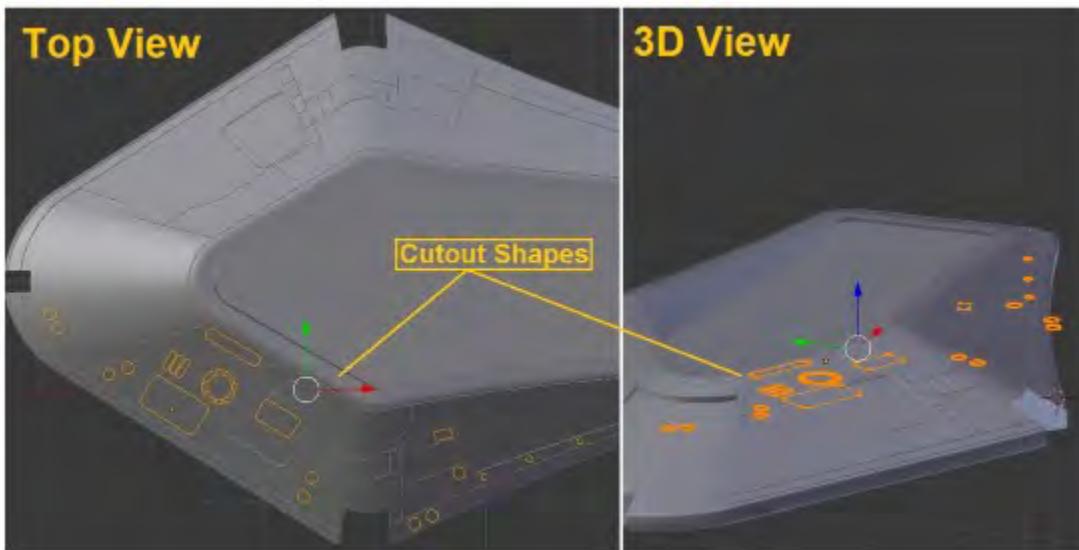


Then, I'll just bevel those edges so that they're more rounded:

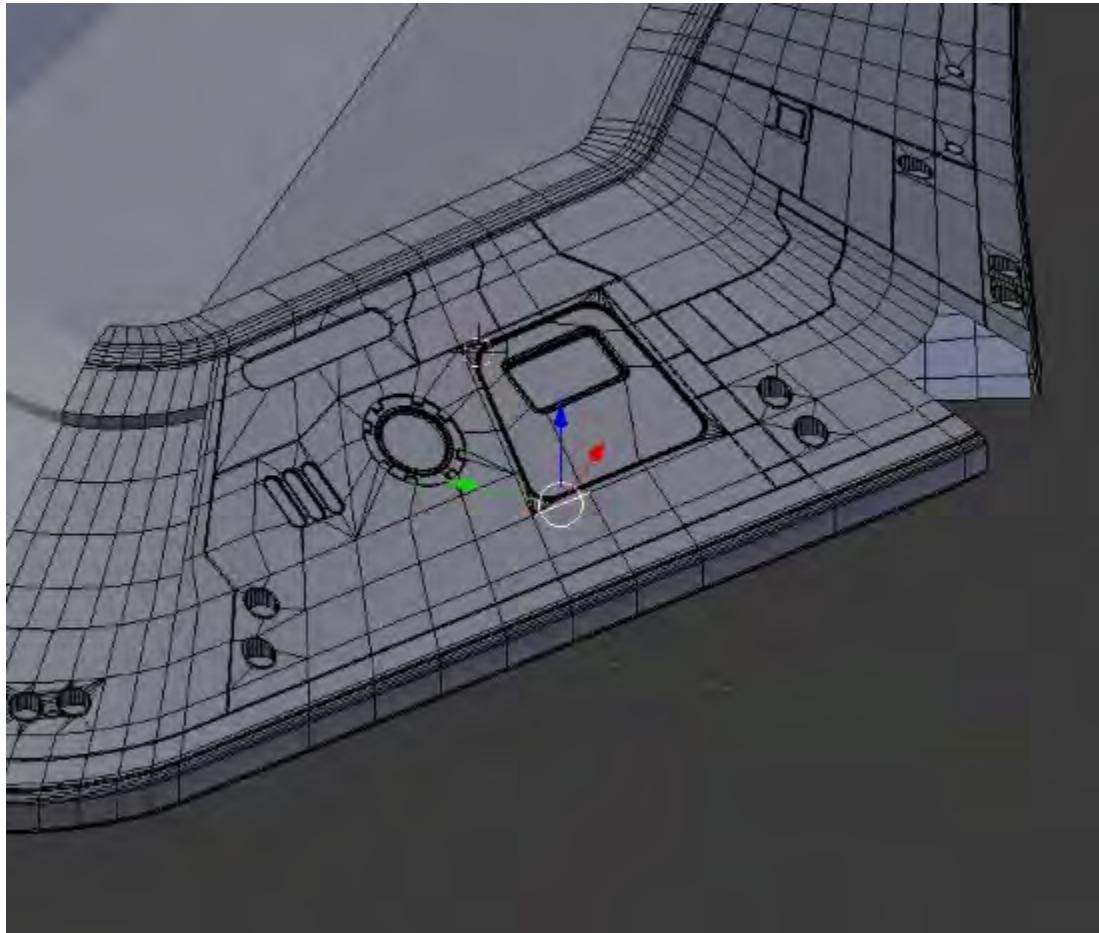


<pagebreak></pagebreak>

Next, we'll create a series of cutout shapes to use for the top of the hull (just like we did on our gun model):

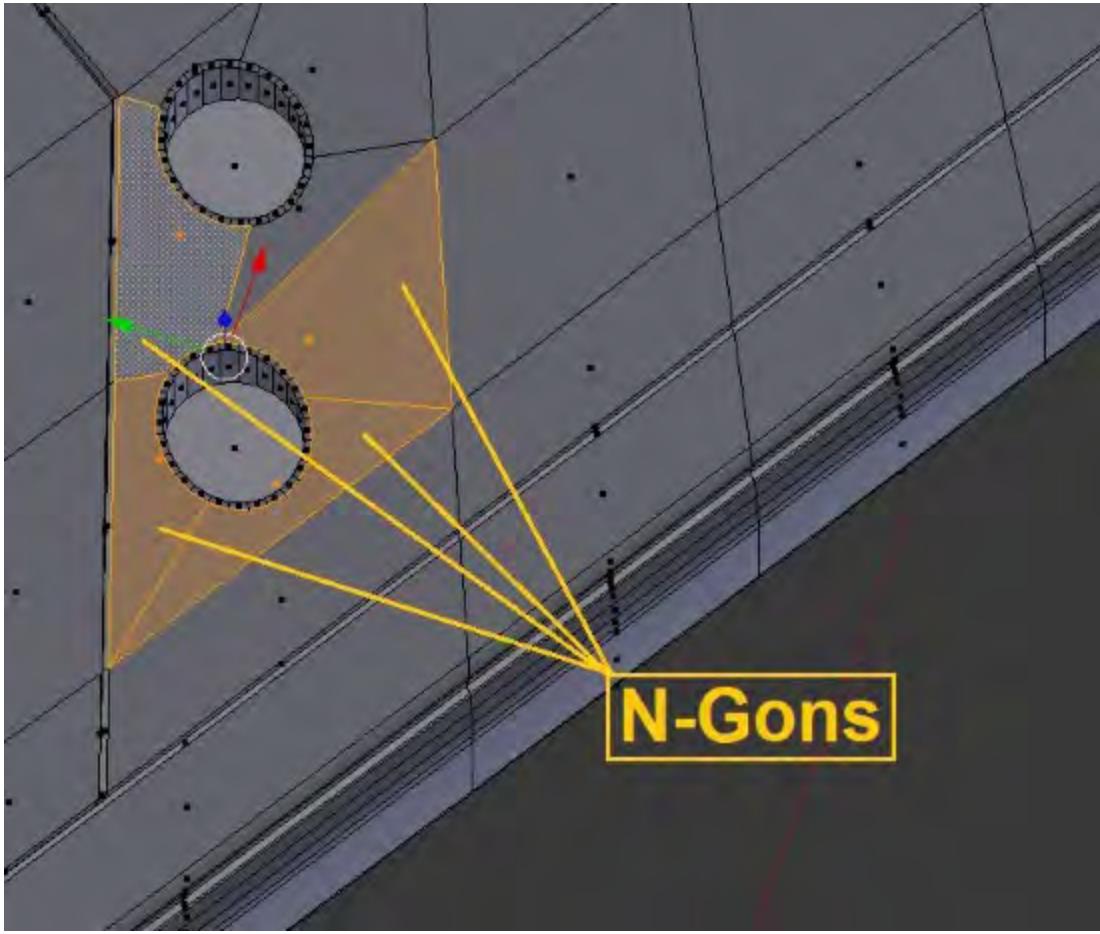


Using the same techniques from , Sci Fi Pistol – Adding Details, we'll cut those pieces into our hull:



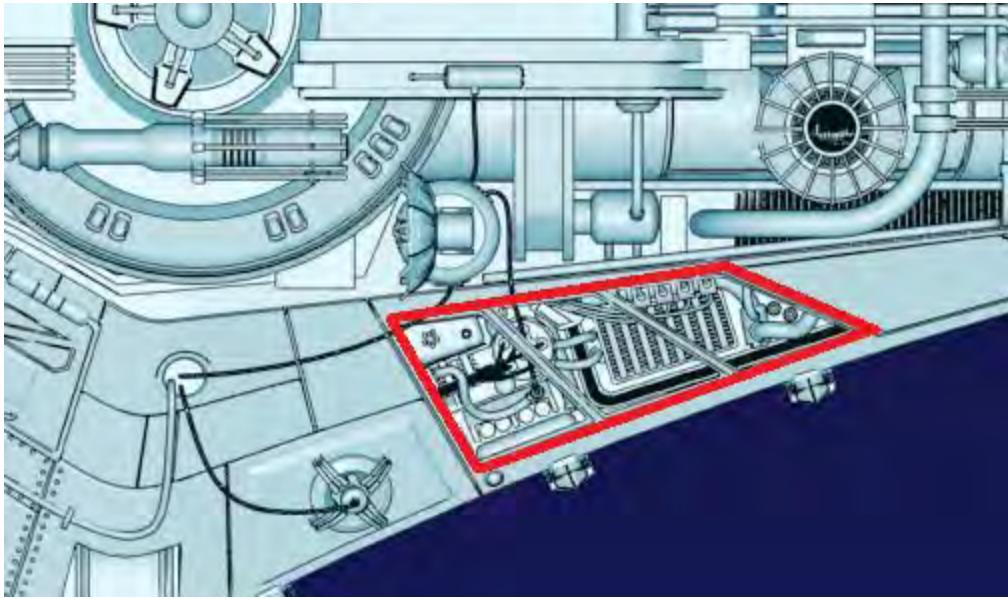
<pagebreak></pagebreak>

I'd like to point out that I've cheated a little bit here. Earlier in the book, I said you should never use N-Gons in curved areas. That's an excellent rule of thumb, but there are times when you can get away with it. For instance, I've done it here:



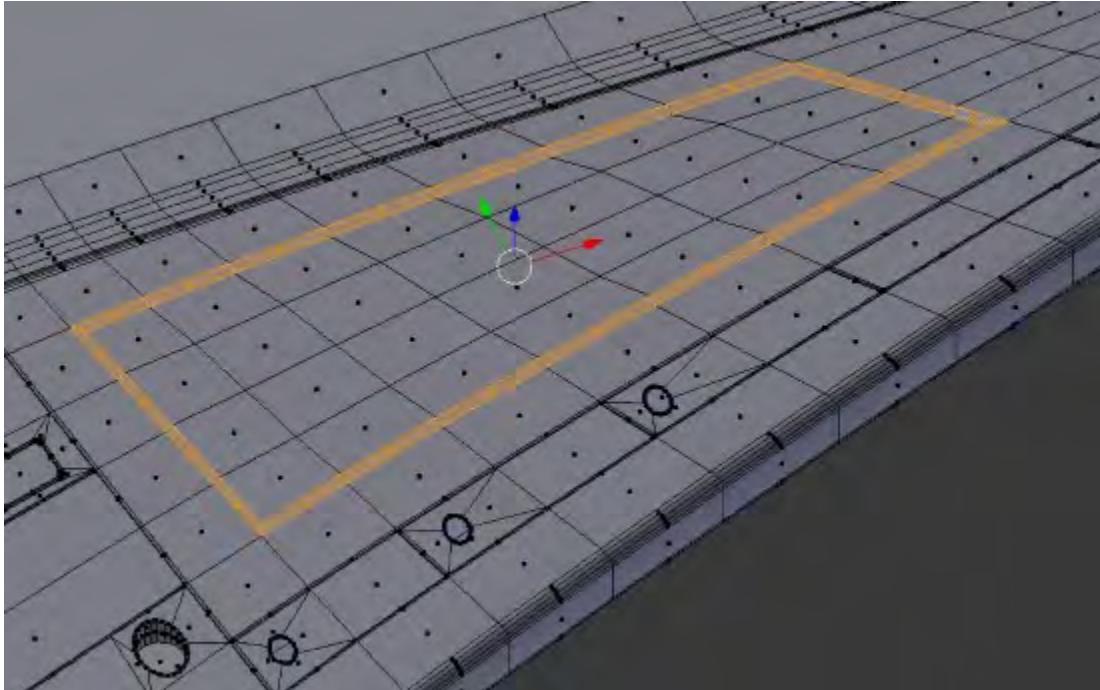
That hull area is technically curved, but it's so close to being flat and the faces are so small that there's no noticeable distortion-it certainly saves a lot of time. This is really a judgment call.

Next, we'll create some space for our mechanical area near the back:

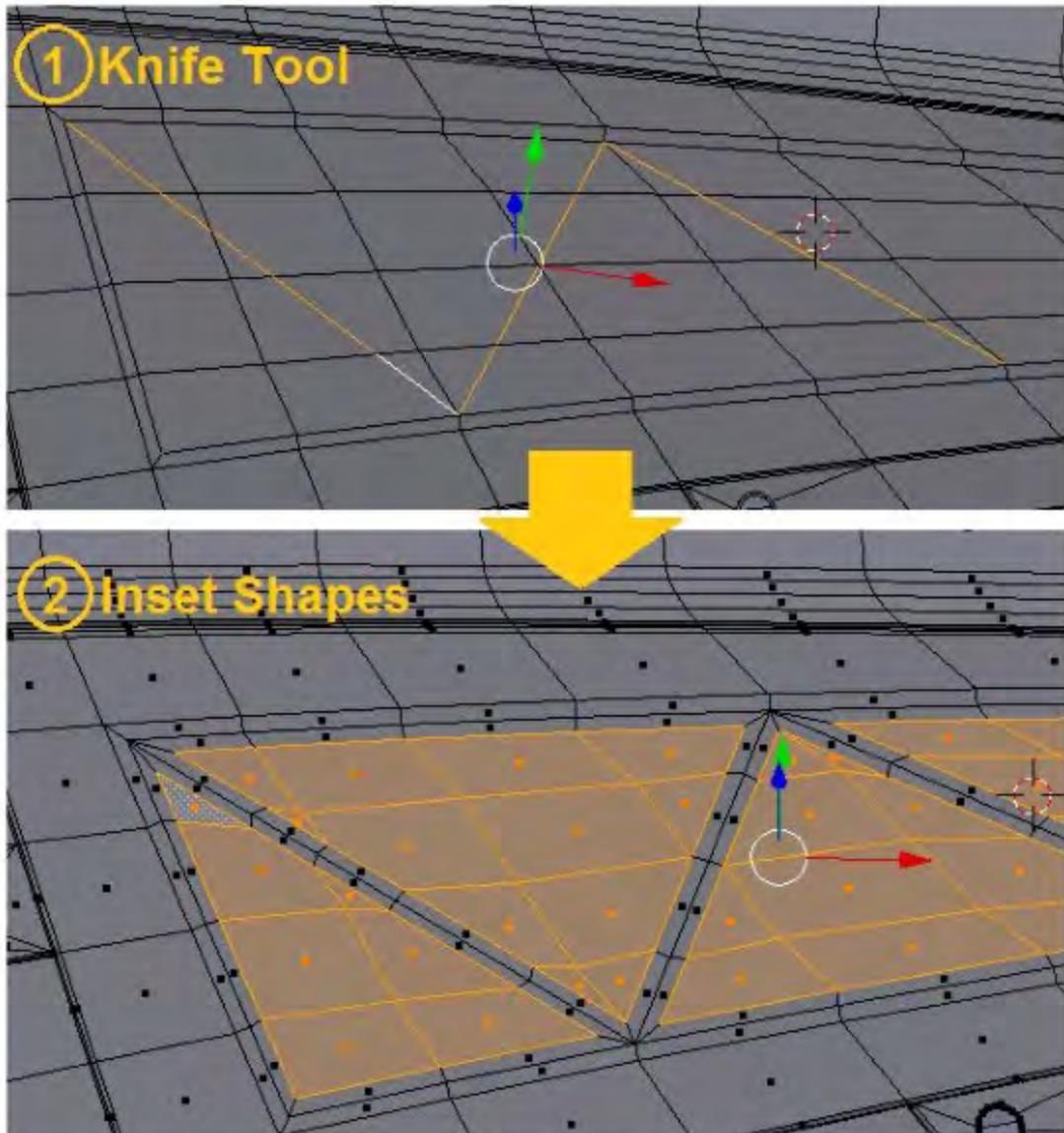


<pagebreak></pagebreak>

We can do this by just grabbing a good area of faces and using the **Inset** tool again:

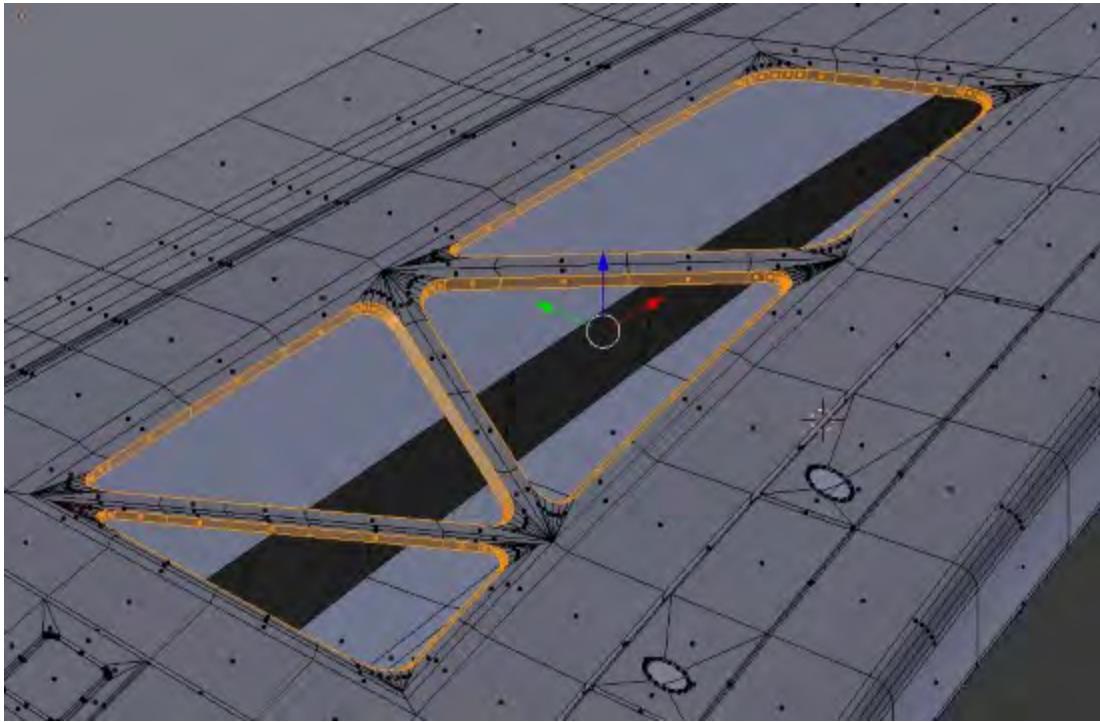


With my Knife tool, I can now create some nice diagonal cuts in those panels:

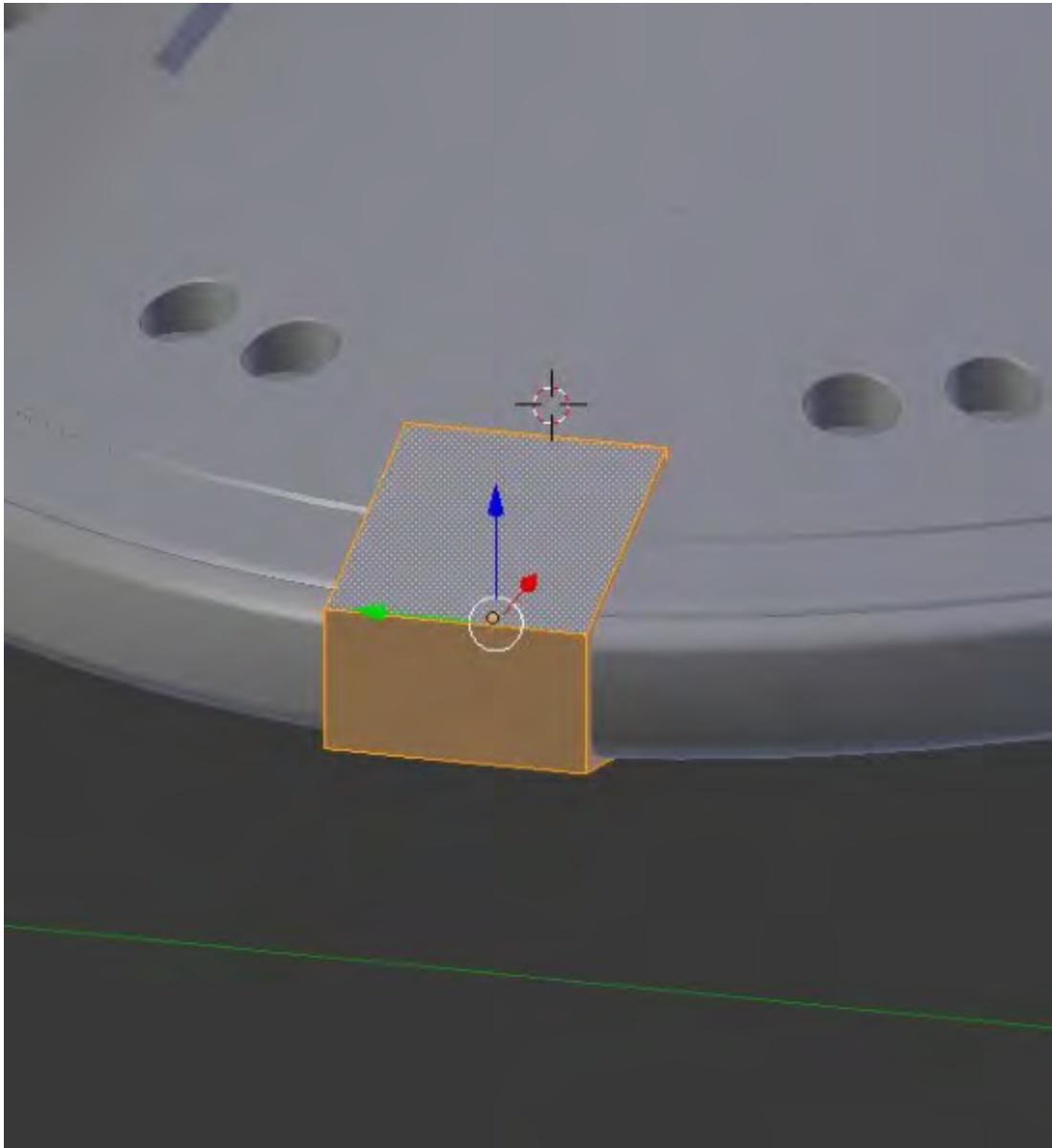


<pagebreak></pagebreak>

Using beveling again, I'll round out those edges:

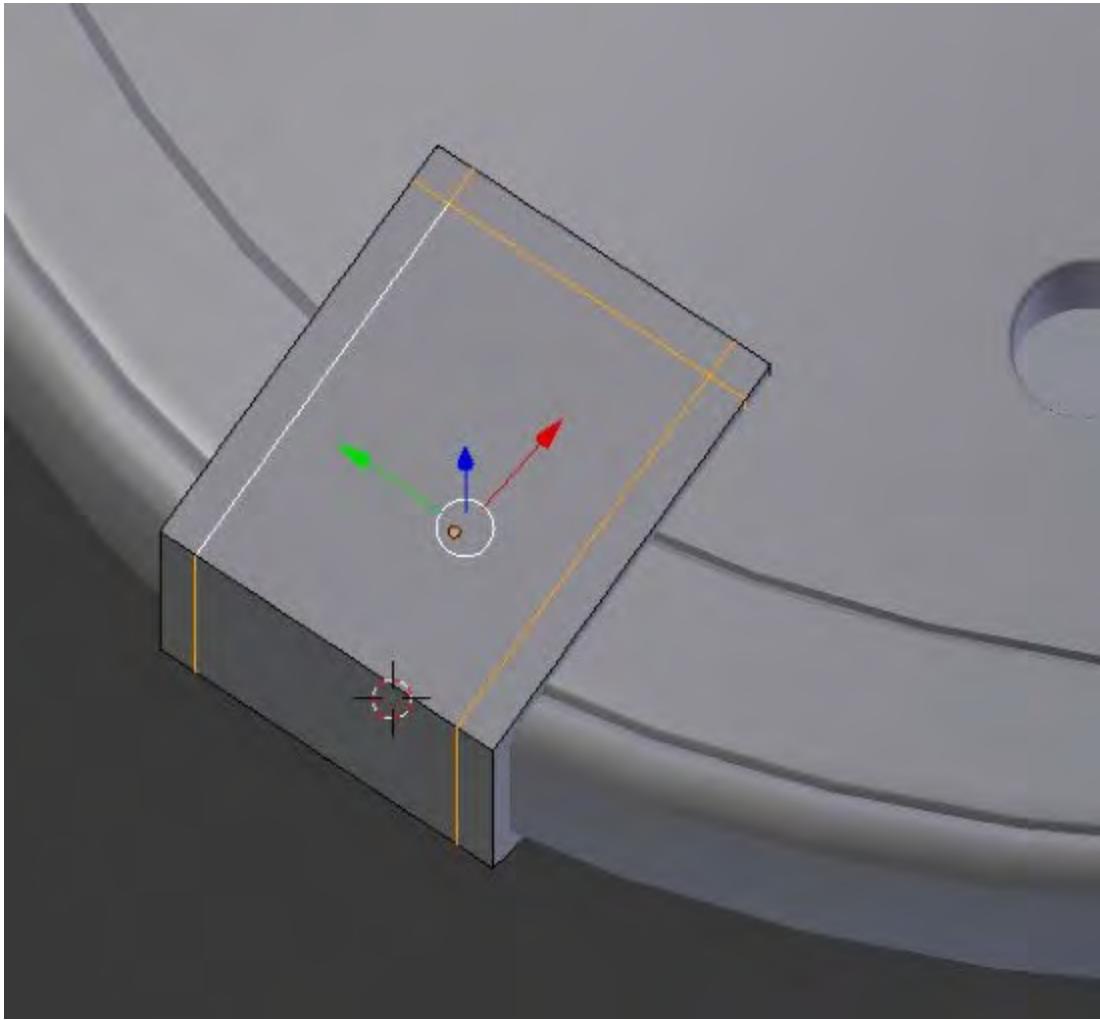


Next, we'll fill in those open gaps for the nose and side-mounted equipment. Start by adding a new **Cube** object and moving it into position:



<pagebreak></pagebreak>

Then, we can run some loop cuts where we want the borders to be:

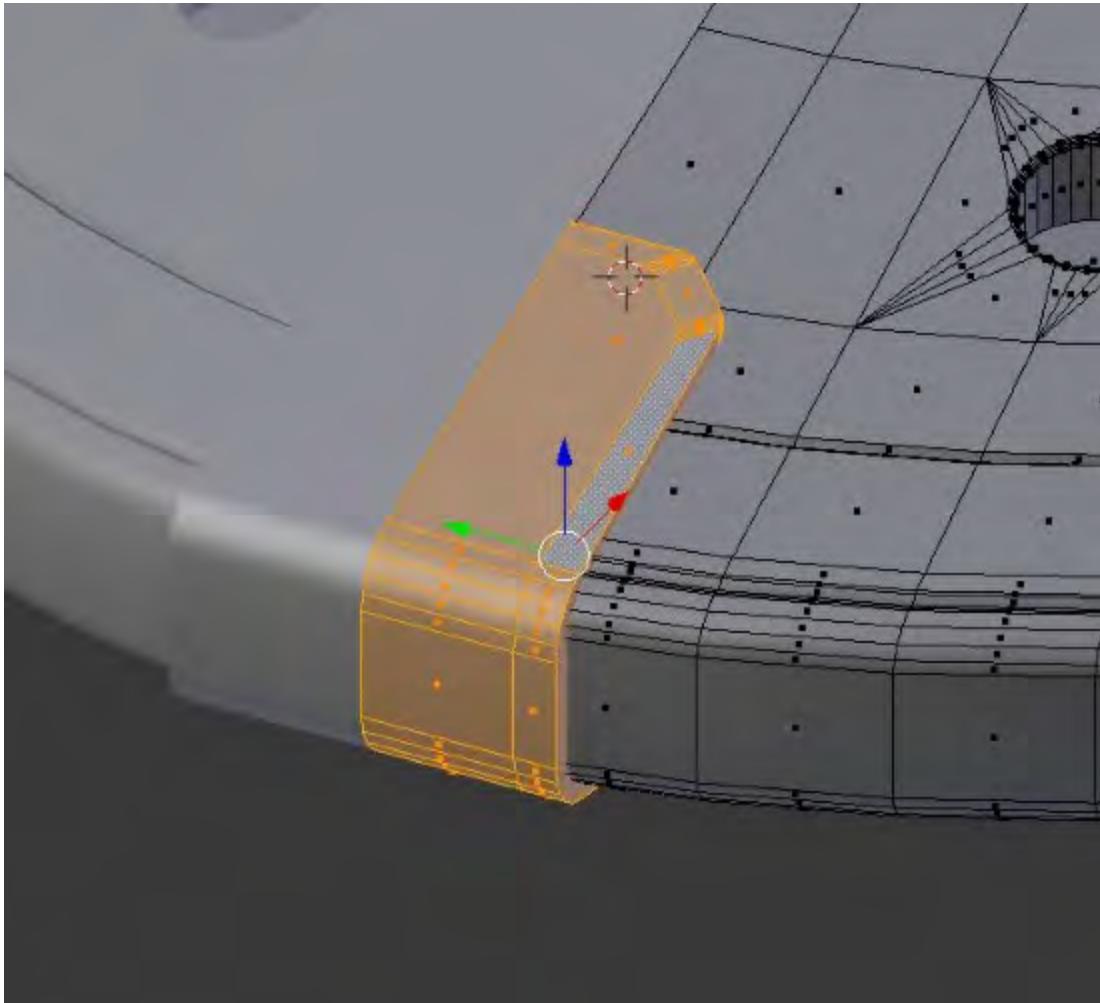


We can then merge those corner vertices, so we get a nice slanted line for beveling:

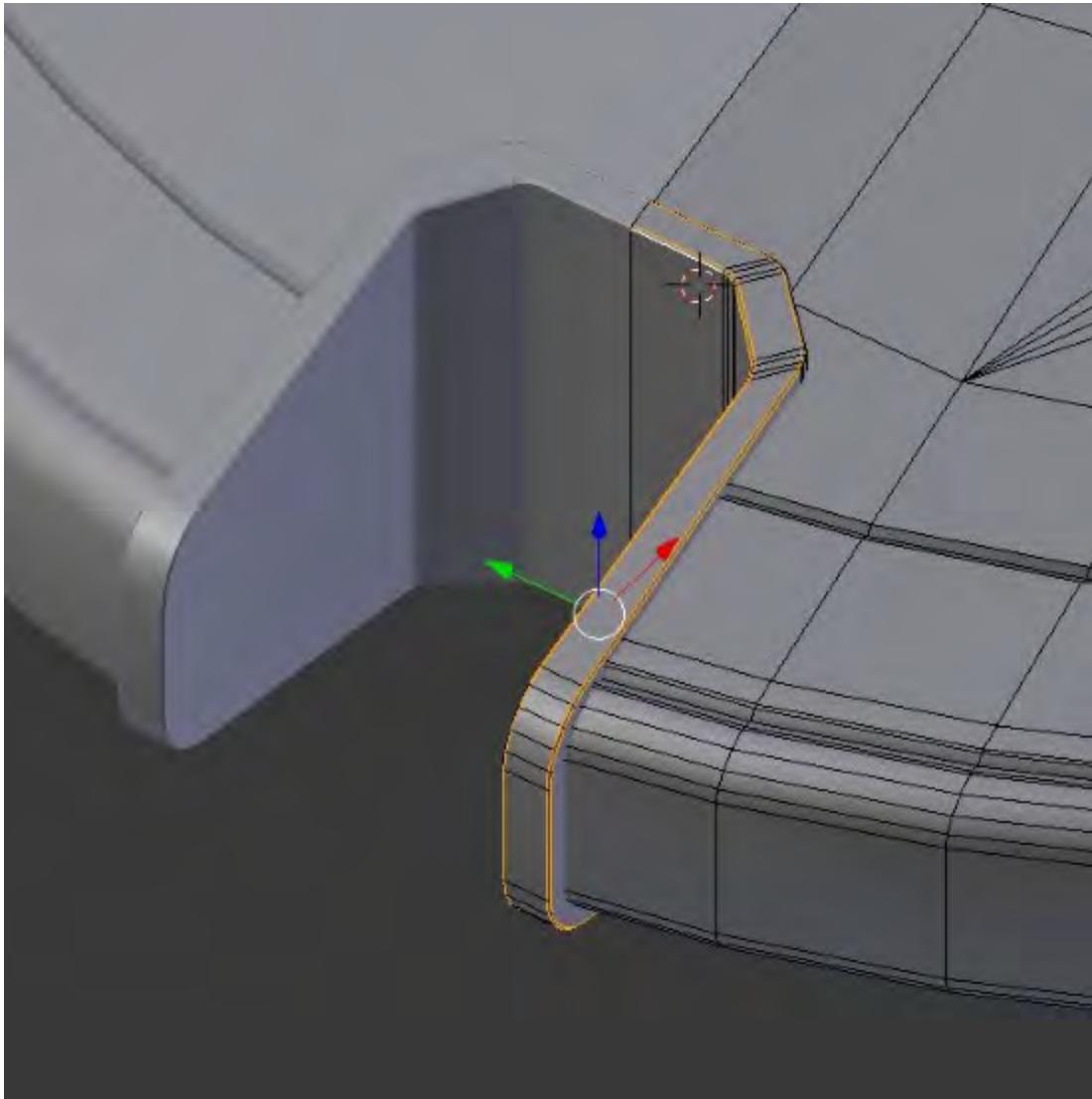


<pagebreak></pagebreak>

Now, we can use the bevel tool to create the shape we want. Then, we'll delete half of it (to make use of the Mirror modifier) and join it to our body:

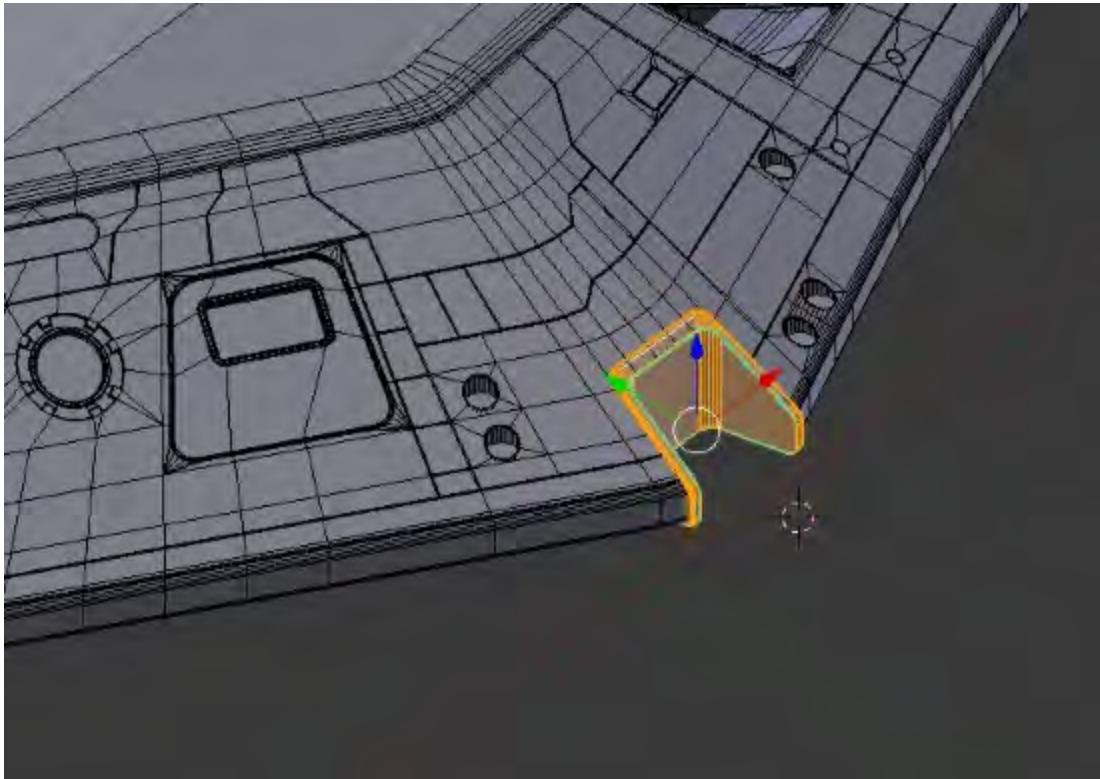


We can then delete the faces created inside the loop and bridge the top and bottom edge loops to create our final shape:



<pagebreak></pagebreak>

We'll use the same technique on the sides of the ship:



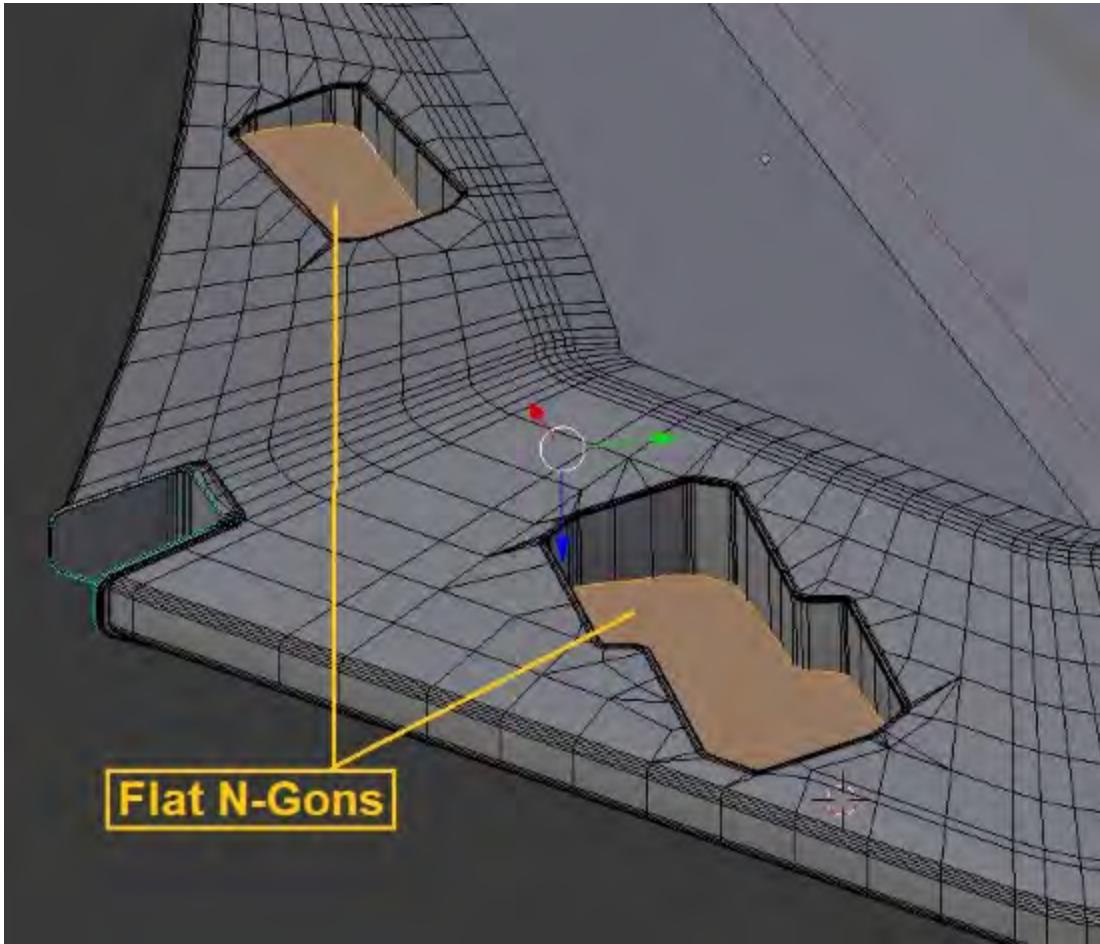
Even though the missile launcher is different from the grappling gun, I'm going to make these cutouts in the hull identical (just for simplicity's sake).

Next, we'll create some cutouts on the bottom for our landing gear:

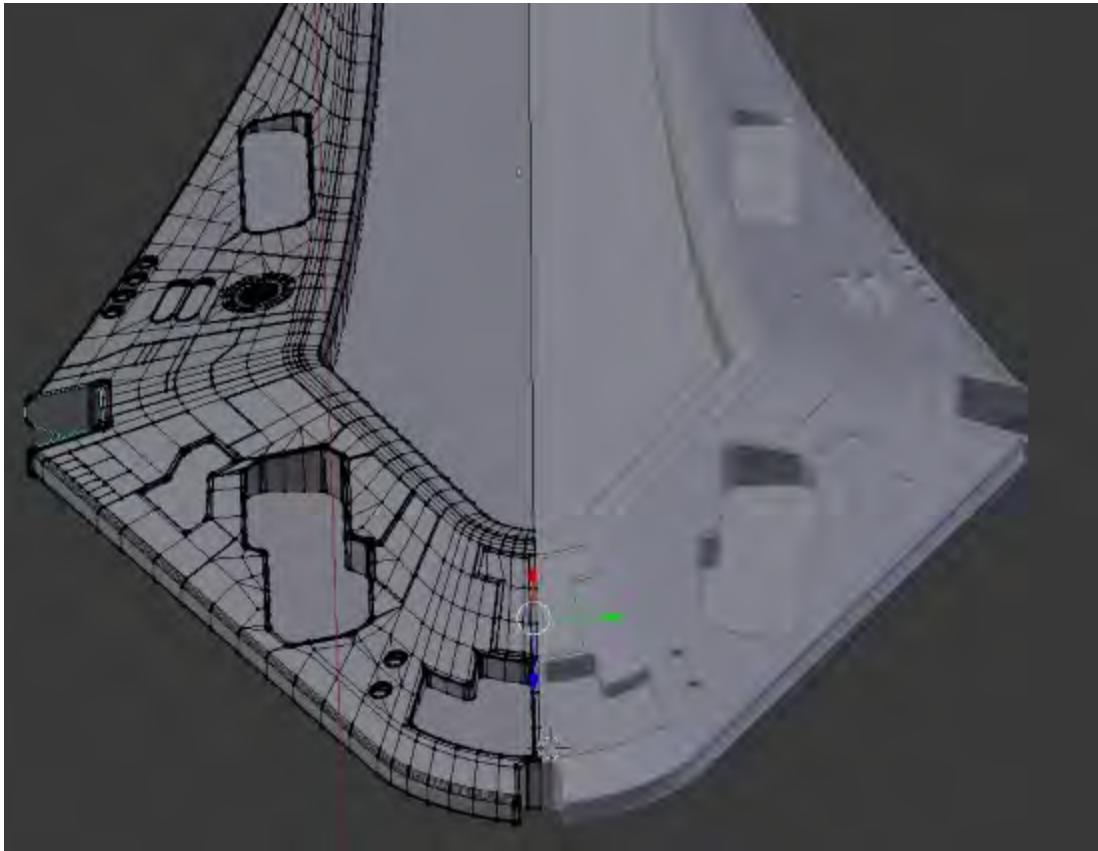


<pagebreak></pagebreak>

If you'd like, you could create actual doors for your landing gear. Judging by the rest of the ship, aerodynamics isn't much of a concern in the design. So, I think, we'll just create holes for them to retract up into:



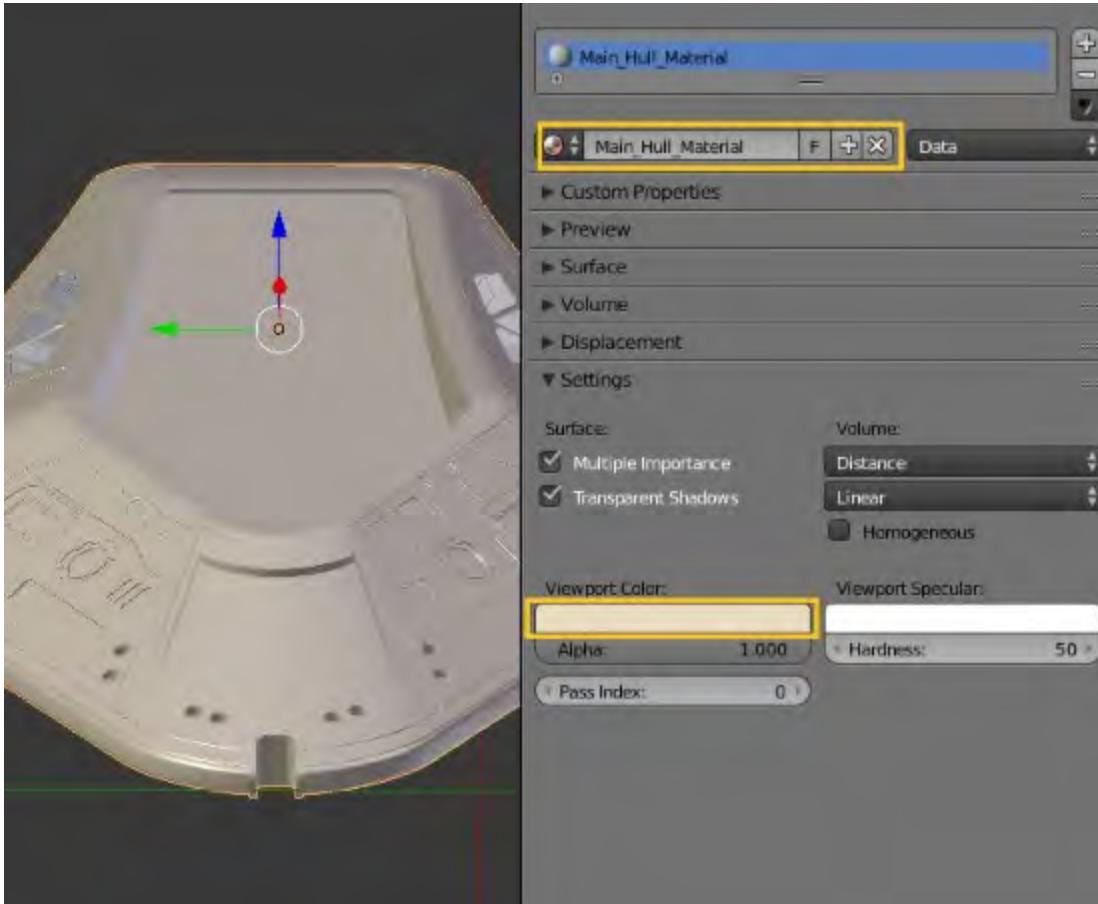
Using the same techniques from before, I'll create some cutouts in the bottom of the ship:



Even though we're not doing materials in this chapter, we are going to create slots for them. In other words, we'll create some materials that don't have any properties to them.

<pagebreak></pagebreak>

Here, I've just created one called **Main_Hull_Material** and assigned it a viewport color:

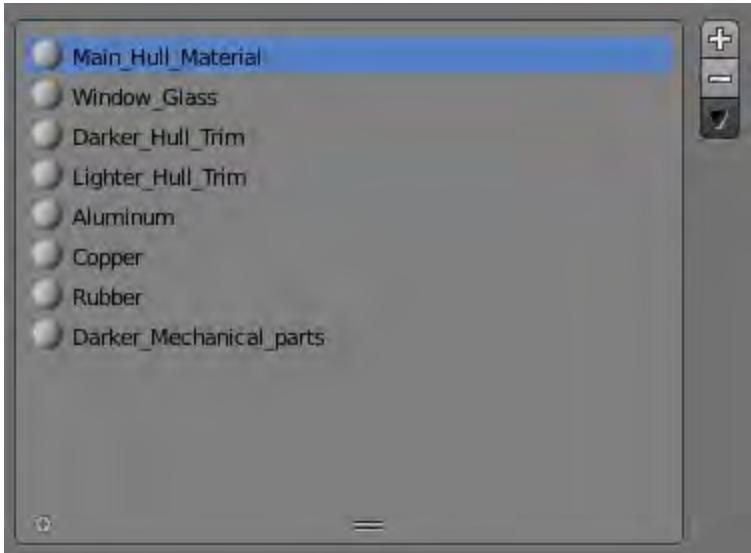


This way, I can assign materials as I model.

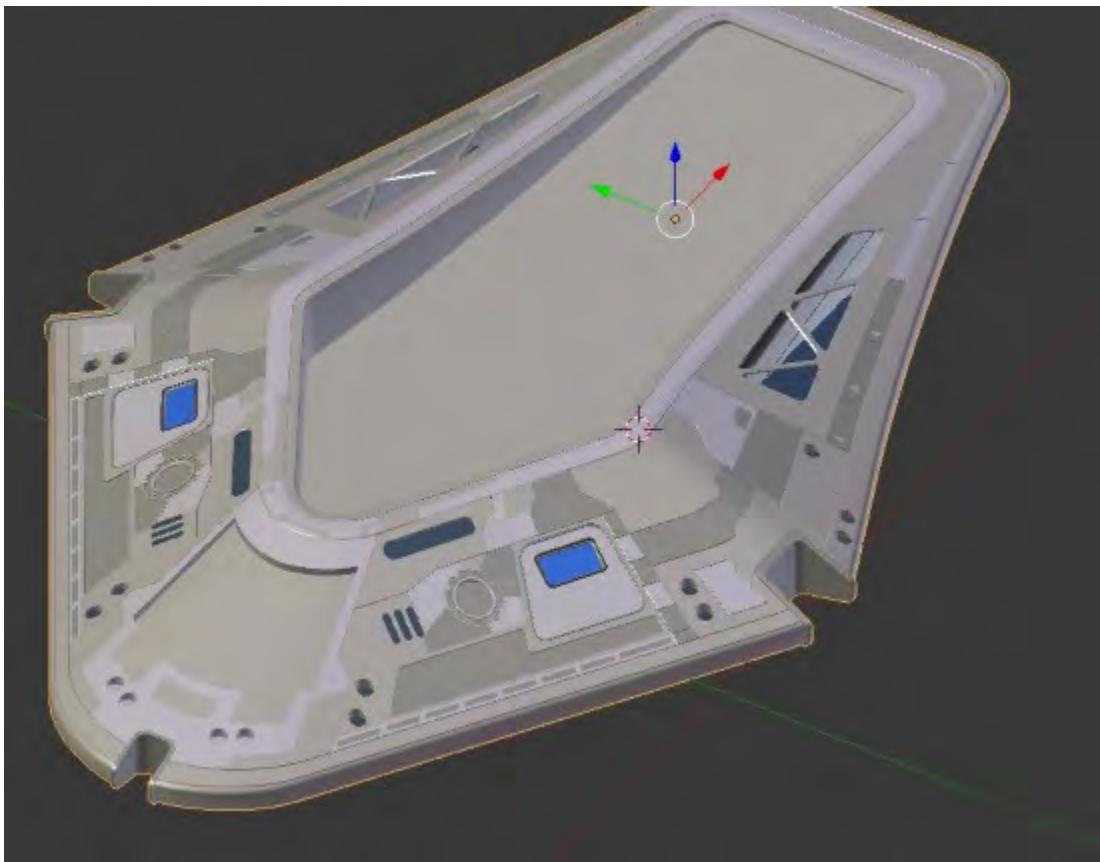
For example, I'll be adding some handles for people to grab onto during EVA operations. There are going to be 221 handles on my final ship model. It's possible to go through all of them and assign them a material later, but it's very inefficient. A much better solution is to just create one of them, assign it a material, and then duplicate it. In order to take advantage of this, obviously, we'll need to have some materials added to our ship ahead of time.

<pagebreak></pagebreak>

So, what I'm going to do is just add a bunch of materials to the ship now so that we can assign them as we go:



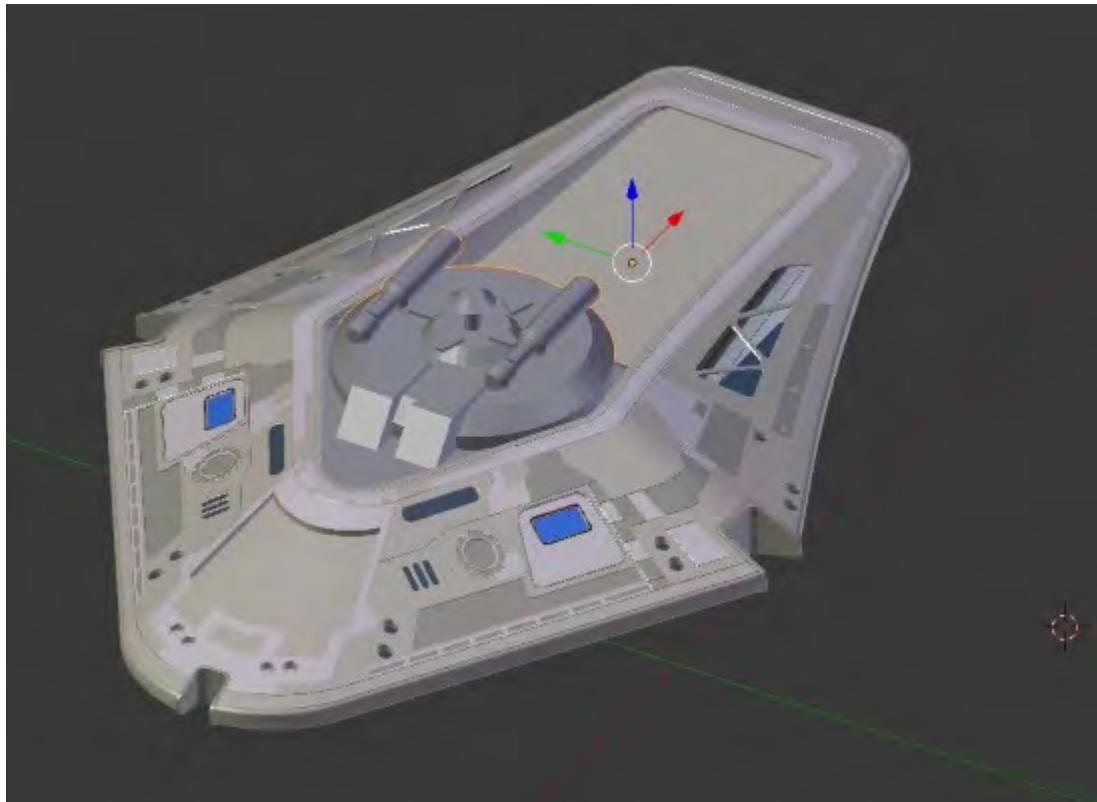
I've given them all different viewport colors so that we can tell them apart:



I think we're at a good stopping point now with the body. We'll probably want to come back to it and work on it later, but I'd like to get the rest of the model up to this same level of detail.

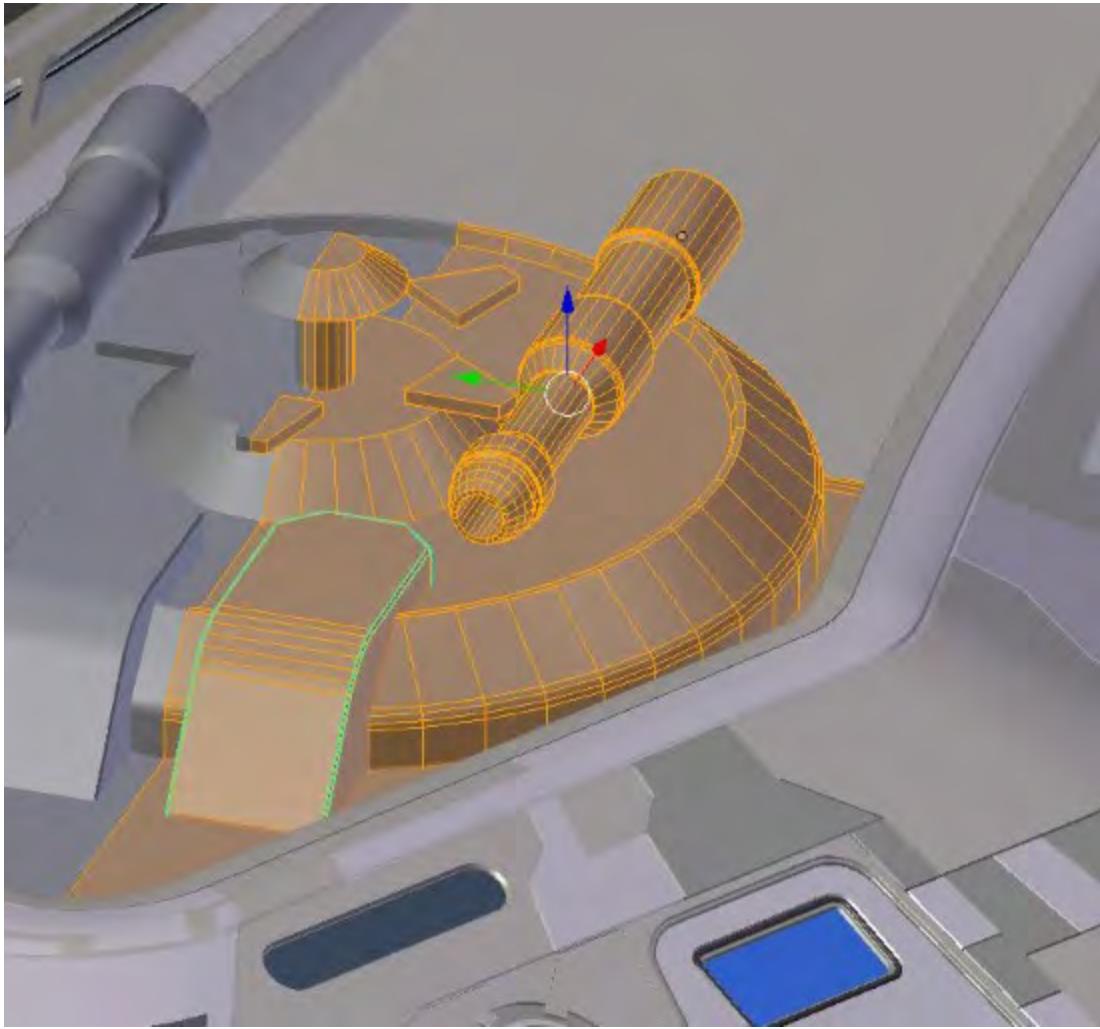
It's generally a good idea to work like this—adding detail in layers instead of finishing one part completely while another is still blocked out.

So, let's go to our other layer (layer 2, in my case) and bring that top section back to the same layer as our body:



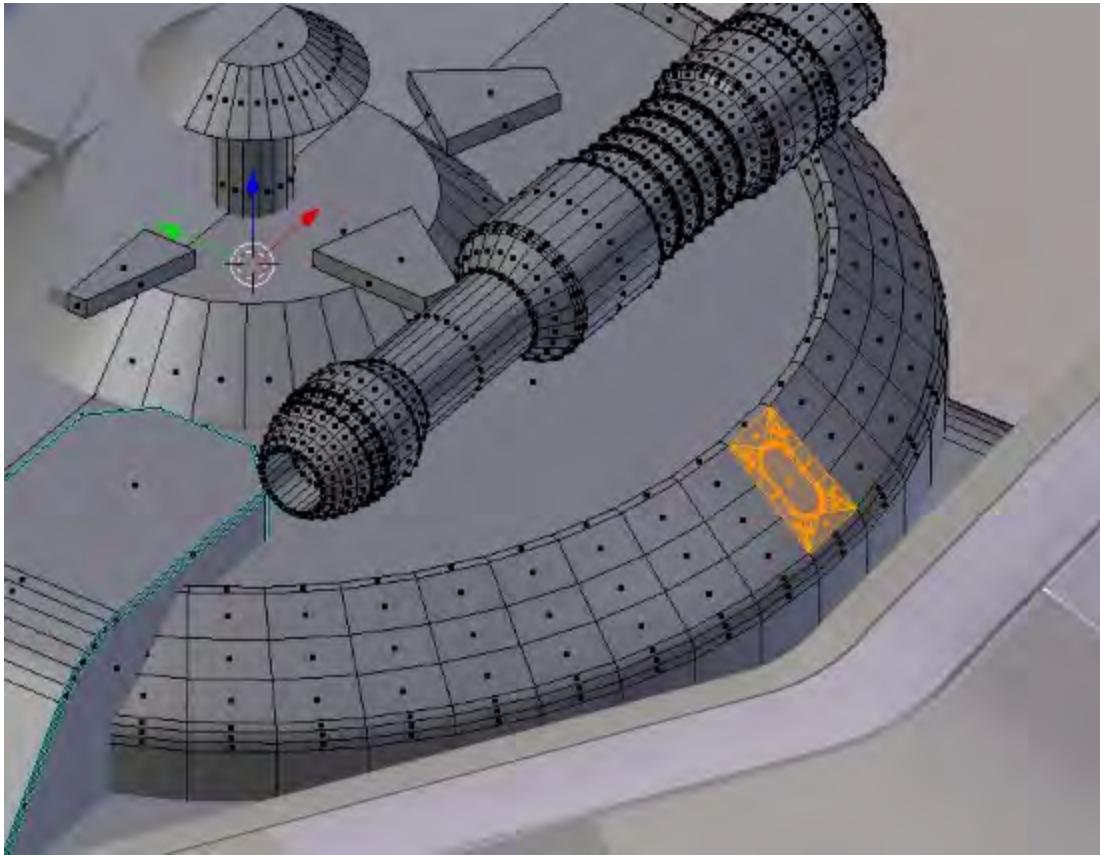
<pagebreak></pagebreak>

First, I'll just do some basic beveling to refine the shape a bit:

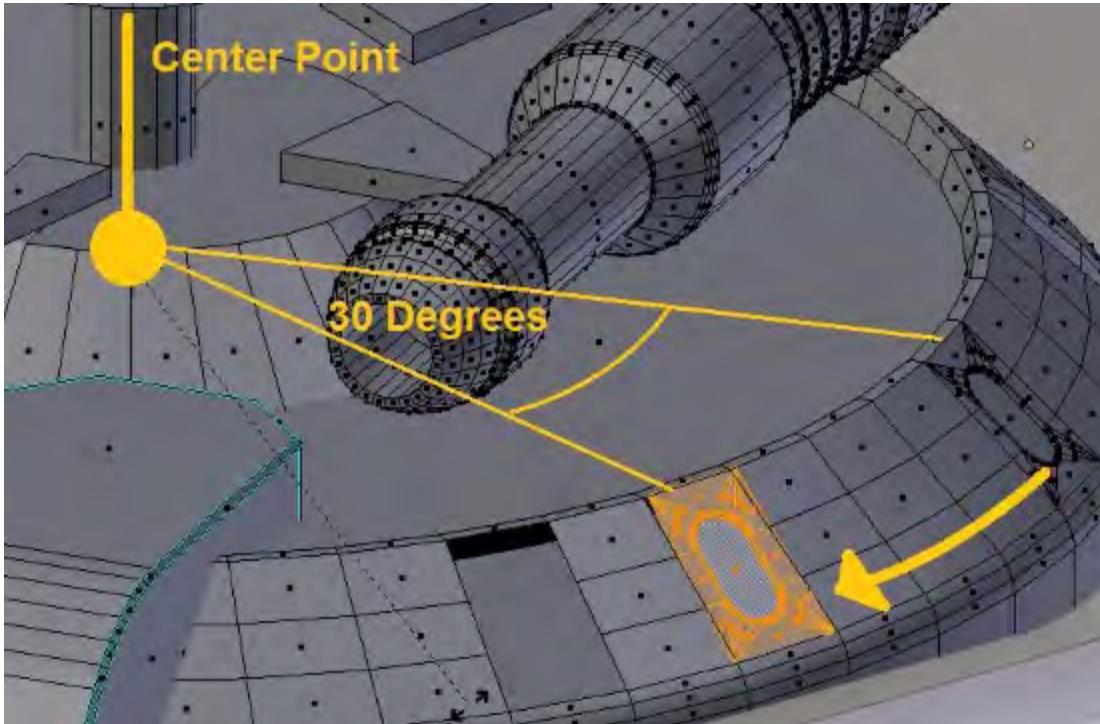


<pagebreak></pagebreak>

Again, we'll use a **Shrinkwrap** modifier to add cutouts. In this case, I'll just start with one:

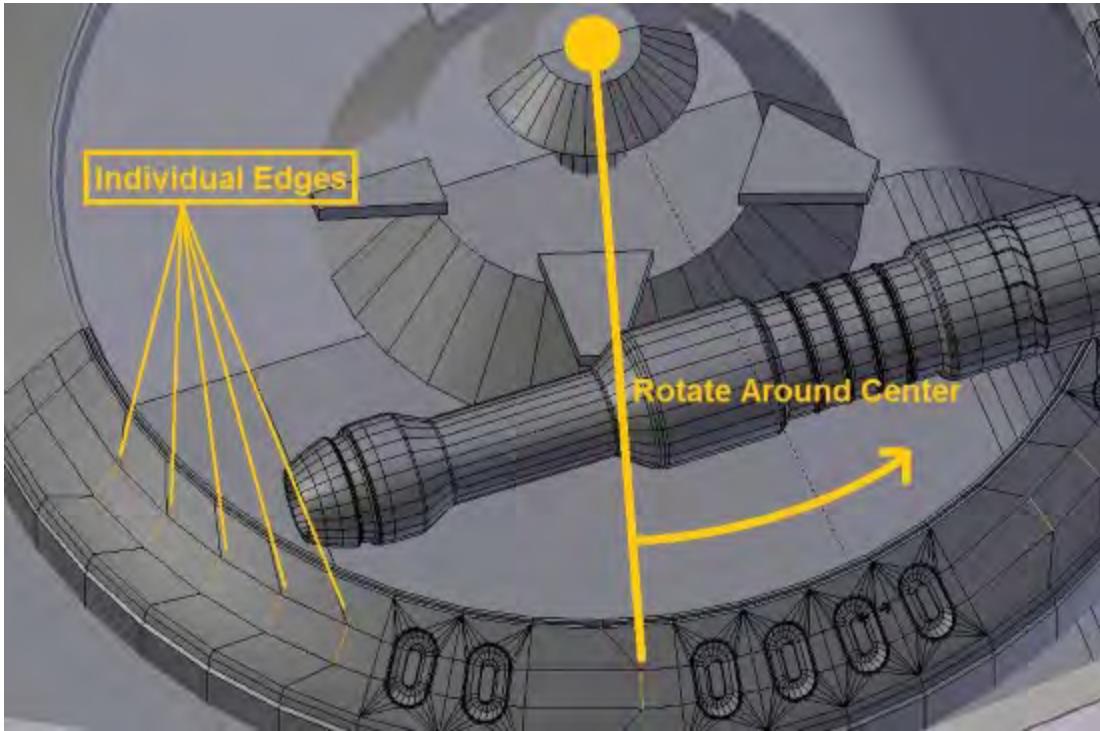


You can then set your **3D Cursor** to the center of that circle and rotate the cutout piece into position (you'll have to delete the original geometry first). Once you do that, of course, you'll need to remove the doubles you created. This saves you the trouble of cutting all those shapes in one at a time:

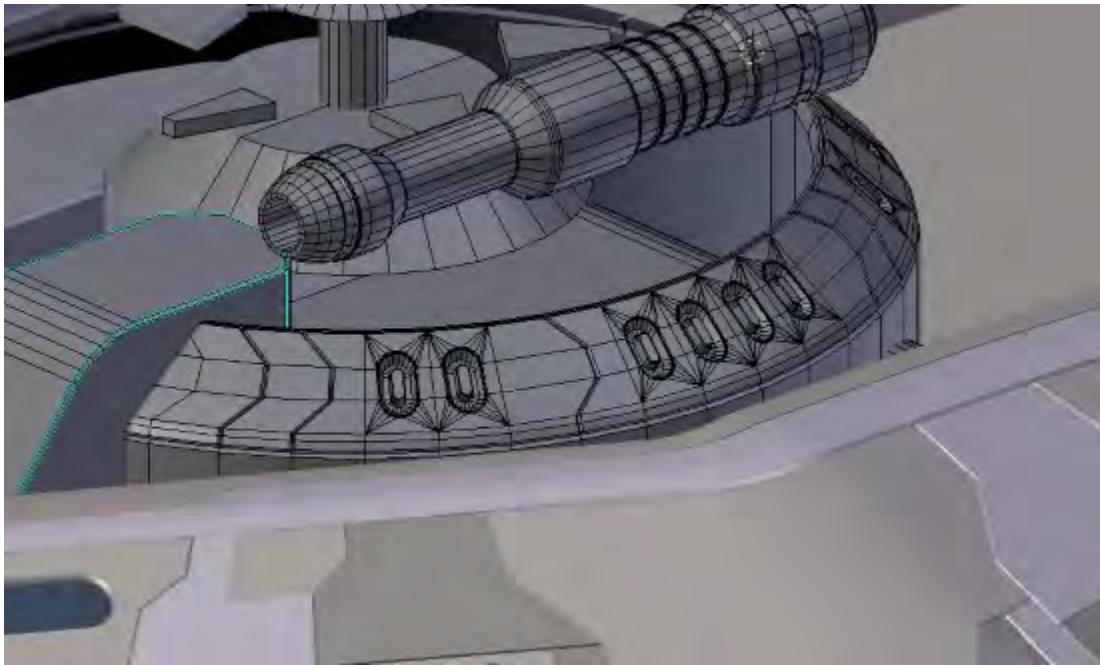


<pagebreak></pagebreak>

Just to add a little detail, I'm going to grab some individual edges and rotate them around the center point as well:

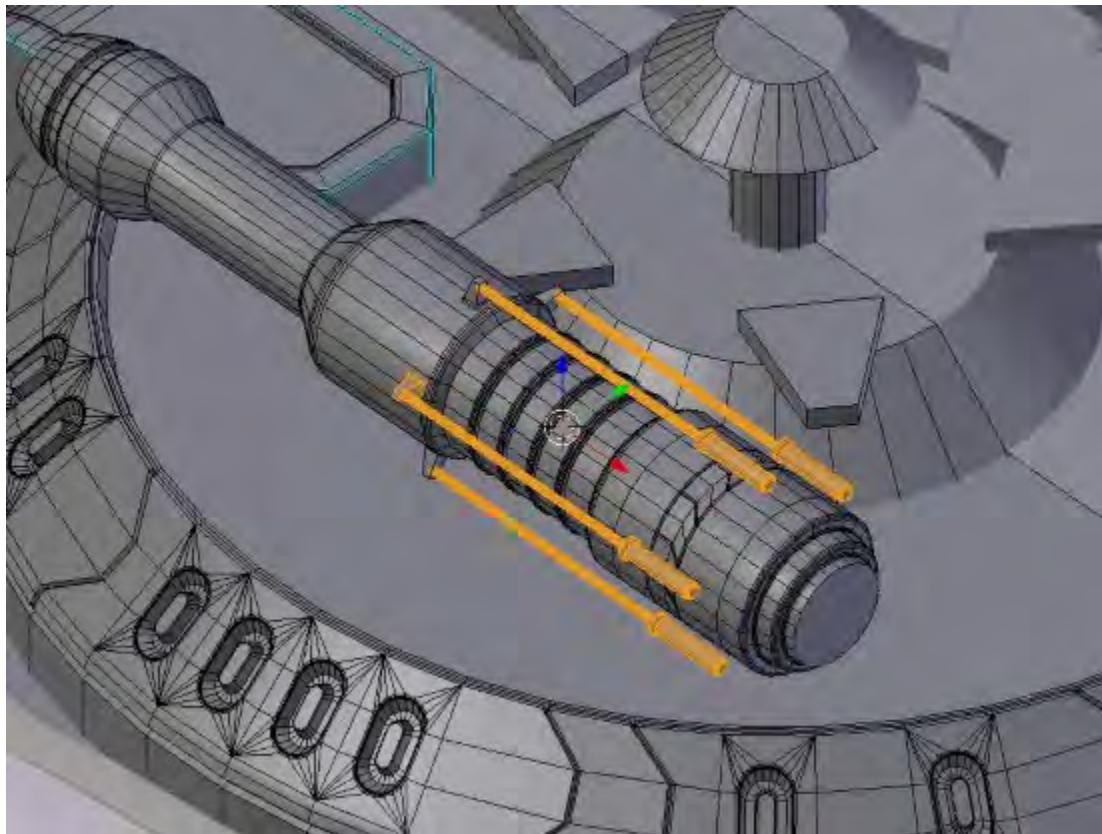


Once we bevel them, I think it looks pretty neat:

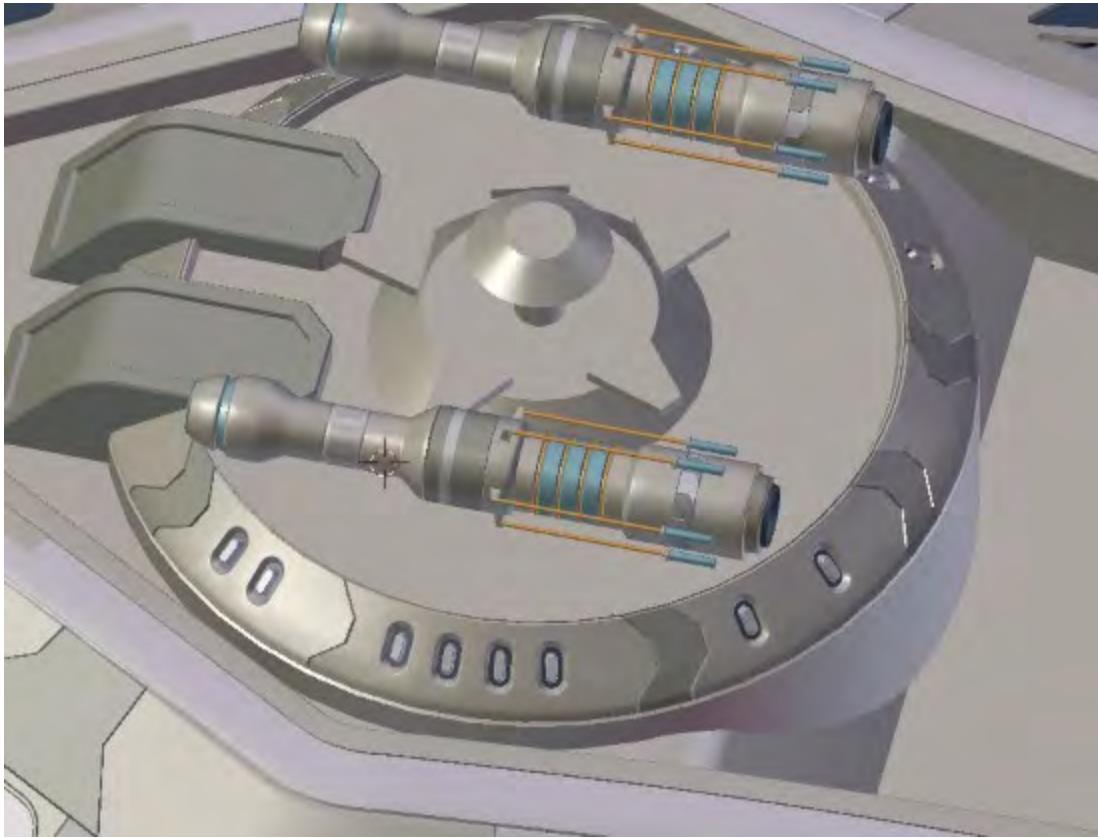


<pagebreak></pagebreak>

Next, I'll just add a little detail to the main cylinders on top. I'm not really sure what these are. They could be weapons, antennas, tractor beams, or something else altogether. It's up to you!

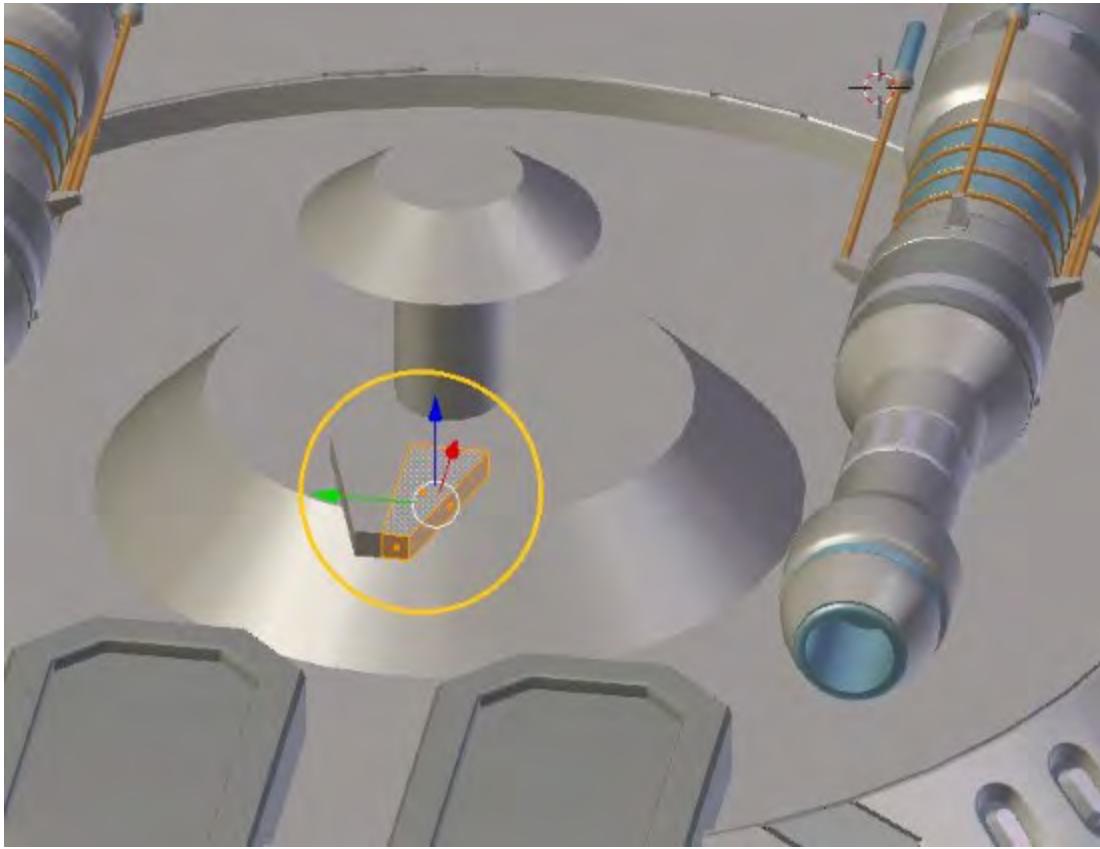


At some point, you'll want to join the top section with the main body. Once you do, you can assign some materials to the different parts:

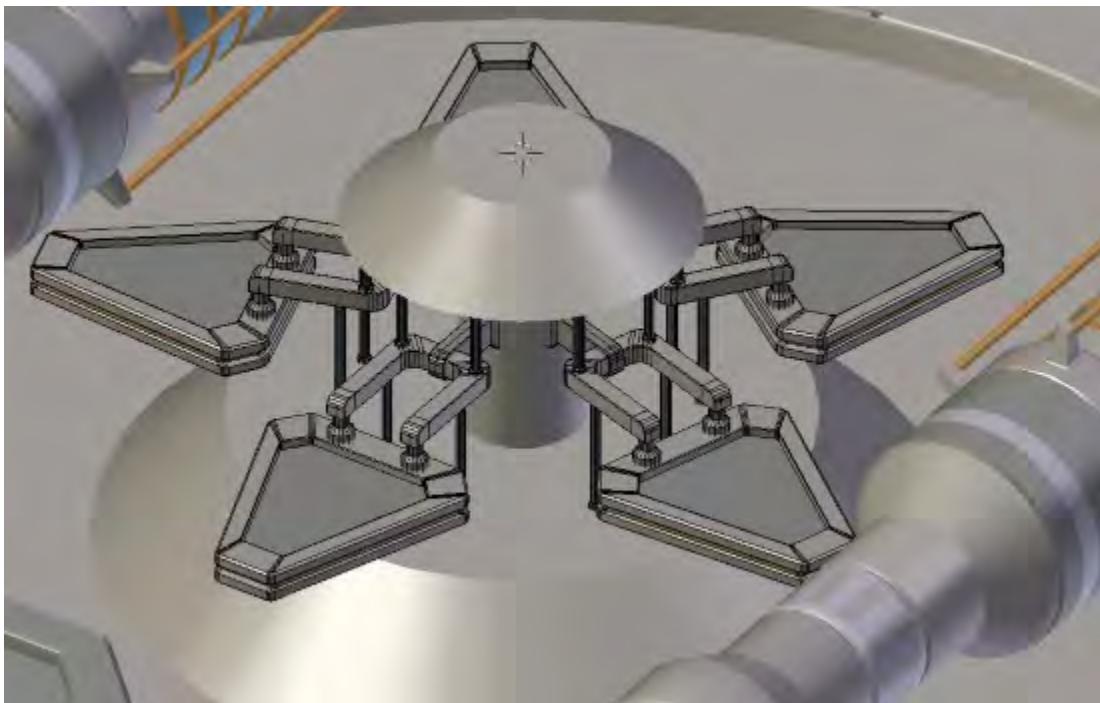


<pagebreak></pagebreak>

Next, we'll work on this small antenna array near the front. There were five of these originally, but I've deleted the rest of them and separated this piece from the main model:



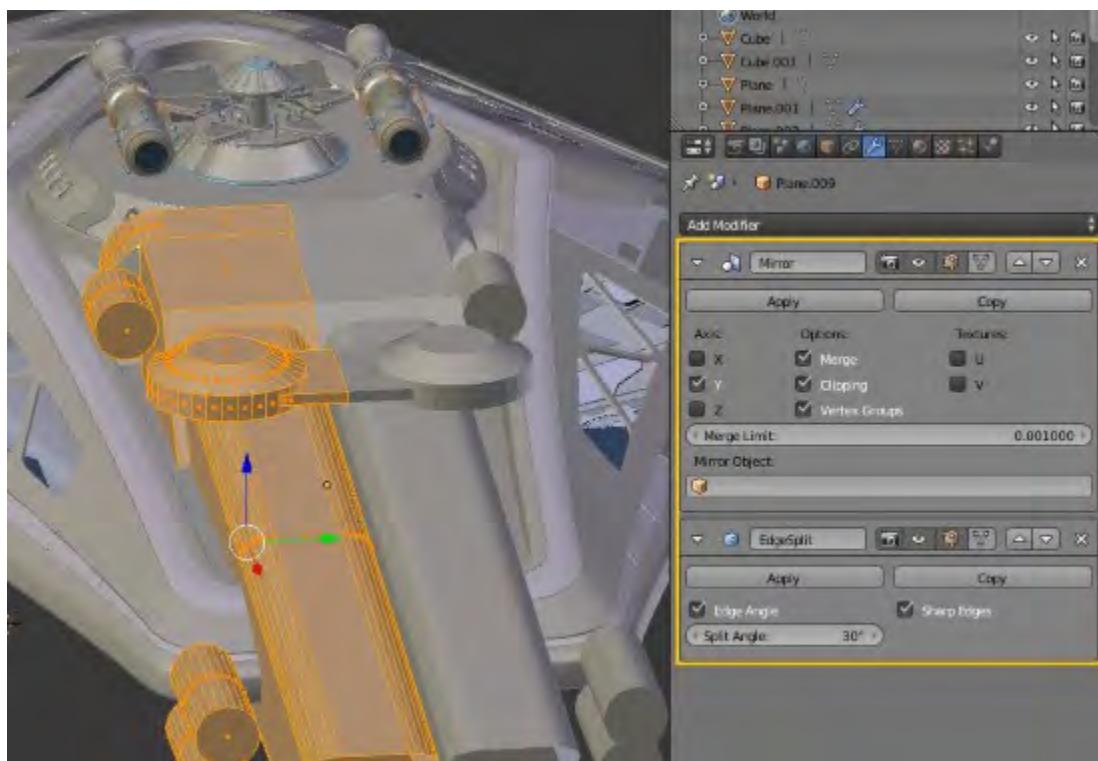
Now, I can add details and materials, and then duplicate and rotate it around the center point:



This is a good example of what I was talking about earlier—adding materials first, then duplicating. It tends to save a lot of time.

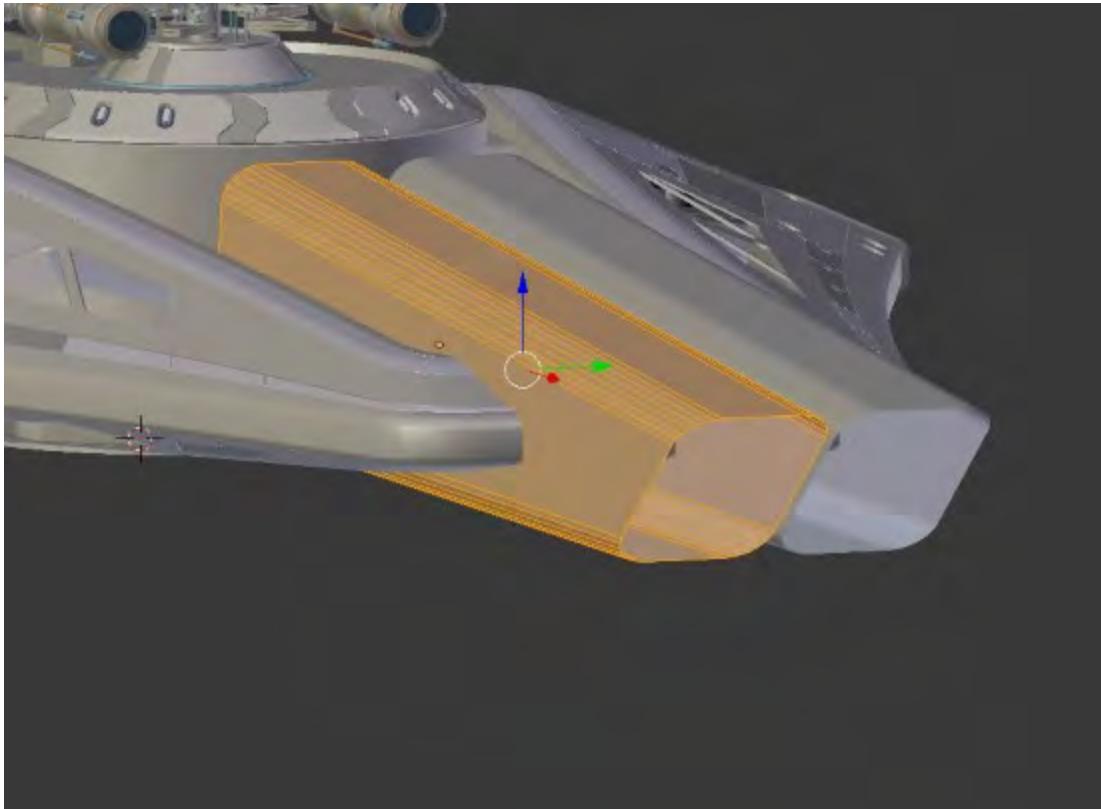
<pagebreak></pagebreak>

Next, we'll bring in the engine section. I'll delete half of it, then add a **Mirror** and **Edge Split** modifier. I'll also add our materials to the engine so that we can assign them as we go.

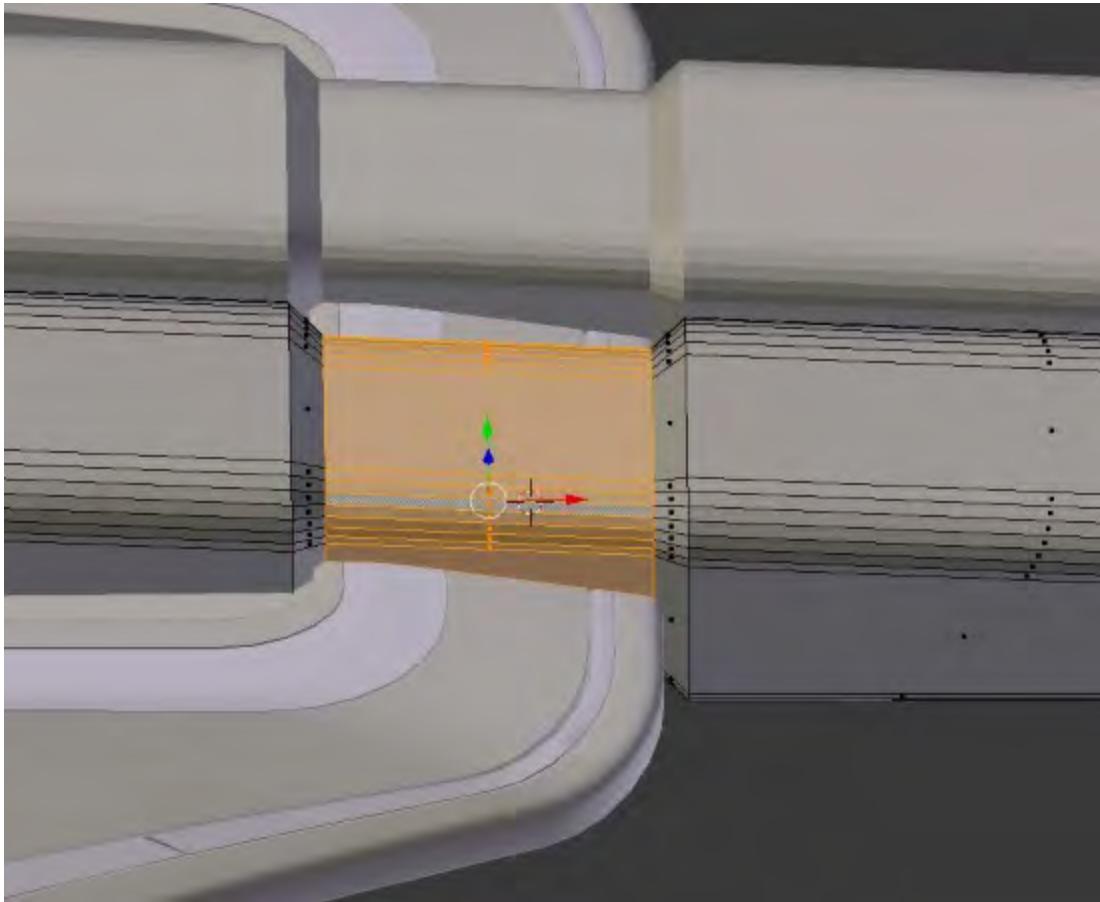


<pagebreak></pagebreak>

Let's get to work on the engines. I'll start with some basic beveling again:

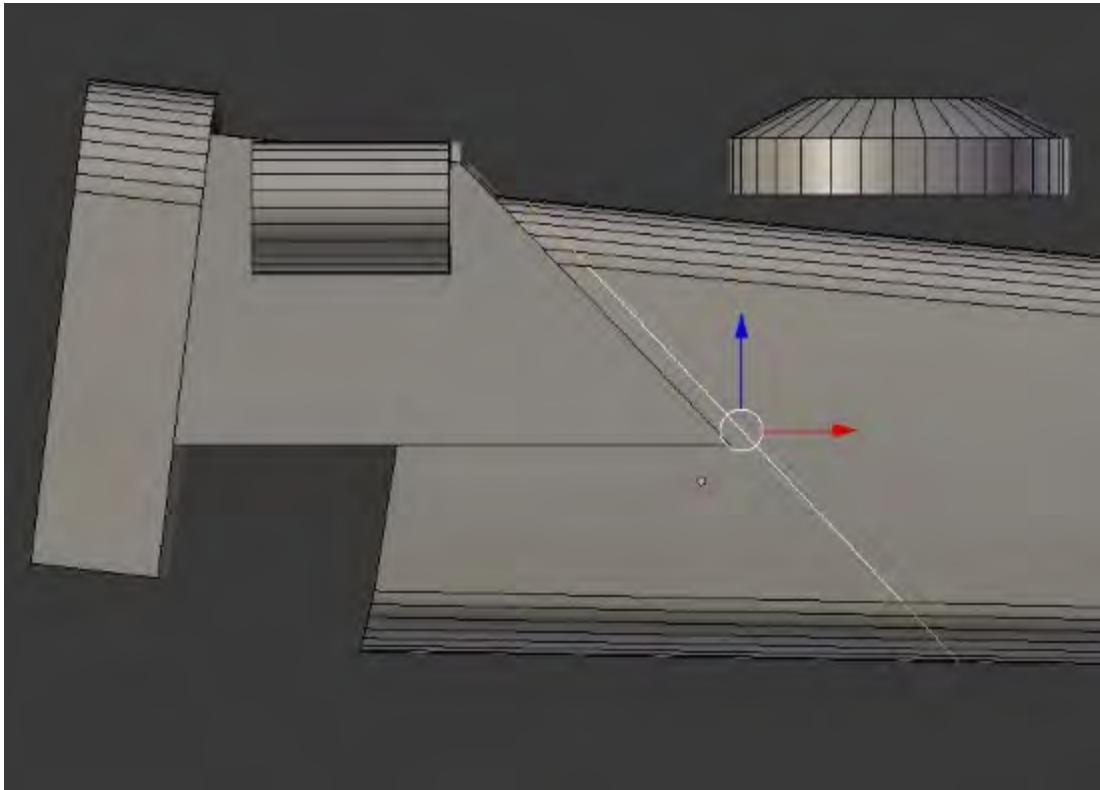


In the middle, we can add just a little detail:

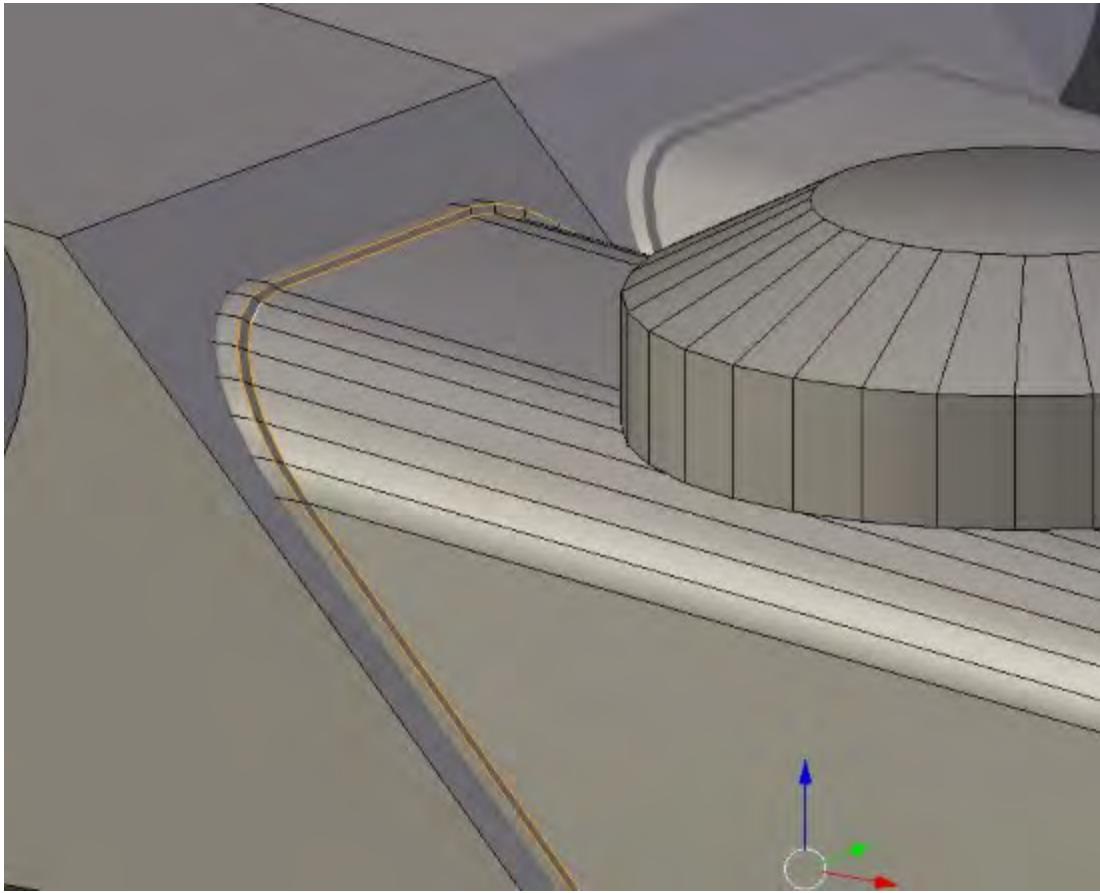


<pagebreak></pagebreak>

Next, I'll use the **Knife** tool to make a cut across the long part of the engine here:

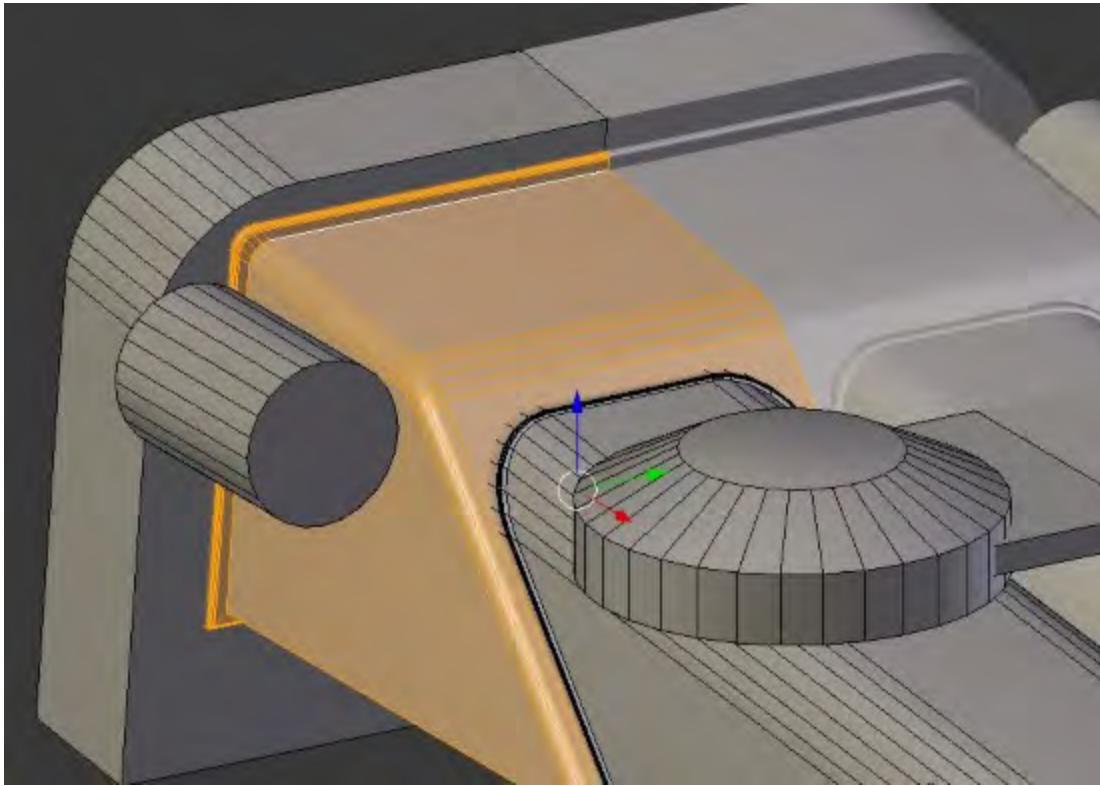


Then, I'll extrude those edges forward so that they make a nice little border:

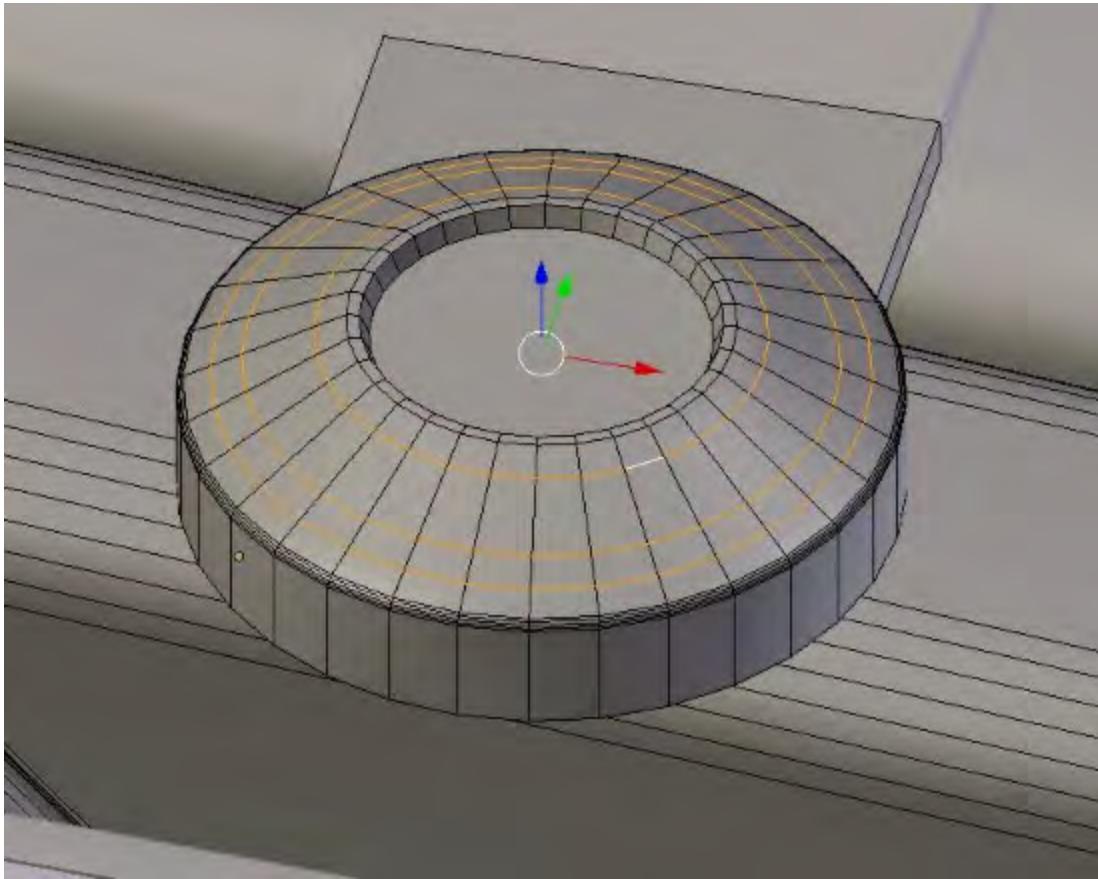


<pagebreak></pagebreak>

I'll do the same thing on the next section(right in front of it):

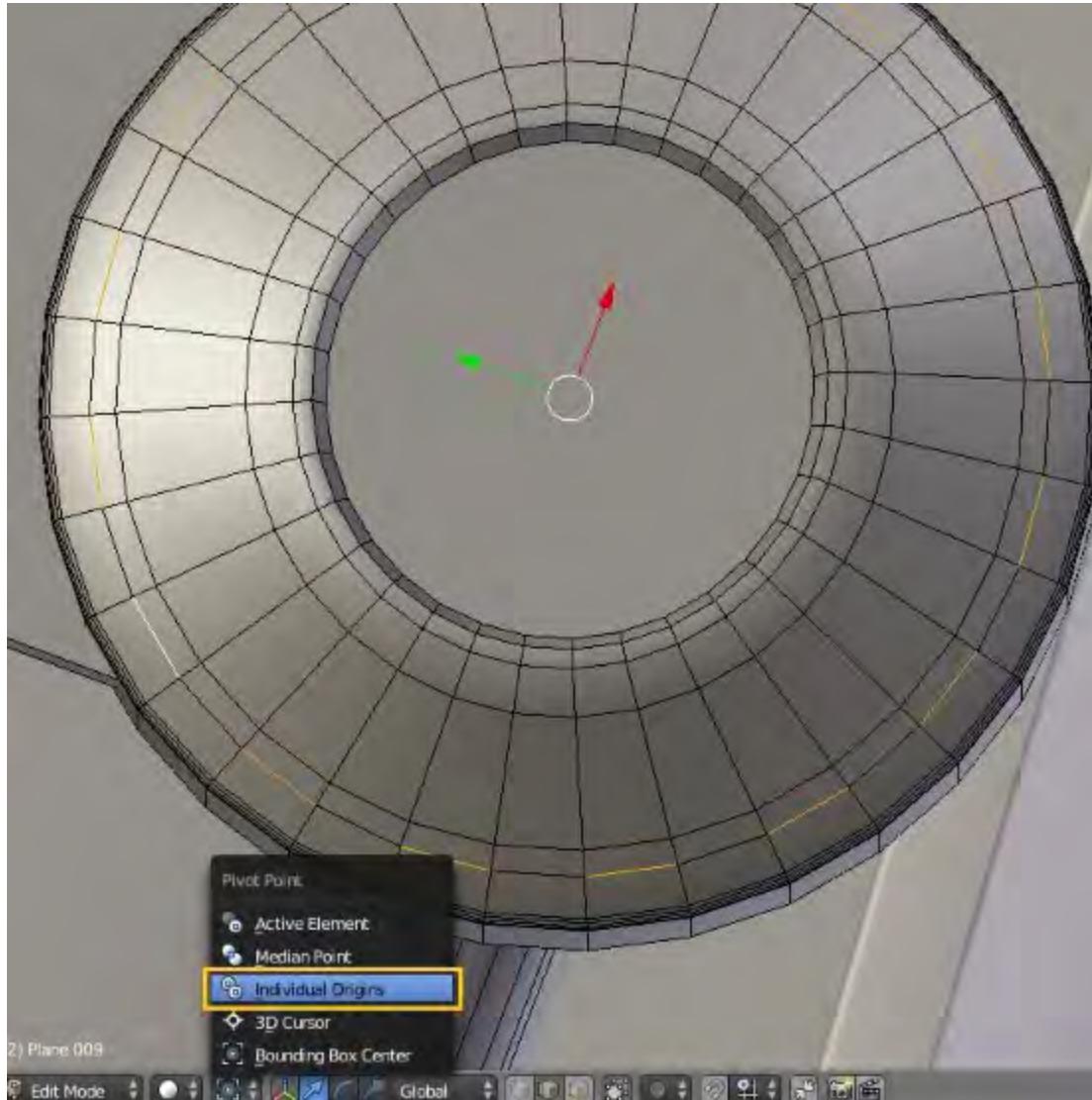


Next, we'll work on the circular parts above the engines. I'll start by adding a few loop cuts:



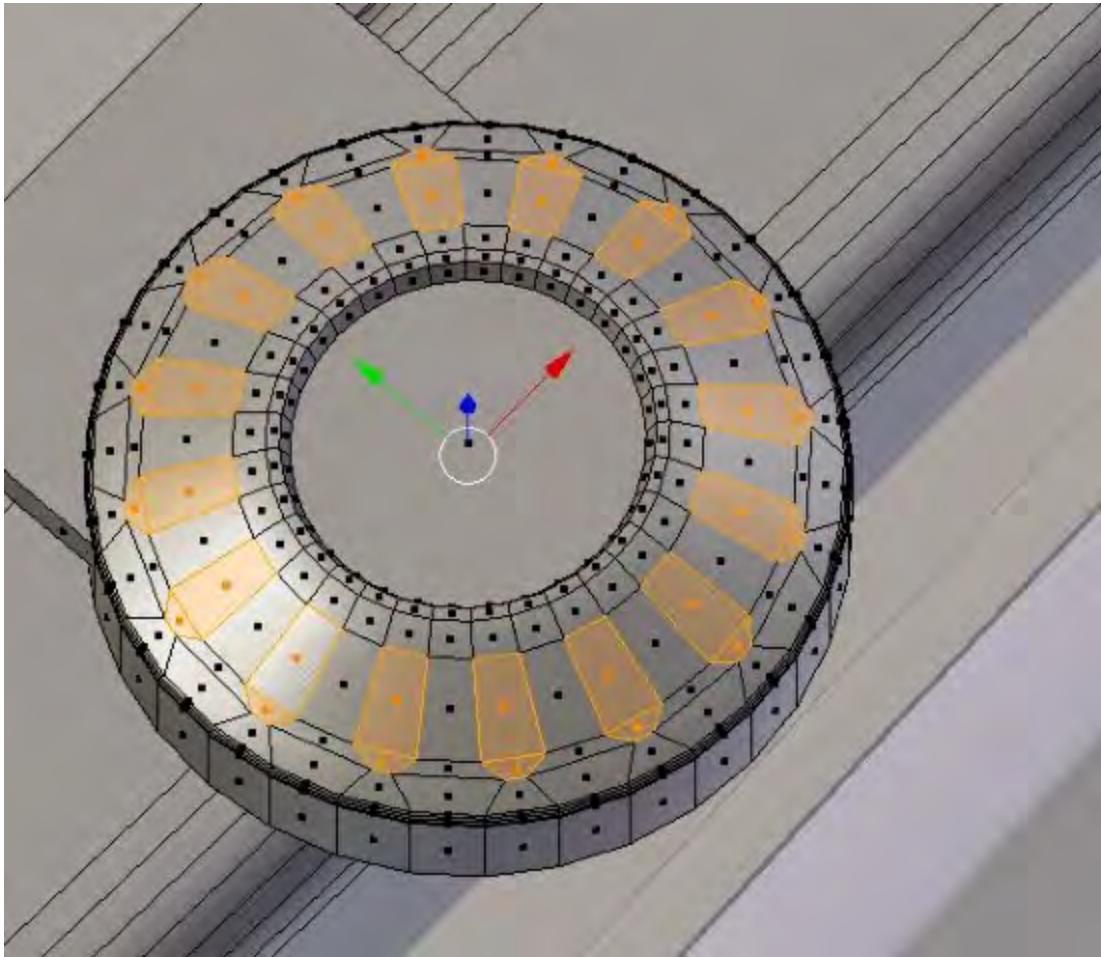
<pagebreak></pagebreak>

Then, setting my **Pivot Point** to **Individual Origins**, I'll select a few of the edges:

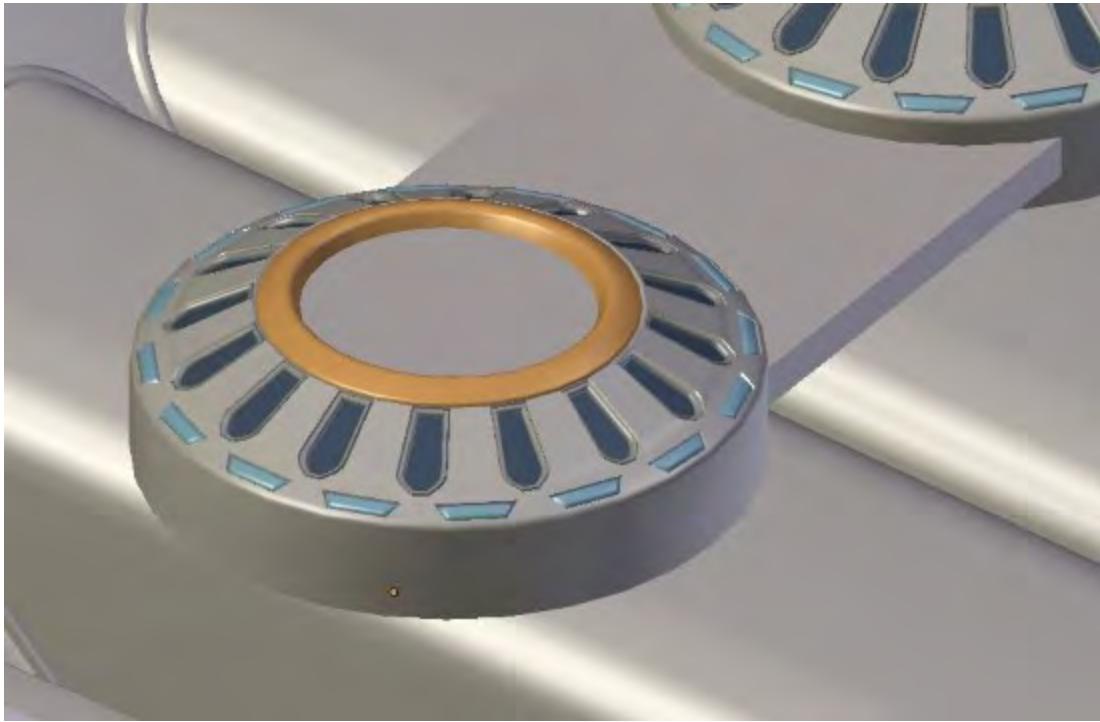


<pagebreak></pagebreak>

I can then shrink those edges individually and extrude the resulting polygons to create some nice shapes:



I'll add some materials to it next:

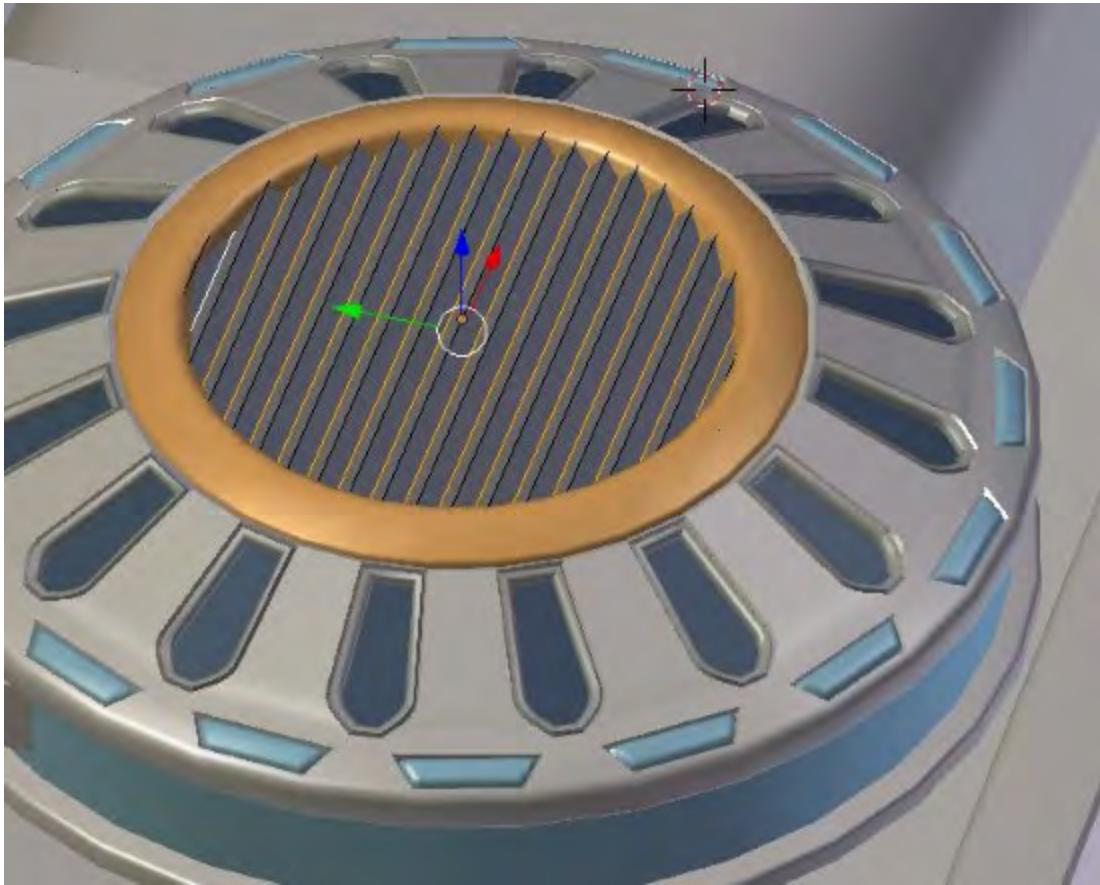


<pagebreak></pagebreak>

I'd like to put a little detail on the top of it, maybe something that looks like a vent. To do this, I'll just add a subdivided plane. Then, grabbing every other edge, you can pull them down a bit (you can also use the **Checker Deselect** tool from the **Select** menu to do this):

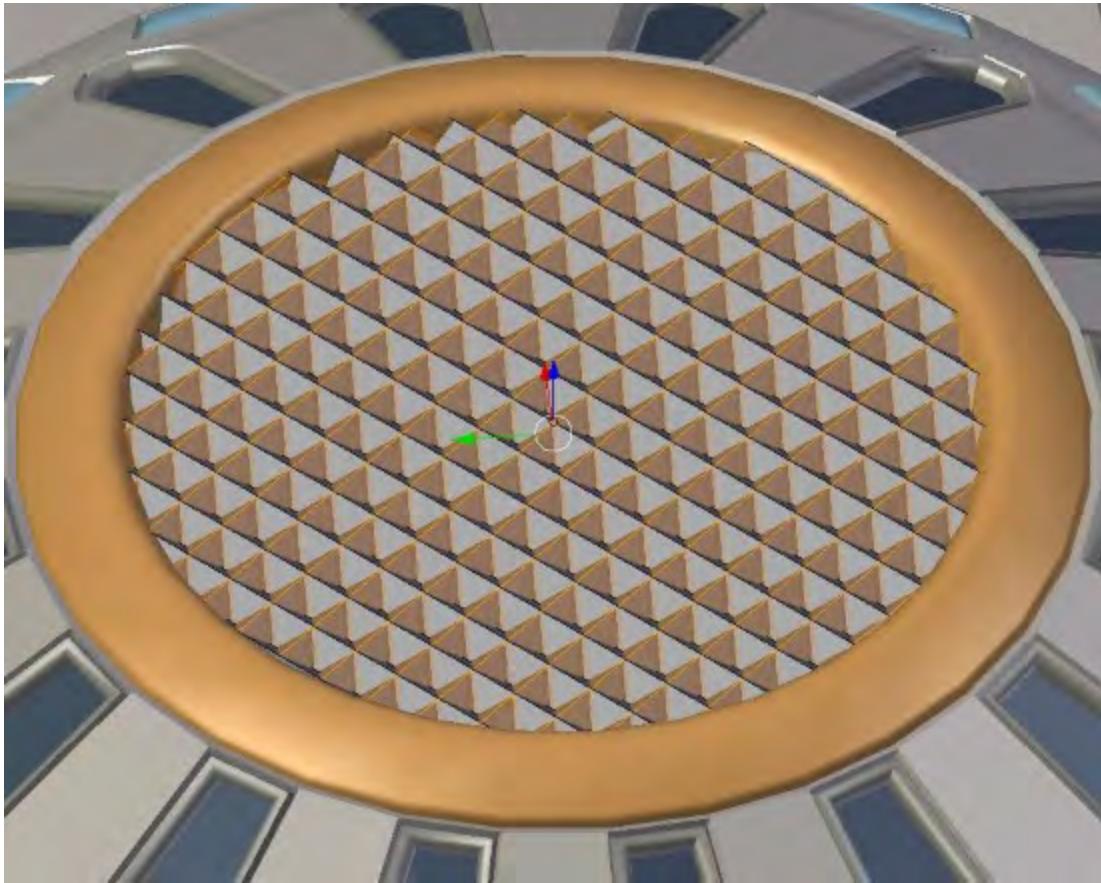
Note

You'll need to pull the corner vertices back so that they don't stick out past the circle.

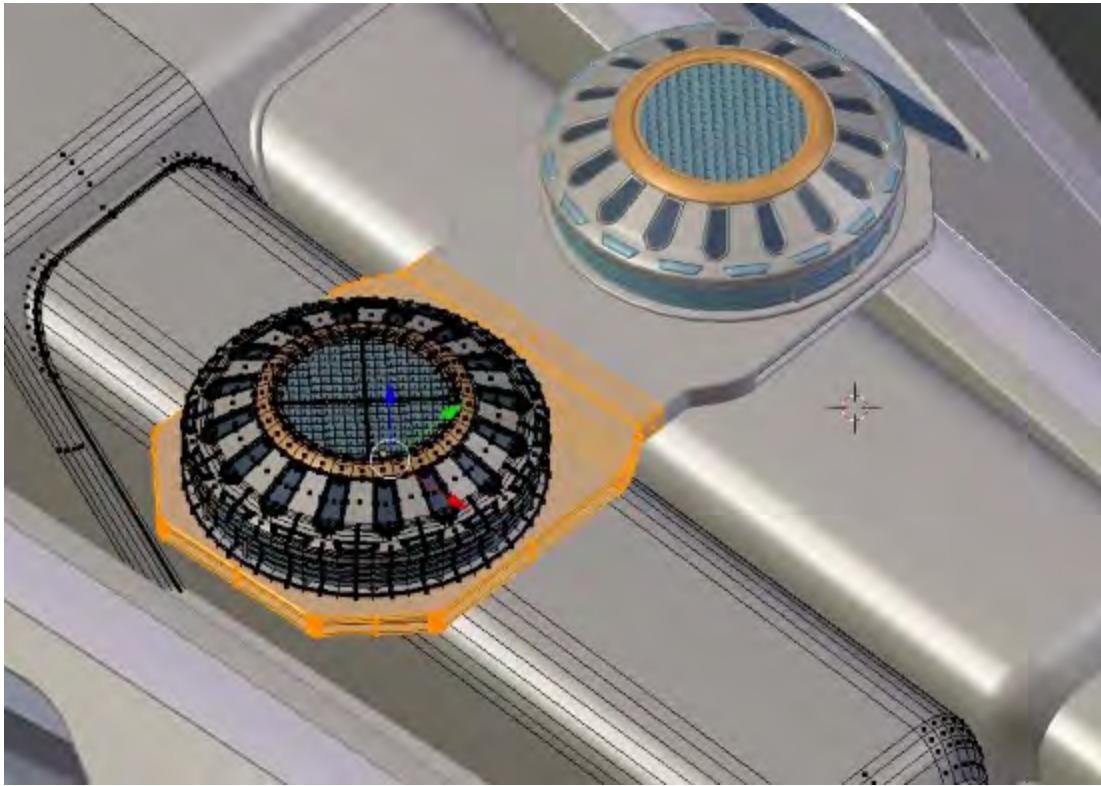


<pagebreak></pagebreak>

This is a quick way to add something that looks like a vent. If you want, you can even duplicate your vent and rotate it by 90 degrees:

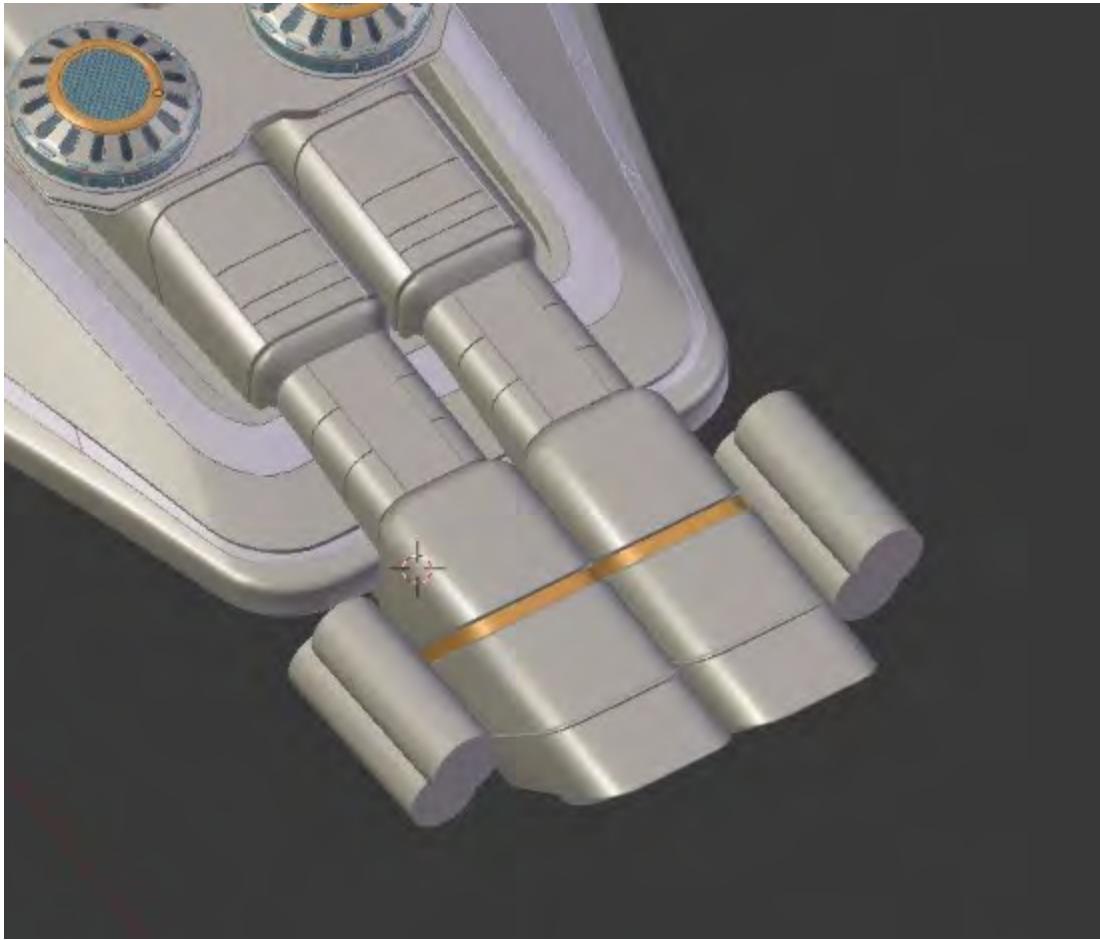


Next, let's carry out some work on the connecting piece:



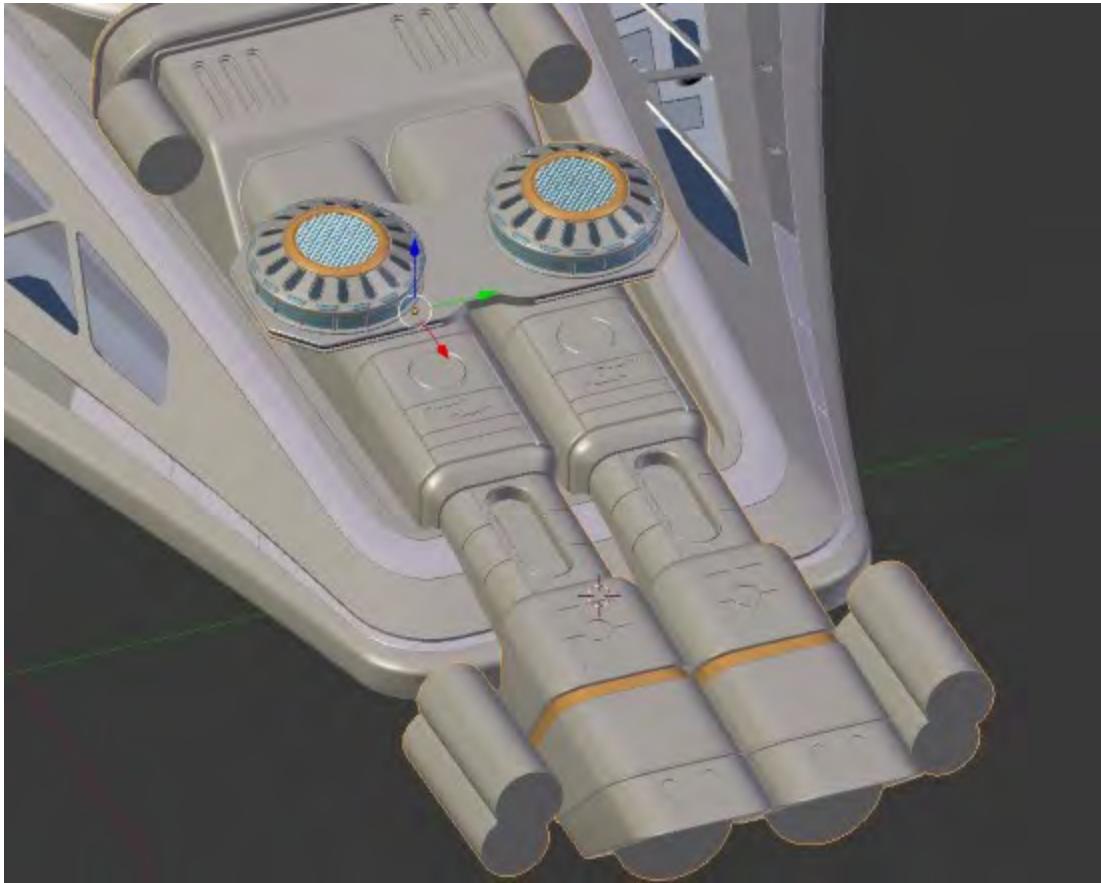
<pagebreak></pagebreak>

Now, we can add some panel lines and cuts to the engine section:



<pagebreak></pagebreak>

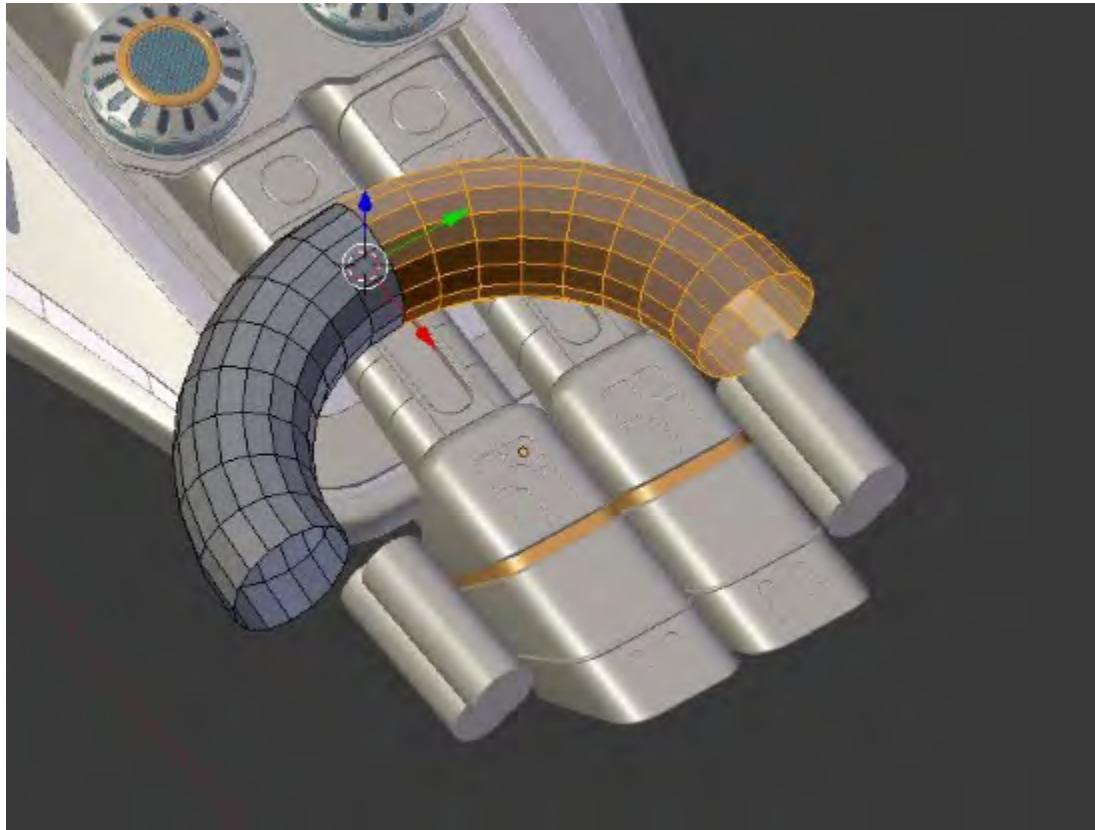
And once again, we'll add cutouts with the **Shrinkwrap** modifier:



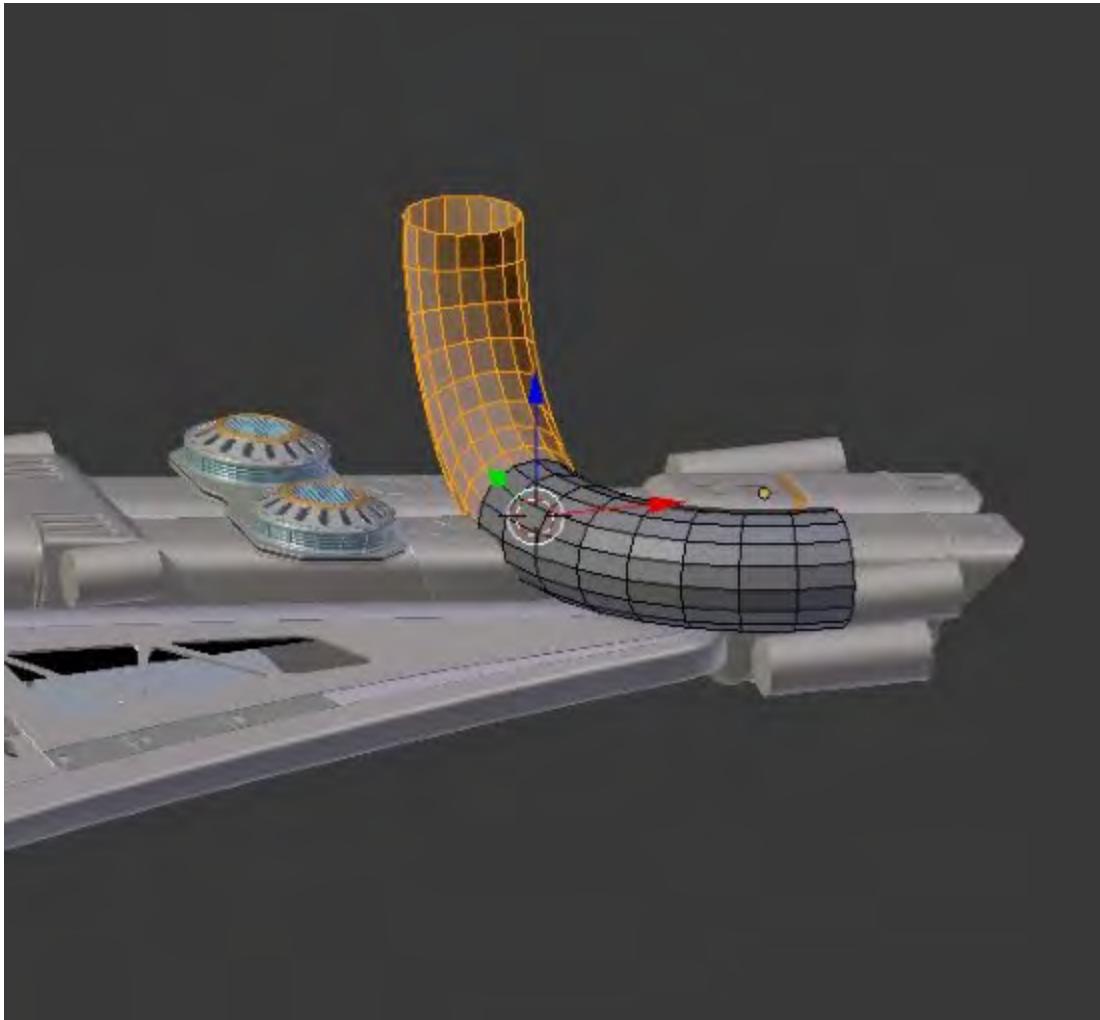
<pagebreak></pagebreak>

Adding detail with pipes

Now, we'll add some pipe sections to the engine. We'll start with **Torus object** for this, the same way we did with the gun model. Just like before, we can cut sections of the pipe. In this case, I've cut the front part in half, then placed the **3D Cursor** in the middle:

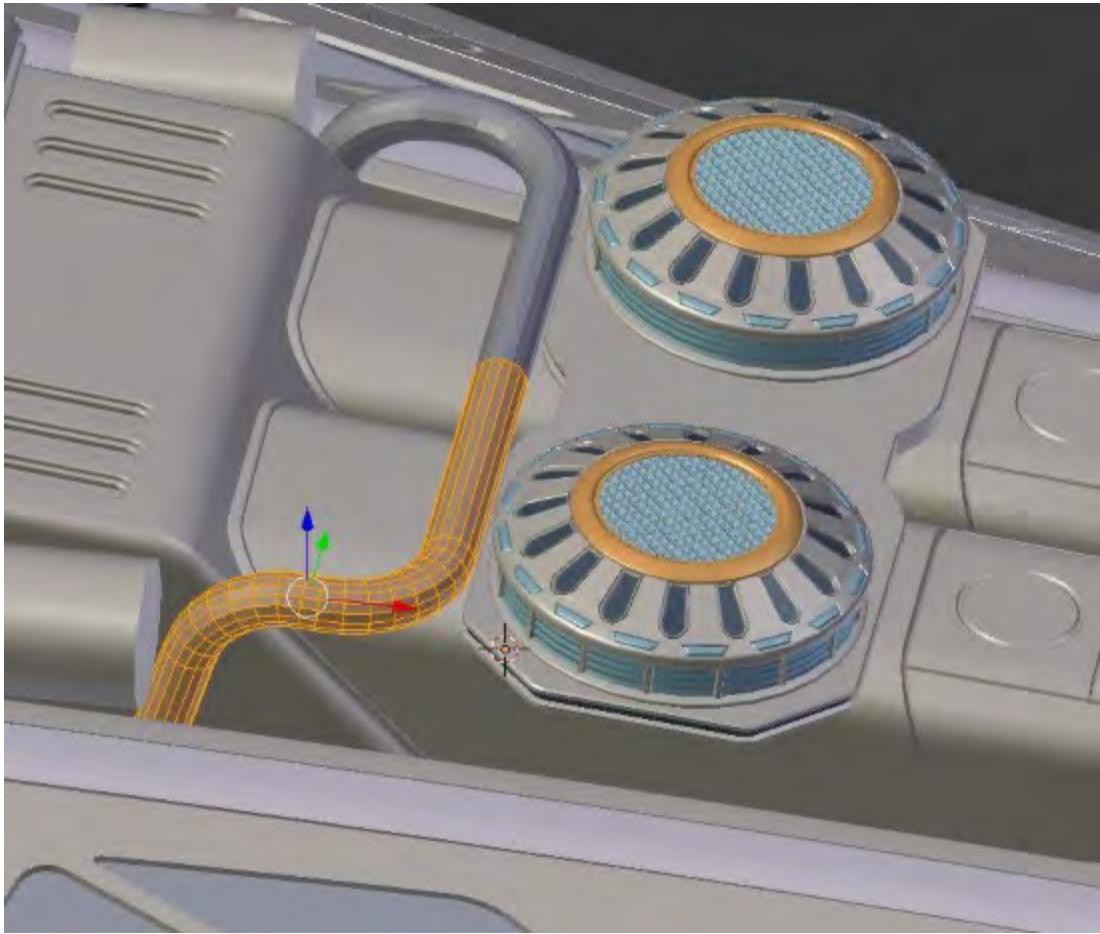


Then, I can rotate one side by 90 degrees around the **3D Cursor**, giving me a nice bend to my pipe:

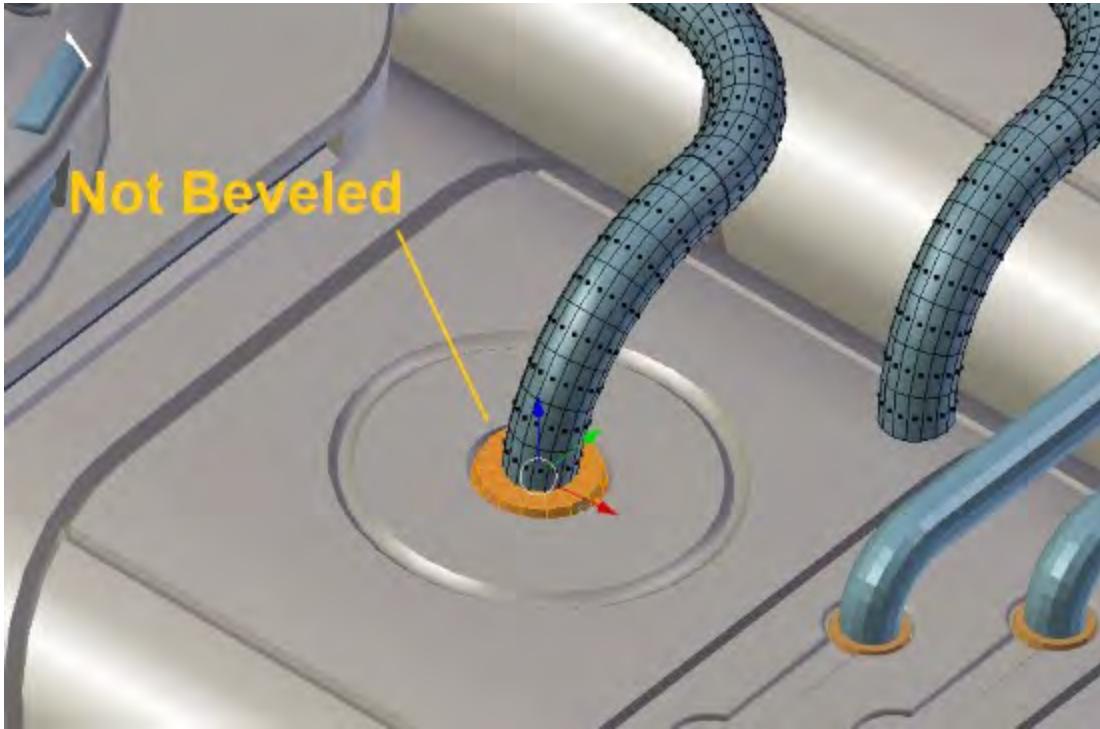


<pagebreak></pagebreak>

I'll first create one larger pipe section across the back of the engines, sort of like a roll bar (though of course, a spaceship wouldn't really need a roll bar):

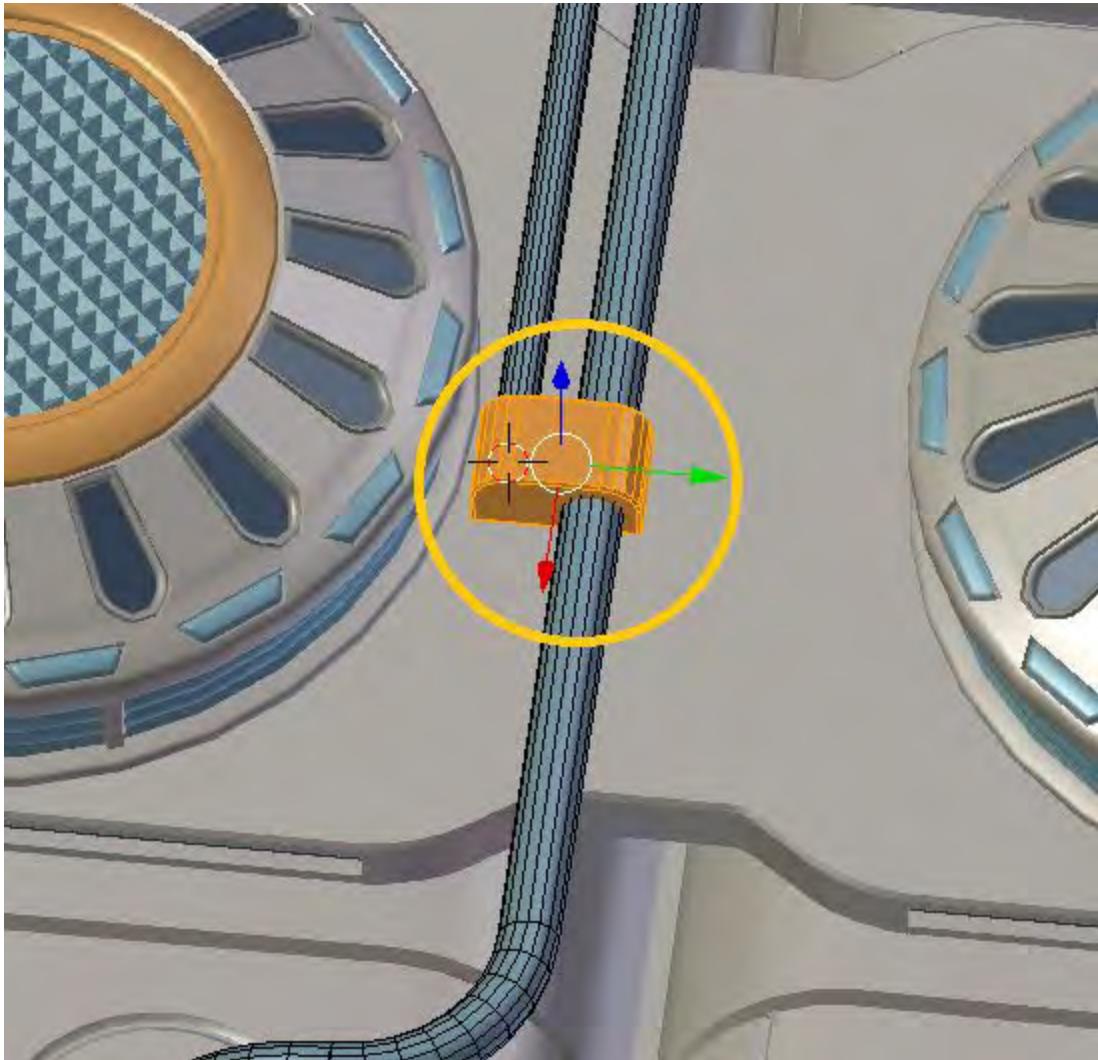


You can then create a variety of pipes going into whatever places you feel are appropriate. I'm going to add little copper connections to the ends of mine. Since they're so small, I'm not going to worry about beveling:

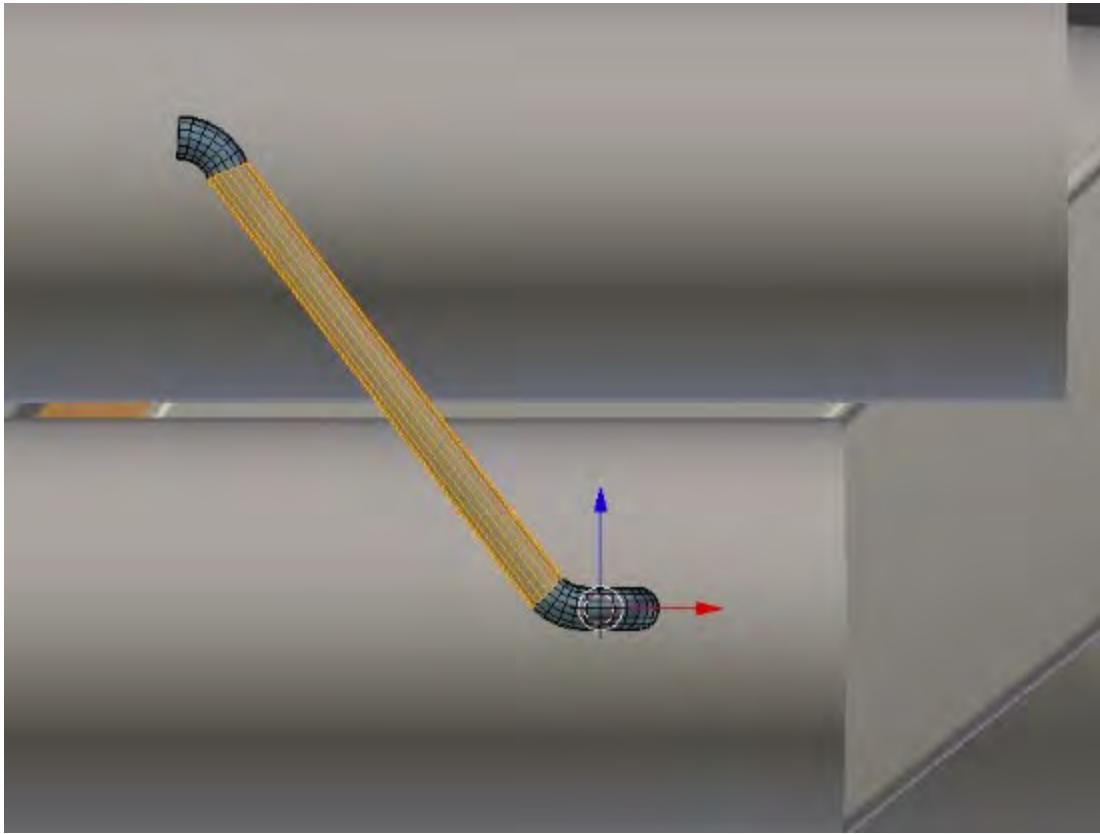


<pagebreak></pagebreak>

As we do this, you can also add small boxes or circles to connect various parts together:

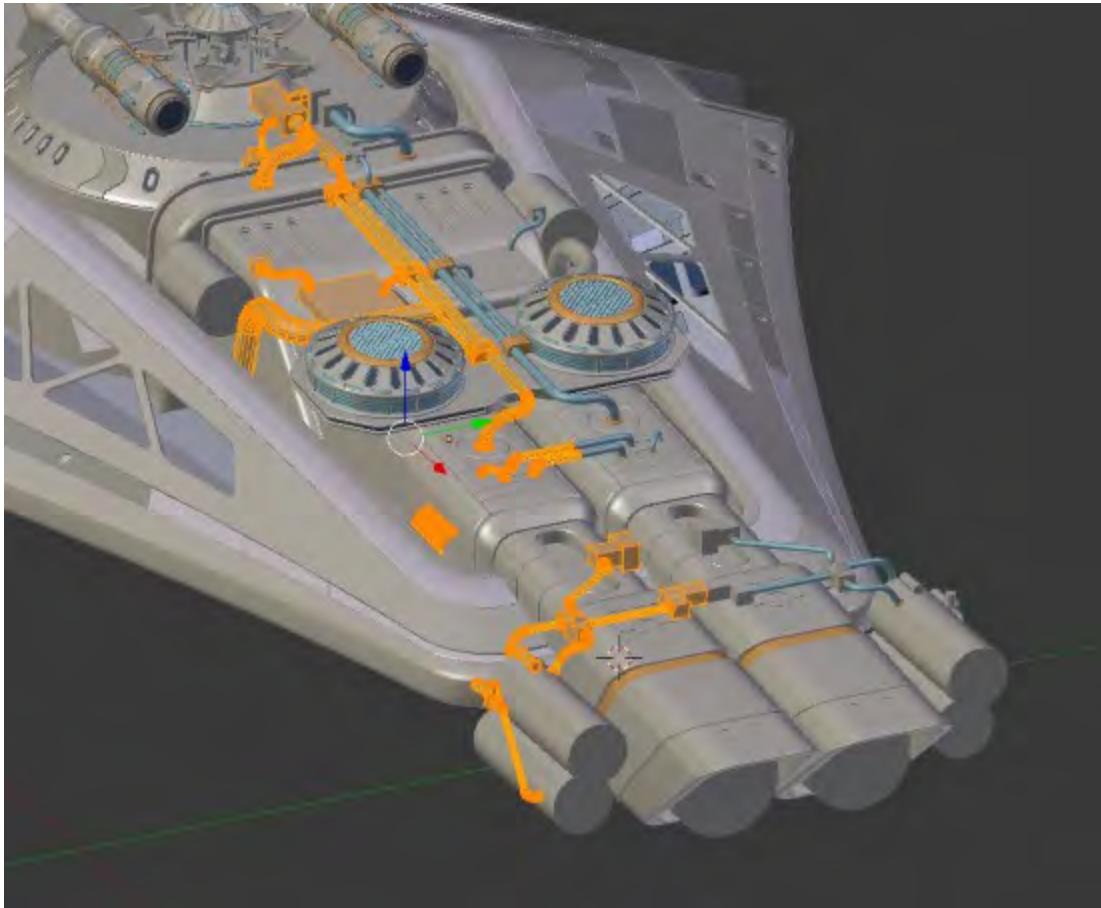


You don't always need to use right angles, either:



<pagebreak></pagebreak>

In the end, here's what I ended up doing with the pipes:

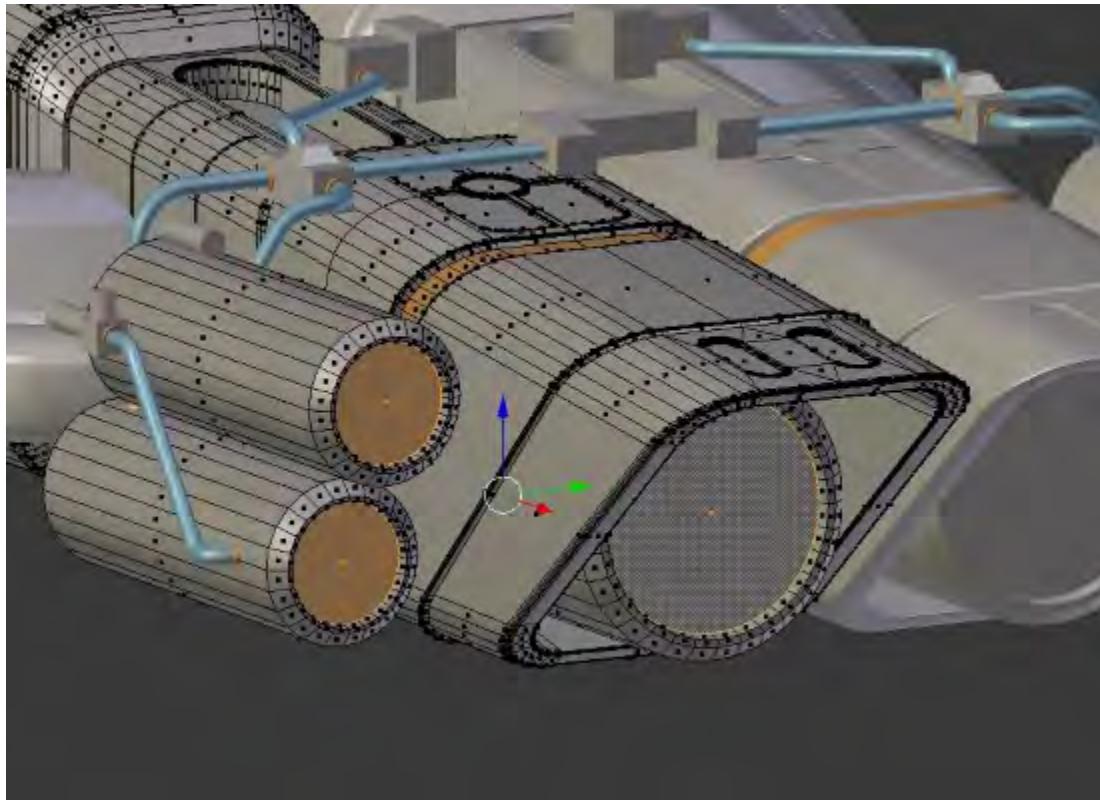


<pagebreak></pagebreak>

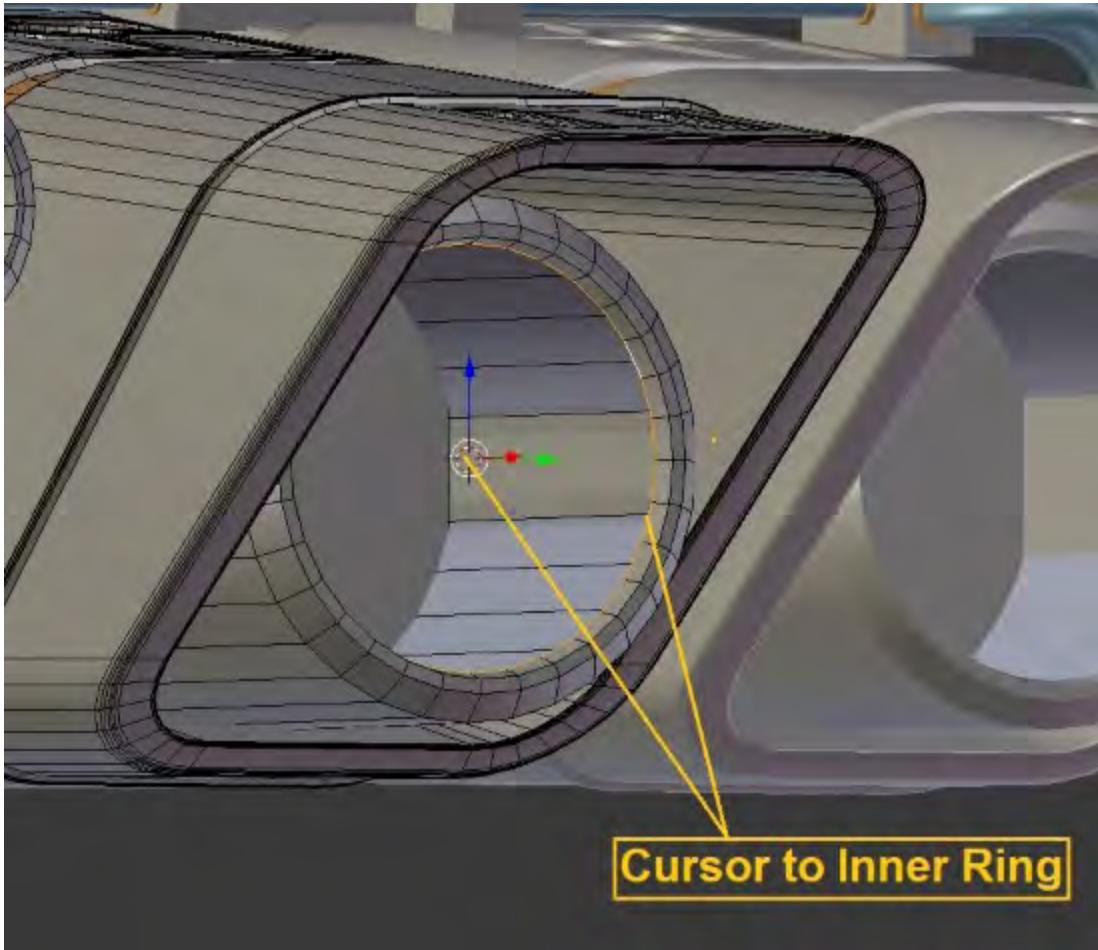
Finishing our main objects

Now that we've got our pipes in place, we'll continue working on the main parts of our model.

First, we'll want to work on the exhaust ports at the back of the engine section. We'll start by just beveling things down a little:



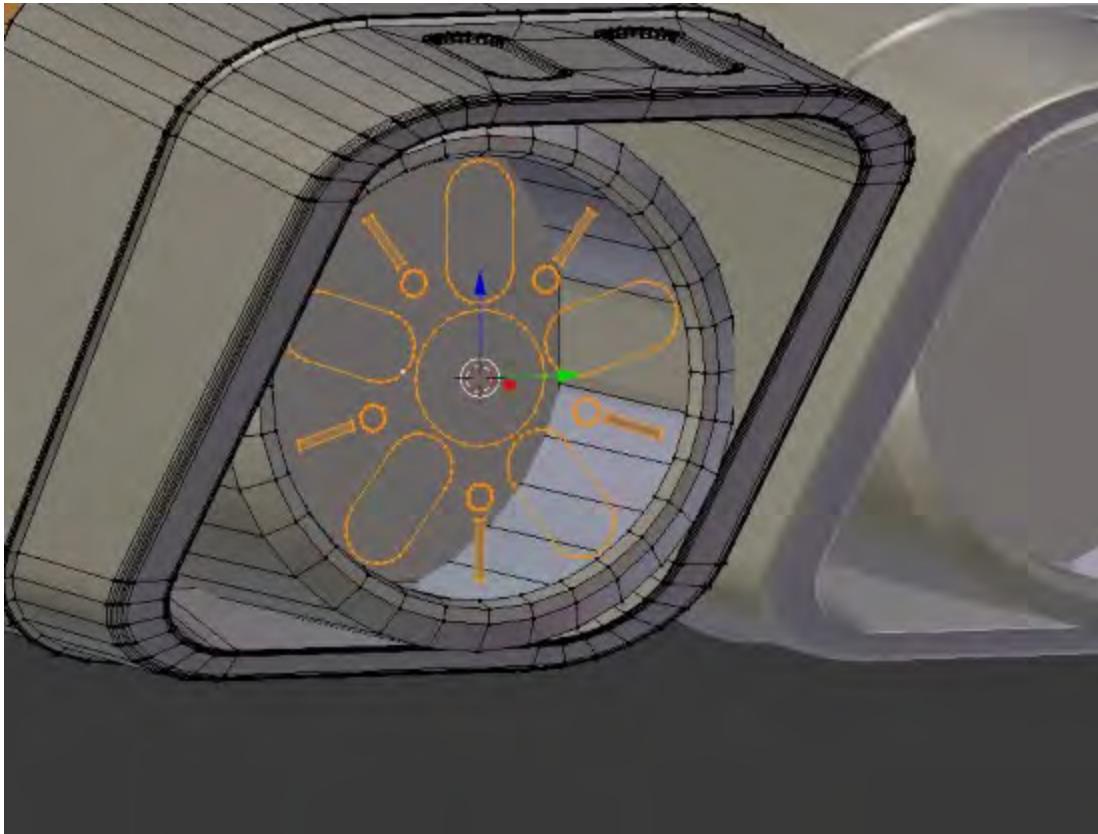
Then, we can delete those end caps and place the cursor at the center of one of the engine openings:



Cursor to Inner Ring

<pagebreak></pagebreak>

At this point, you can add shapes to cut out without using a **Shrinkwrap** modifier—the shapes are already lined up:

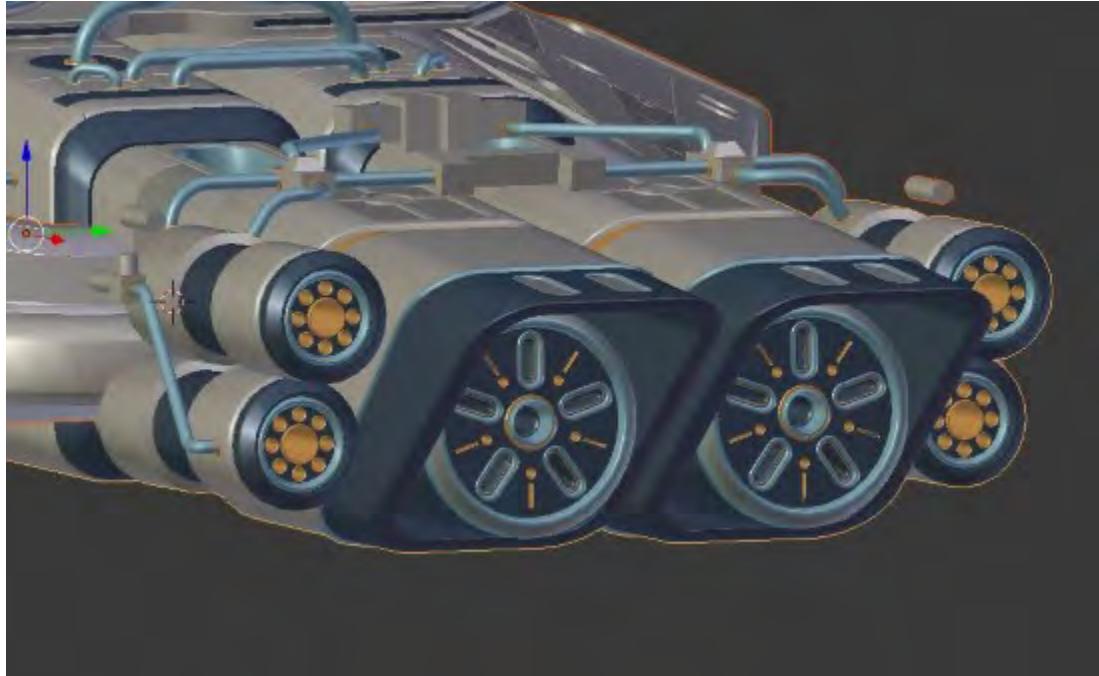


When you're done, you'll need to fill in those faces. Then, we'll extrude and bevel:



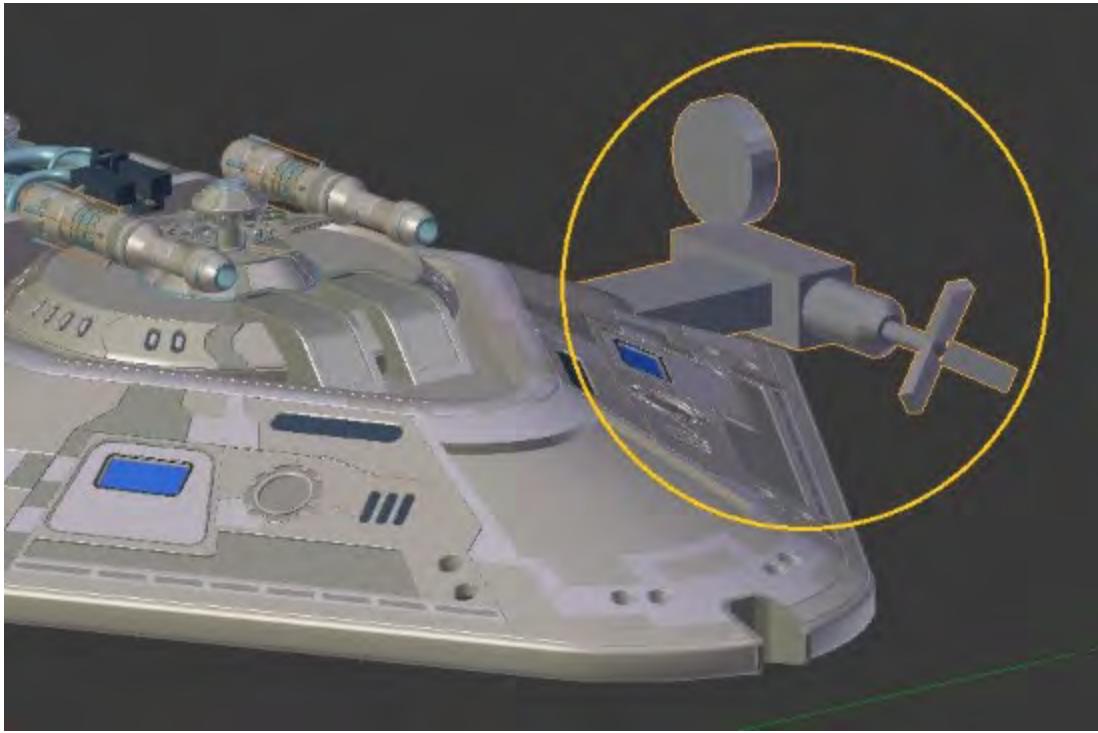
<pagebreak></pagebreak>

Don't forget to add materials as well:



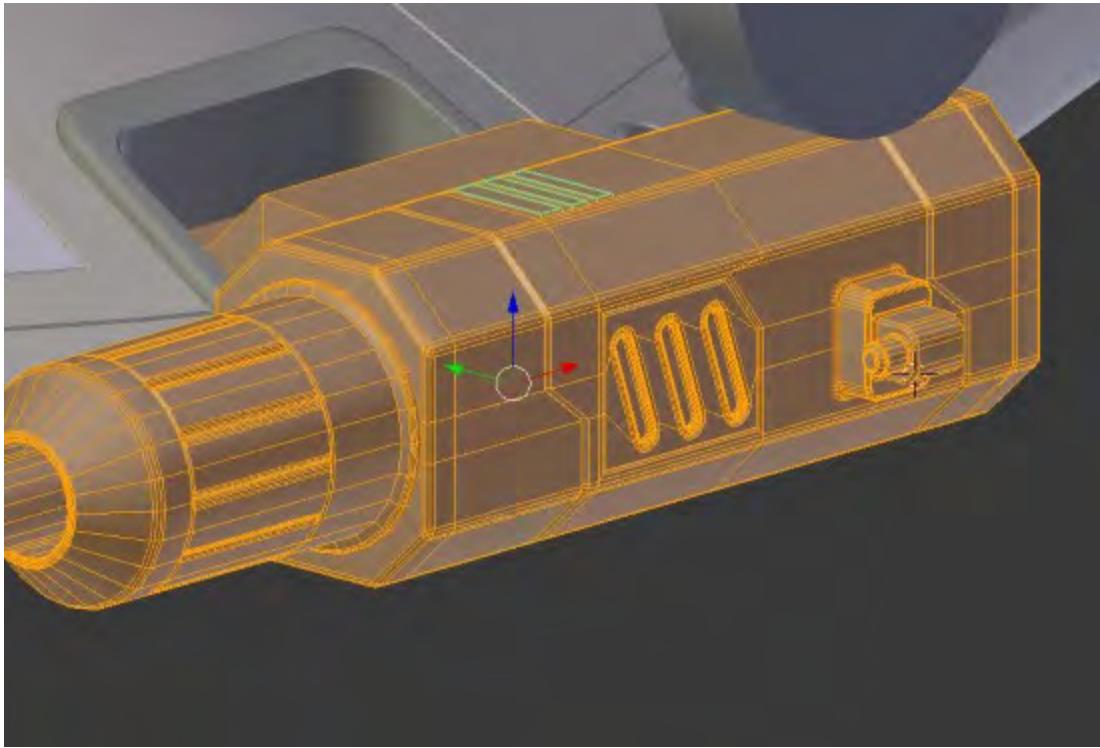
I think that's a good stopping point for now, as far as the engines go.

Next, we'll bring in the grappling gun:

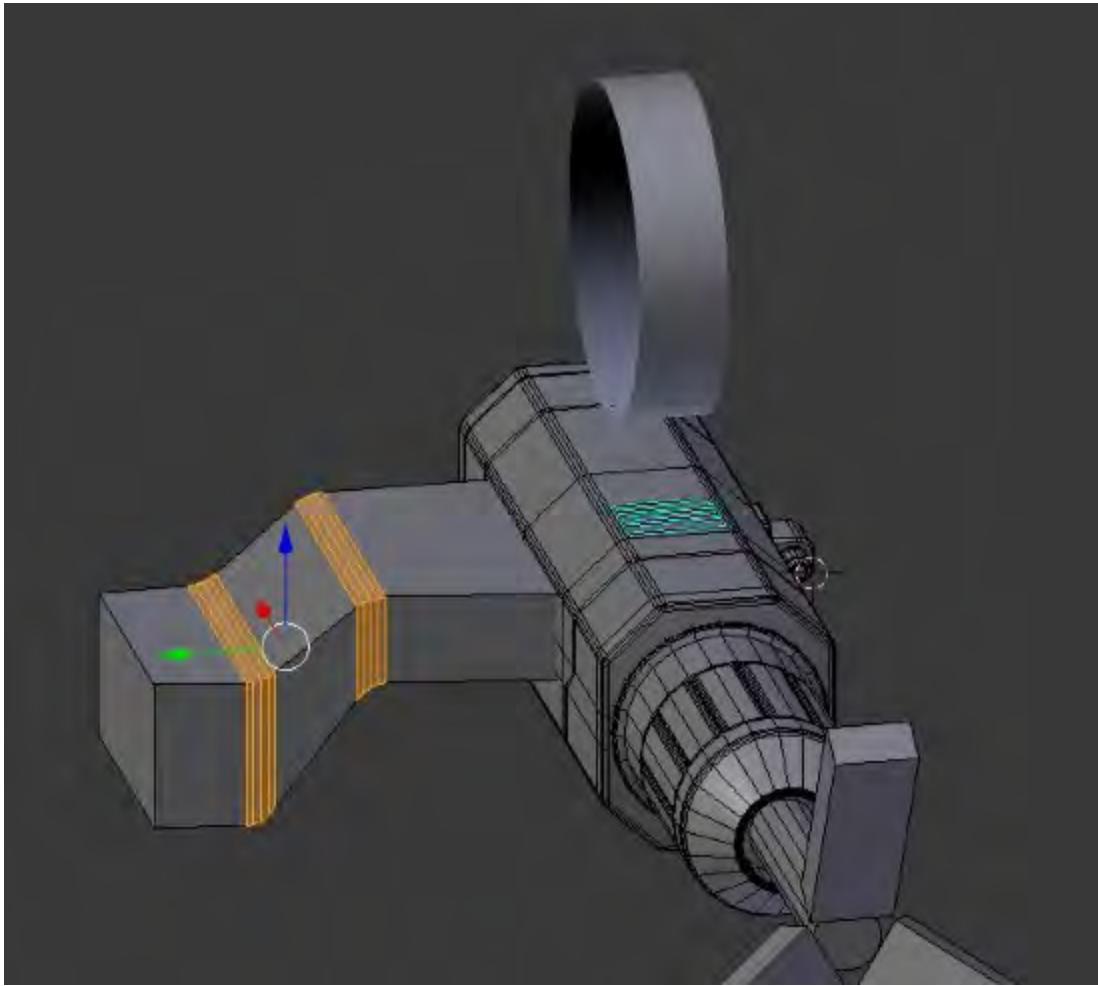


<pagebreak></pagebreak>

Using the same methods we used on the other sections, I'm just going to add a bit of detail to the main section of the grappling gun:

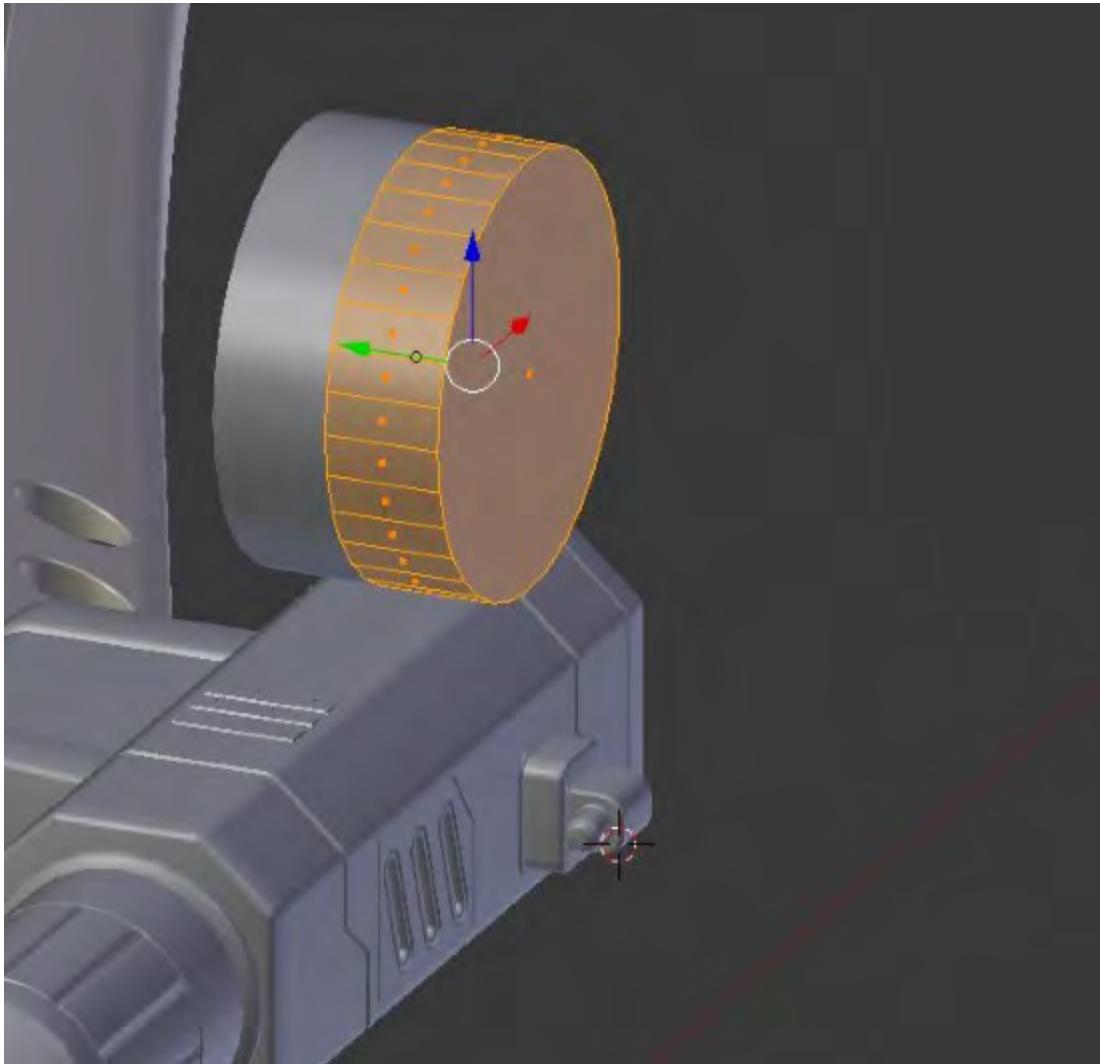


I'll also bevel the arm that connects it to the body and separate the cable reel into another object:

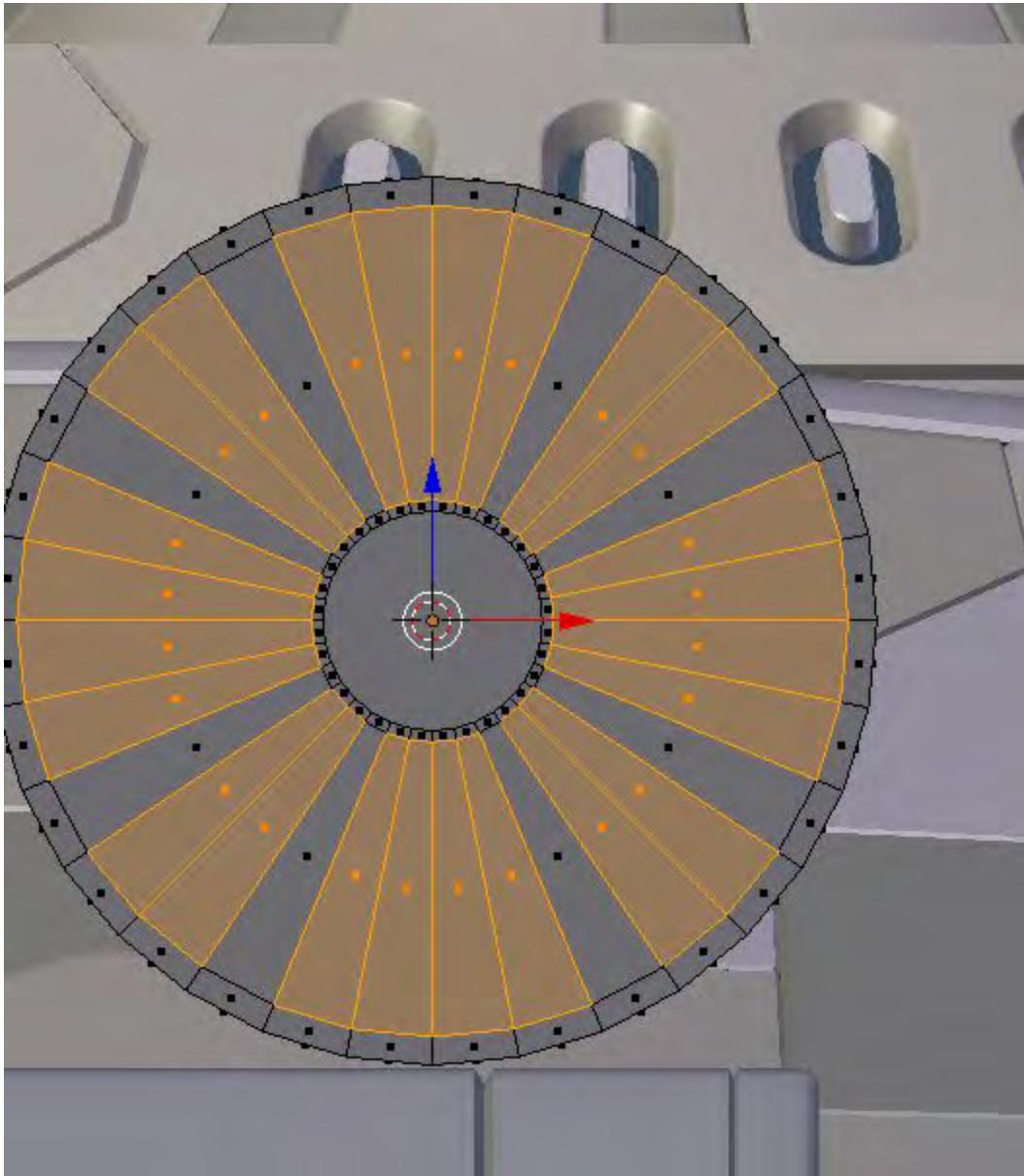


<pagebreak></pagebreak>

Next, we'll add a mirror modifier to it:

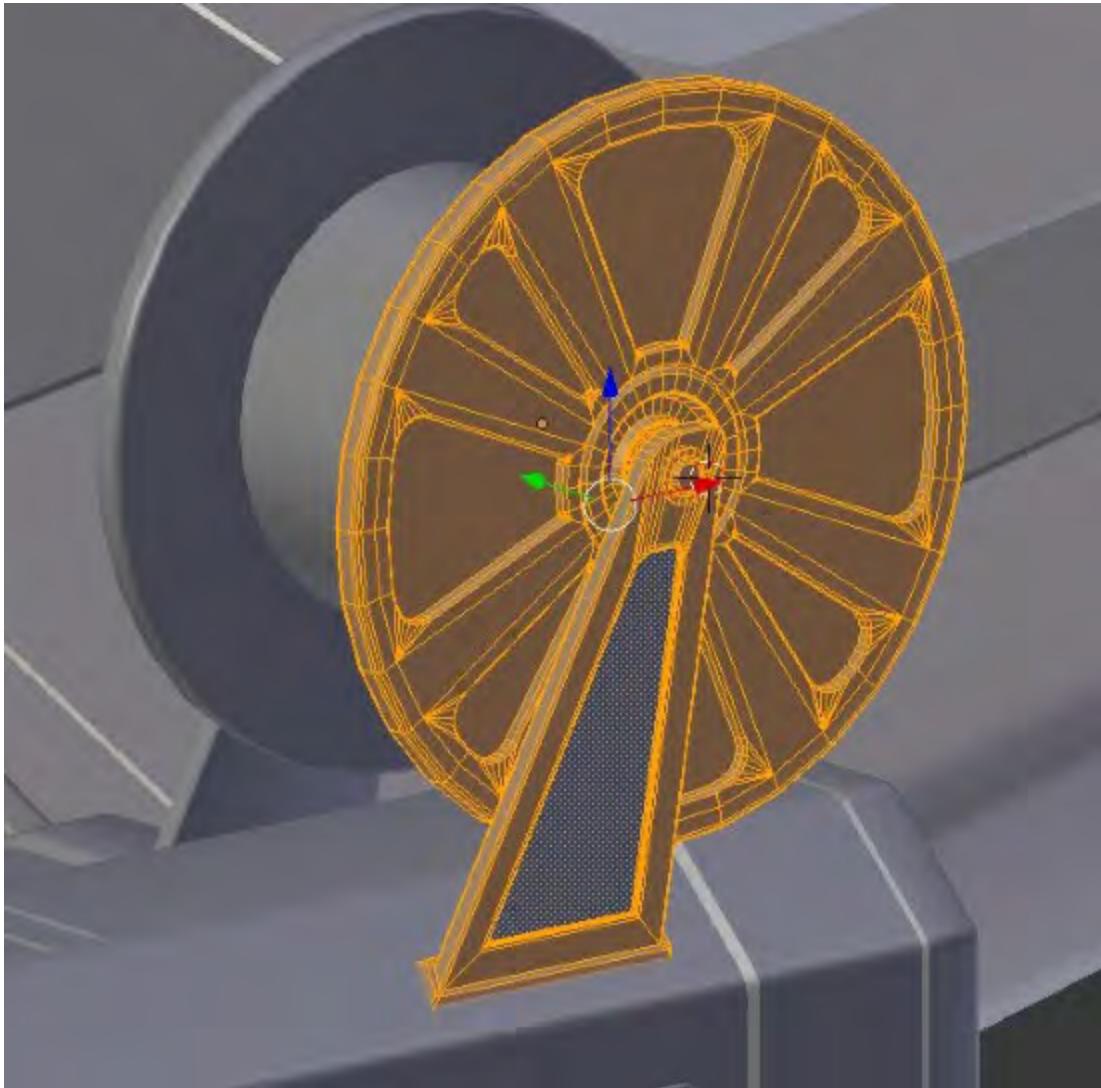


I'll pick a few faces here to create cutouts:

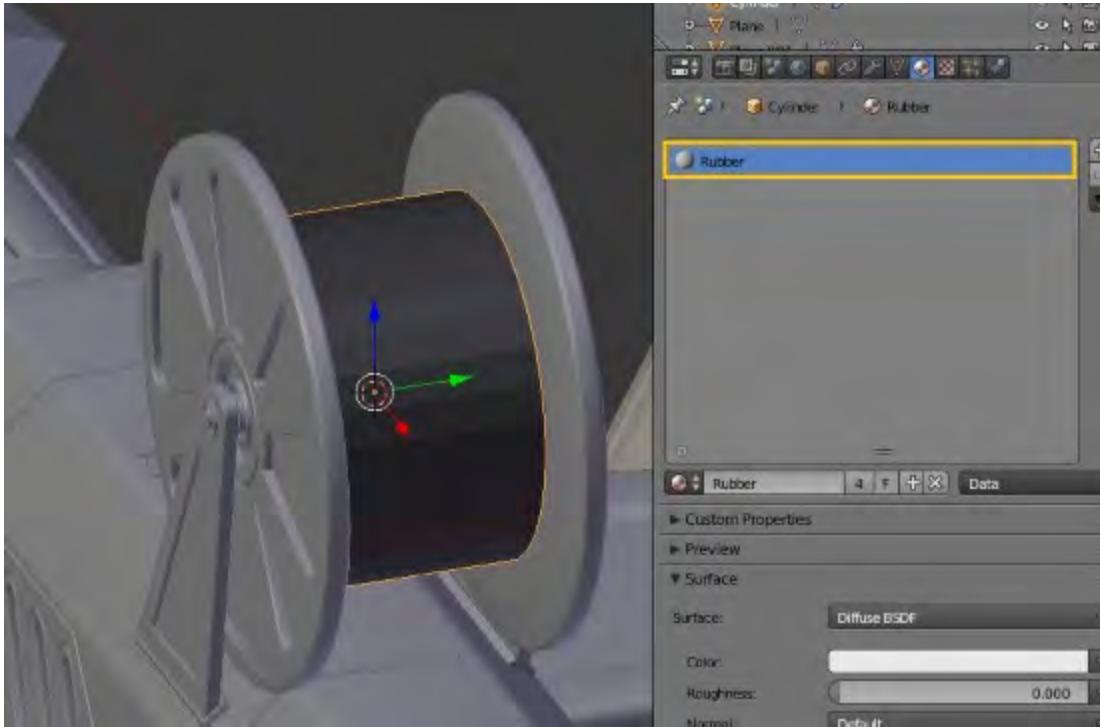


<pagebreak></pagebreak>

Then, we can round out those corners and add struts to connect it to the main section of the grappling gun:

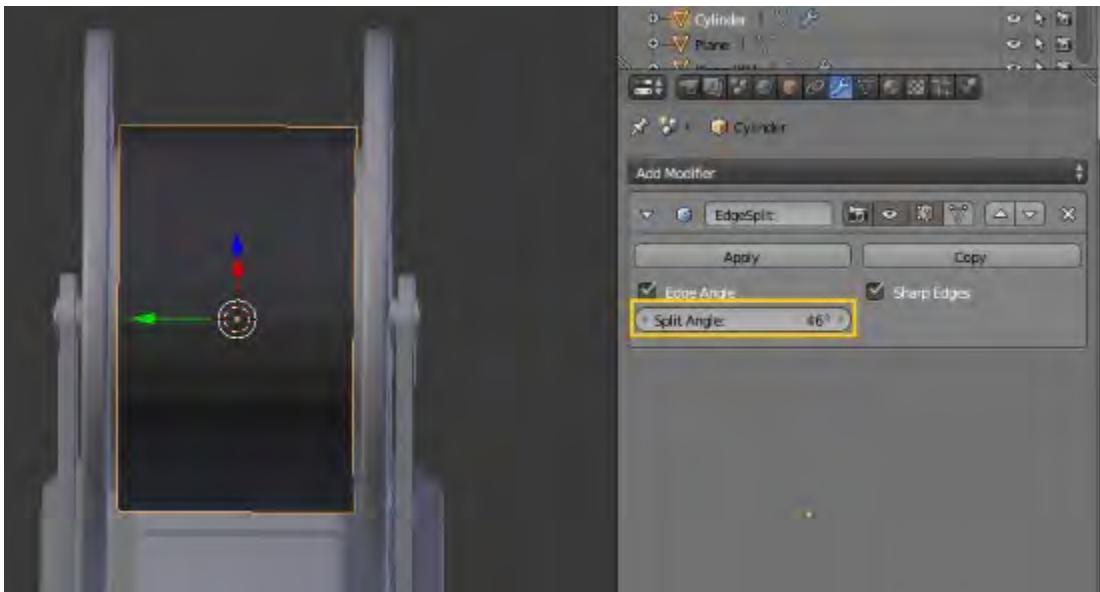


Next, I'm going to add a new cylinder and assign the **Rubber** material to it—this will be our spool of cable:

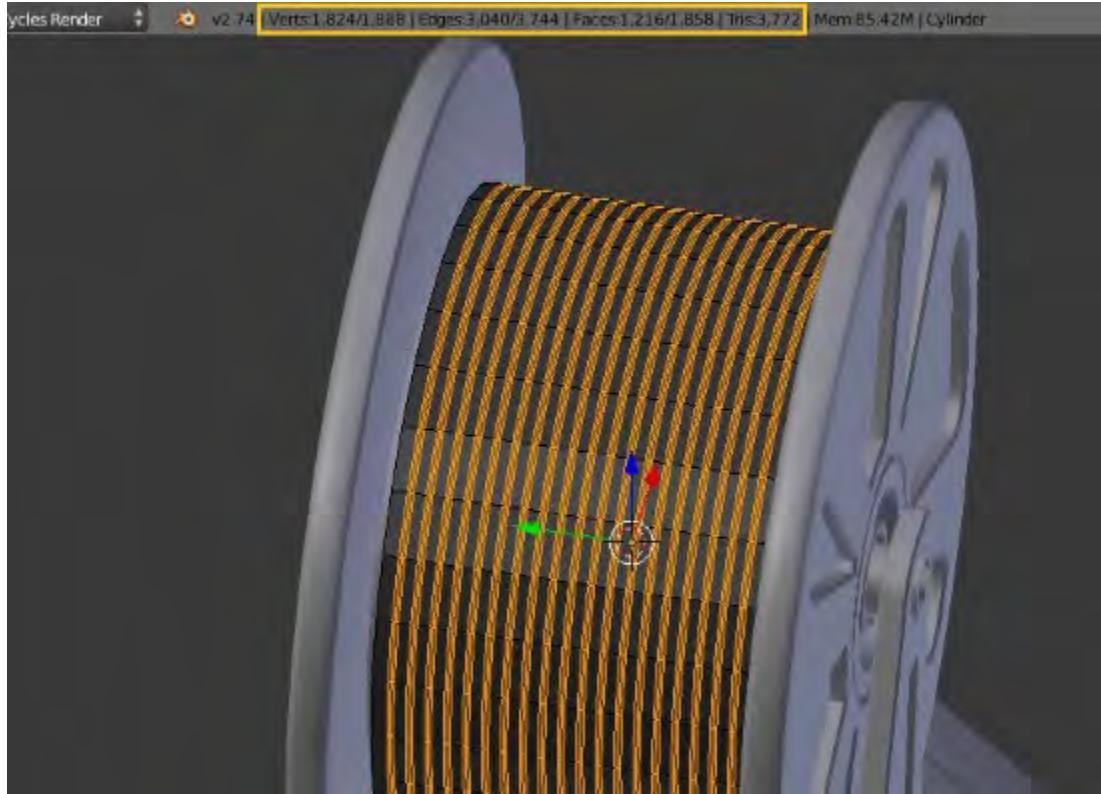


<pagebreak></pagebreak>

I'm going to add an **Edge Split** modifier to my cable here, but we'll set the **Split Angle** to a higher value:

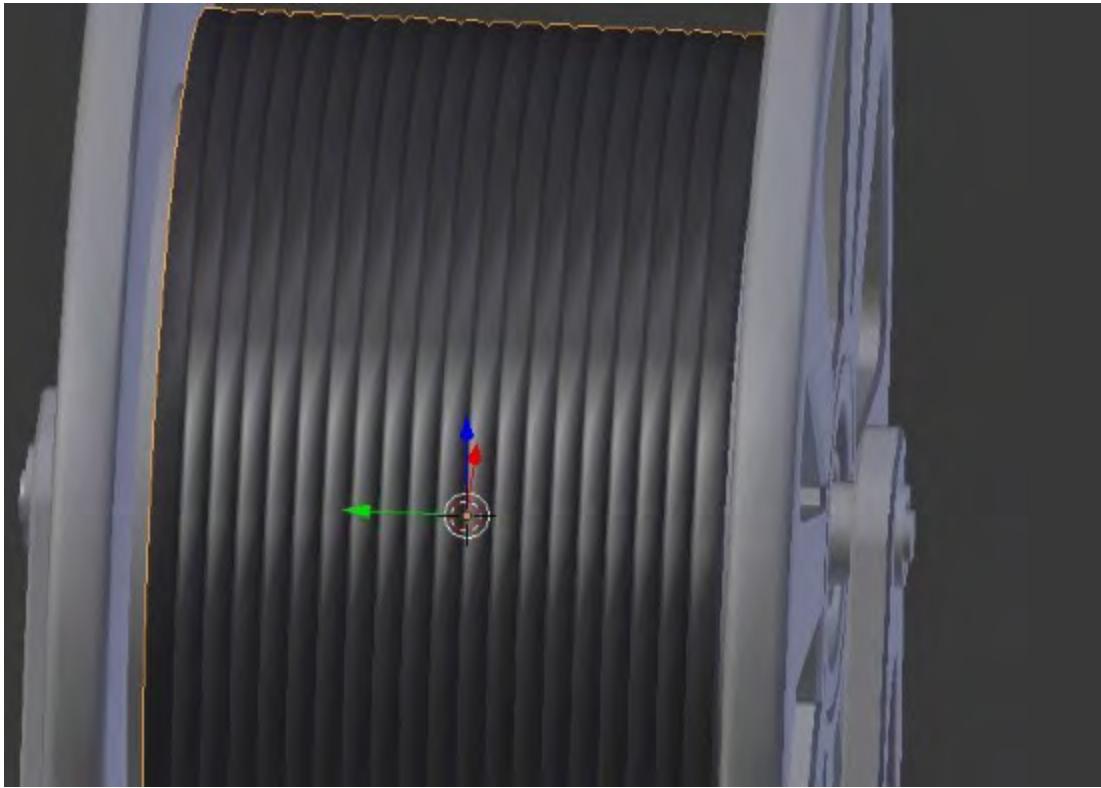


That way, edges that would normally be sharp (more than 30 degrees) will be smooth. The reason for this is just that we're going to add a lot of geometry to the reel anyway (the cable is very thin), and I don't want to have to bevel it a huge number of times:

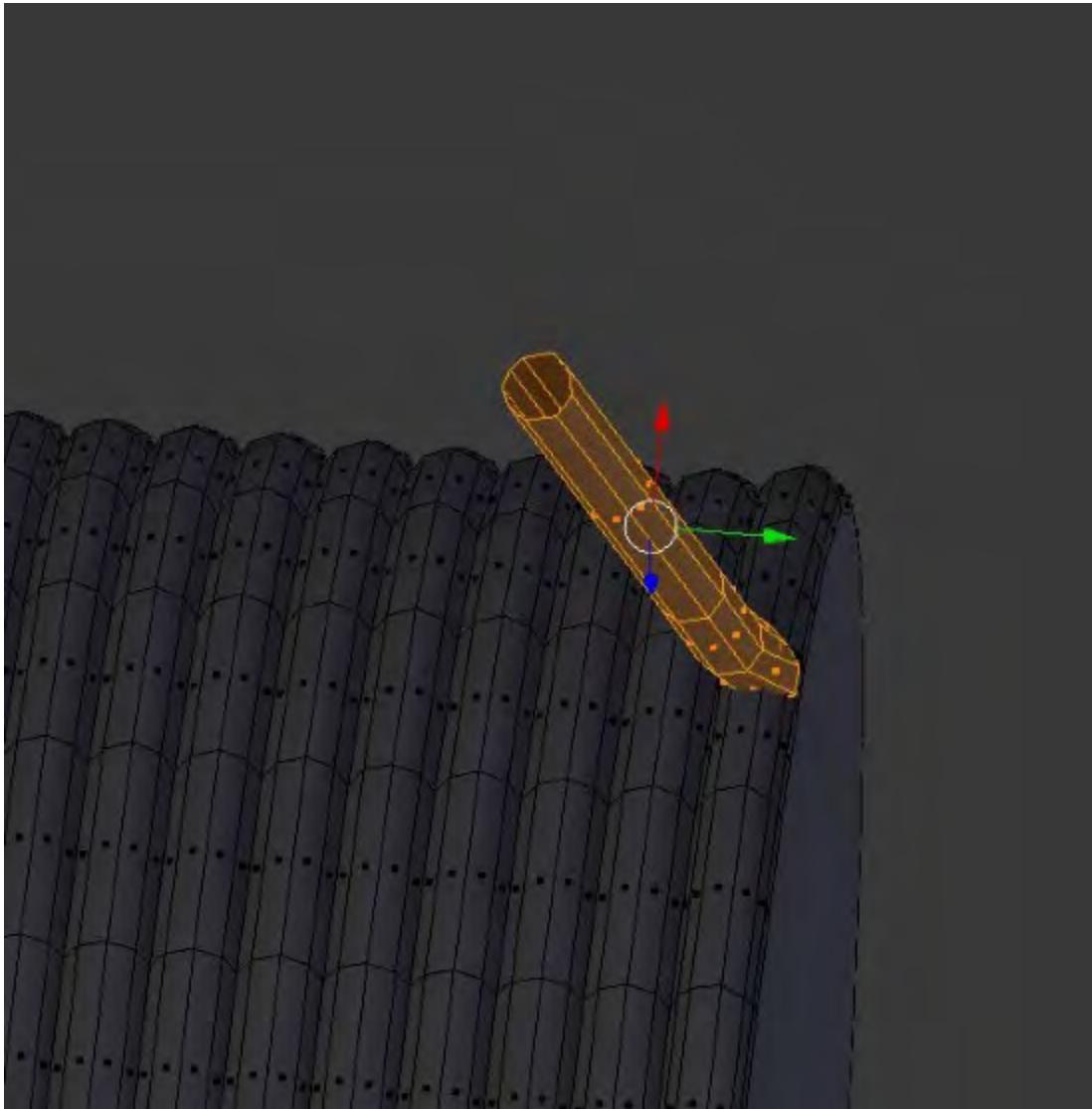


<pagebreak></pagebreak>

This is what my cable ended up looking like:



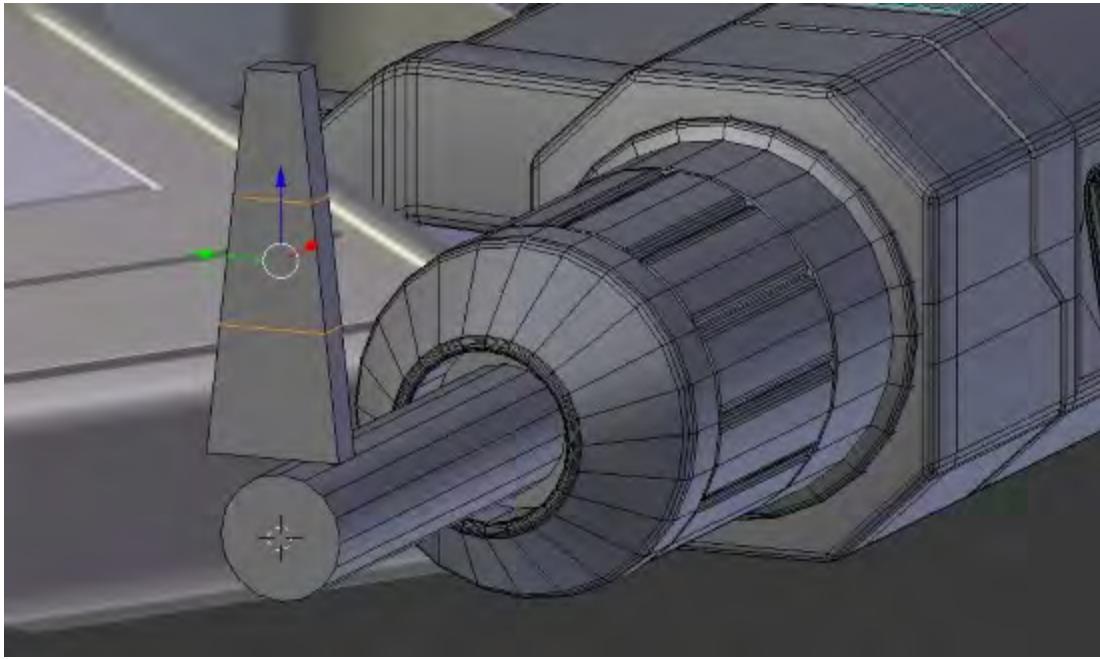
Of course, the problem with creating a cable reel this way is that there's no "end" to the cable. In order to fix that, we'll just come in and delete one small section and extrude it down. Then, we can fill in the back faces to create a small cable:



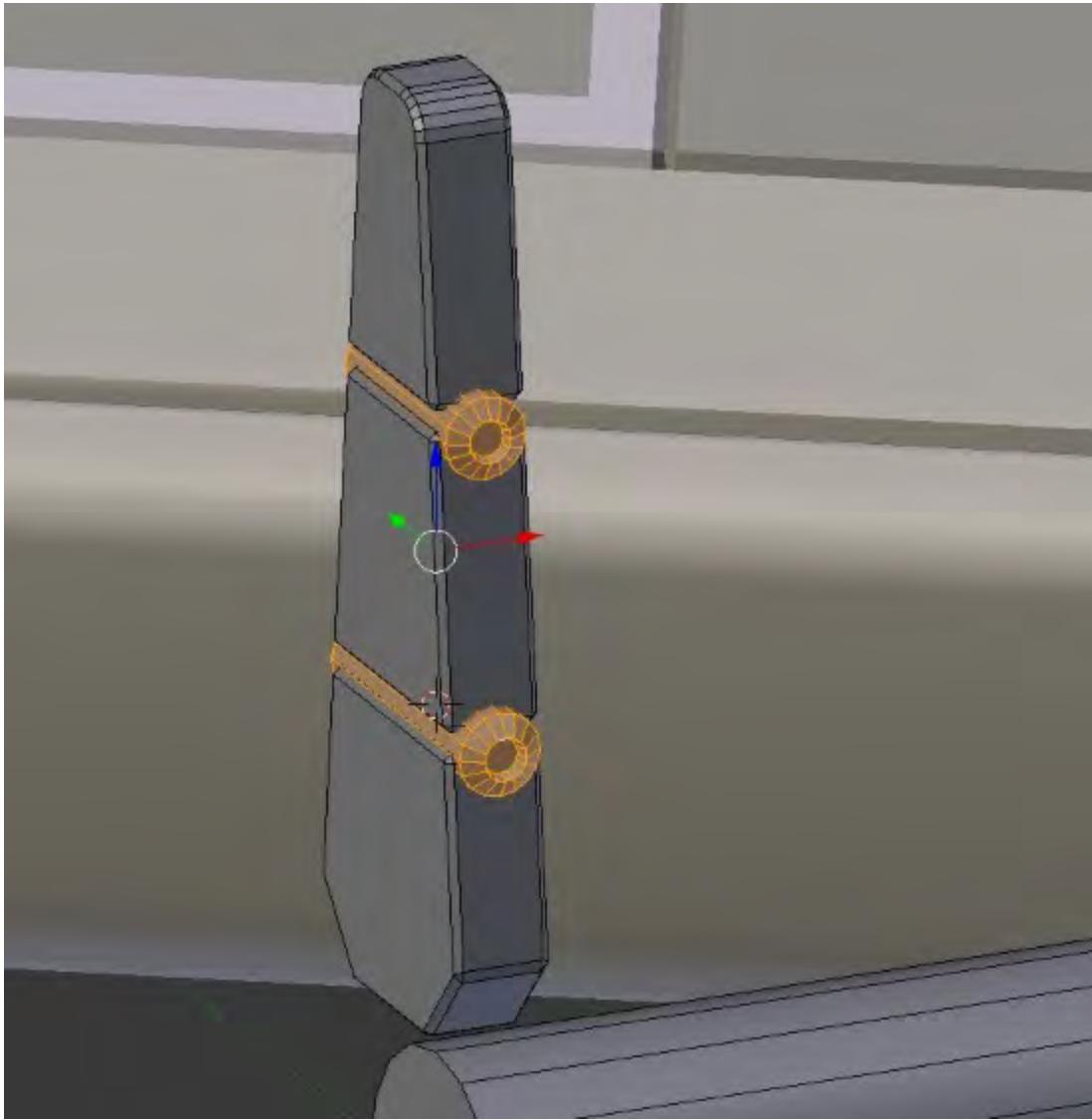
It's not perfect, but it's such a tiny detail that it will look good 99% of the time. Of course, you're always welcome to improve it. One such method might be to use the Screw tool that we'll look at in just a moment.

Now, let's turn our attention to the grappling claw. The original geometry was good for showing us the basic shape, but it's probably easier to just delete those parts now.

Instead, I'll add a scaled cube and just put two loop cuts in:

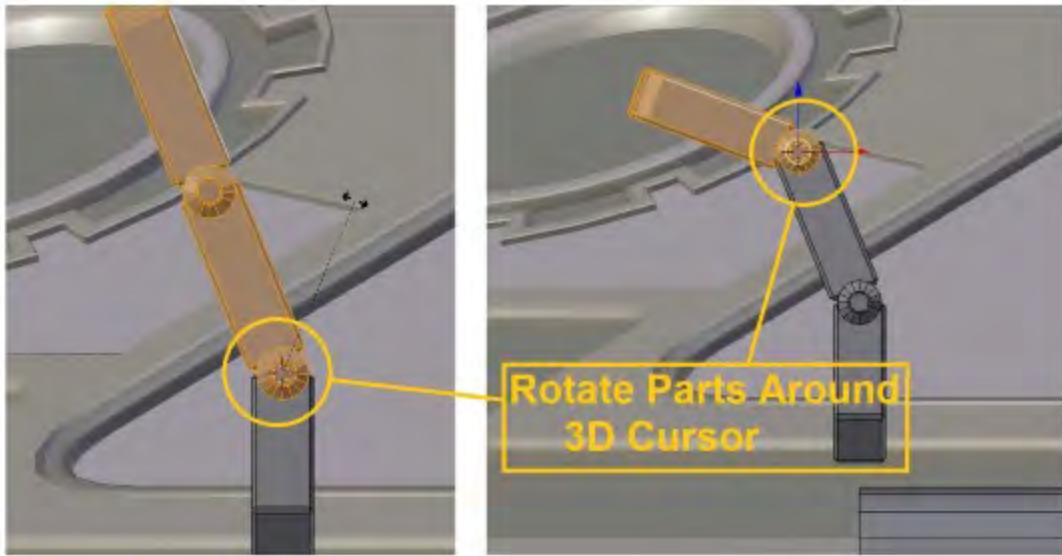


Then, with a little beveling, we can start to get the shape we want. I'll add a few small cylinders to link the segments:



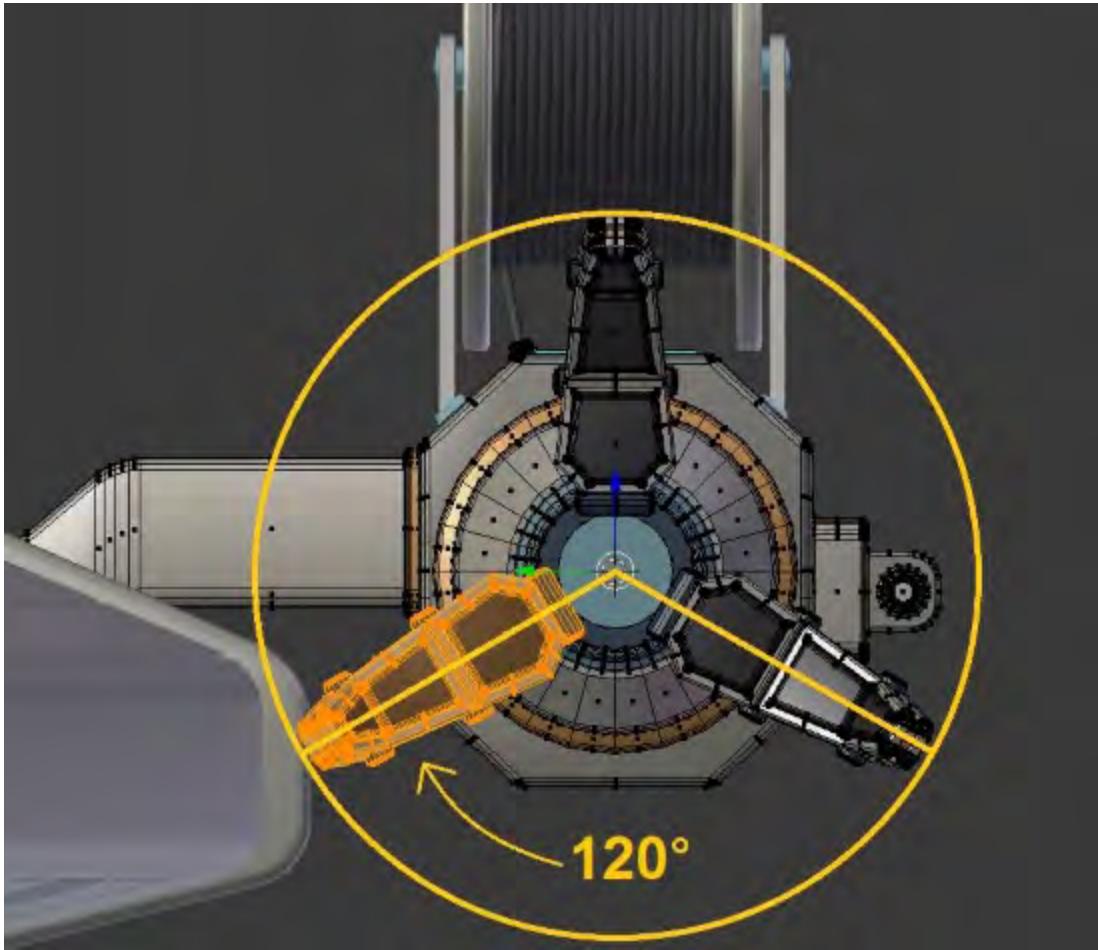
<pagebreak></pagebreak>

We can then set the **3D Cursor** to those smaller cylinders and rotate the parts how we'd like them:



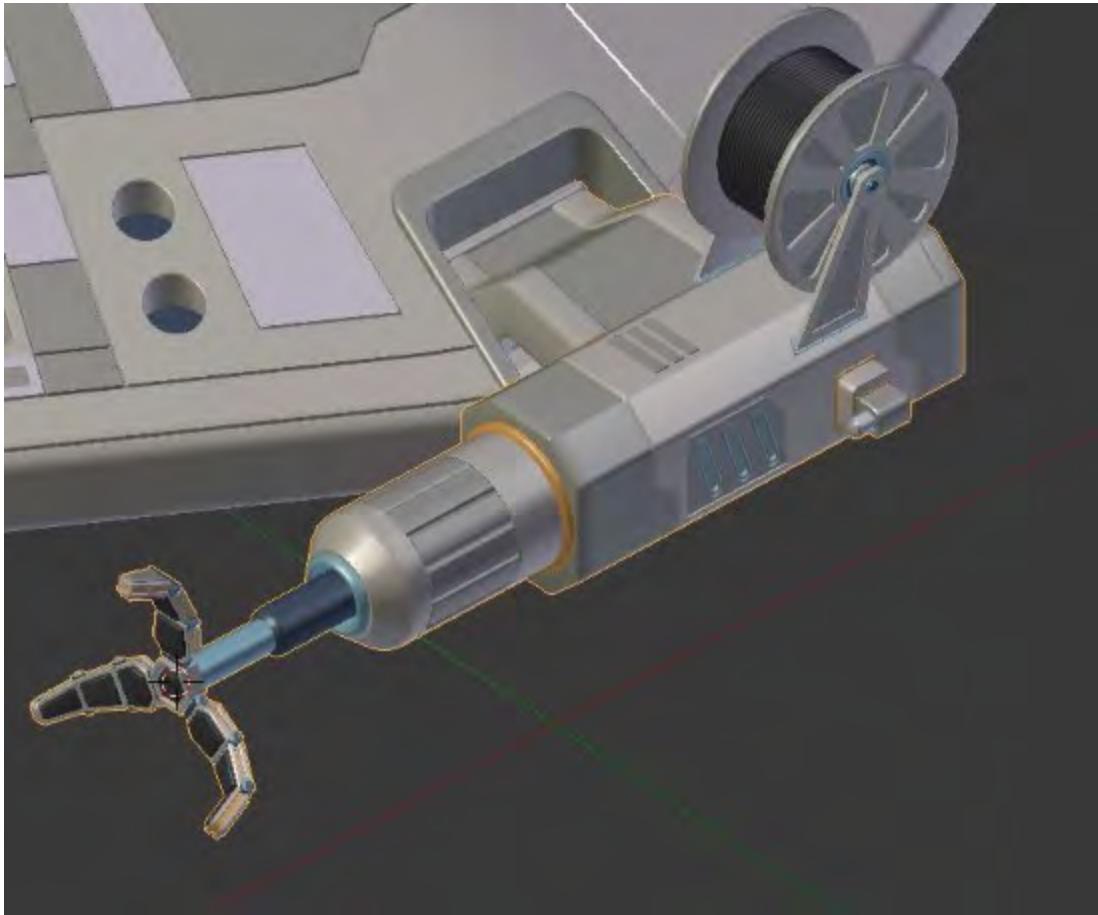
Before going further, let's add materials to our grappling gun. We'll do that now so that when we duplicate the finger, we won't have to do three times the work.

With materials added, we'll duplicate the finger of the grappling claw. Since I want three of them evenly spaced around a circle, I'll be rotating each copy around the center point by 120 degrees:



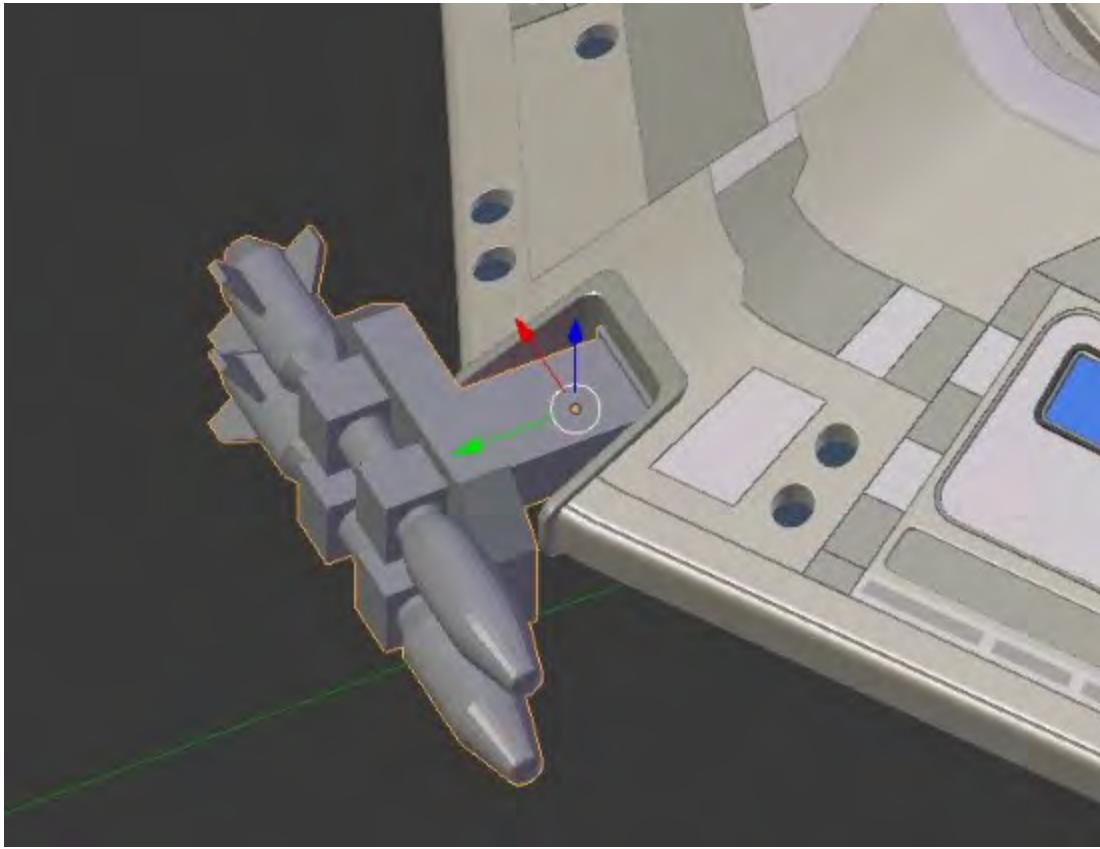
<pagebreak></pagebreak>

Then, we'll just do a middle part to link the three fingers, and we're done (for now) with our grappling gun:

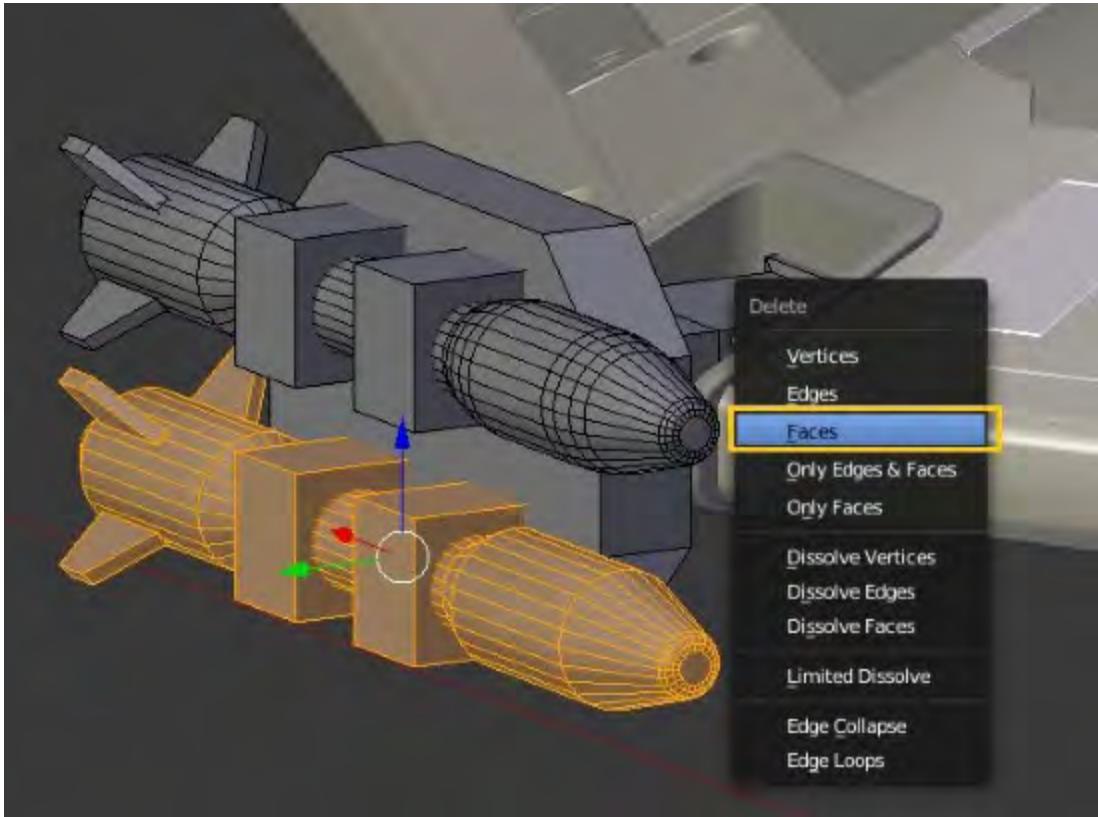


<pagebreak></pagebreak>

We'll work on the missile launcher next:



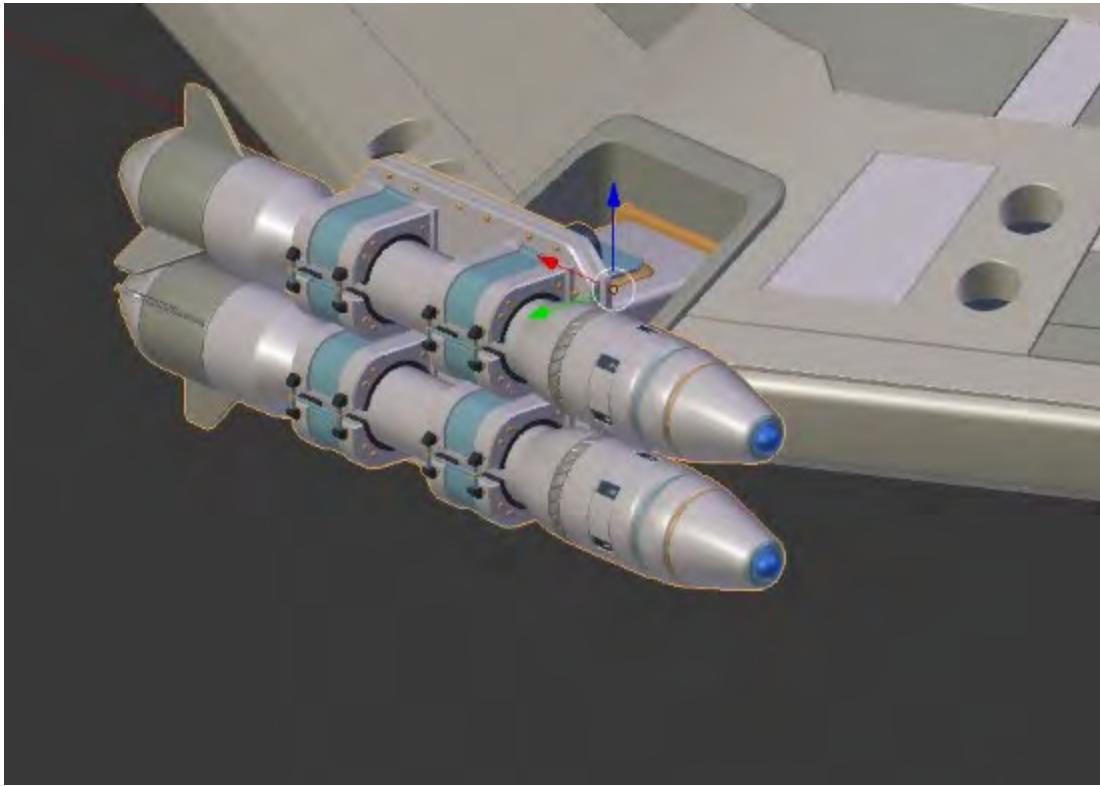
I'm going to start by just deleting one of the missiles. It's easier to just detail one of them and then duplicate it:



<pagebreak></pagebreak>

We won't cover the missile launcher step by step here, because it's not different (from a modeling standpoint) than the grappling gun we just did. In fact, it's not different than the pistol we made in the first few chapters of the book. Hopefully, all of these techniques are becoming very familiar to you by now.

Here's what I ended up doing with the missile launcher:



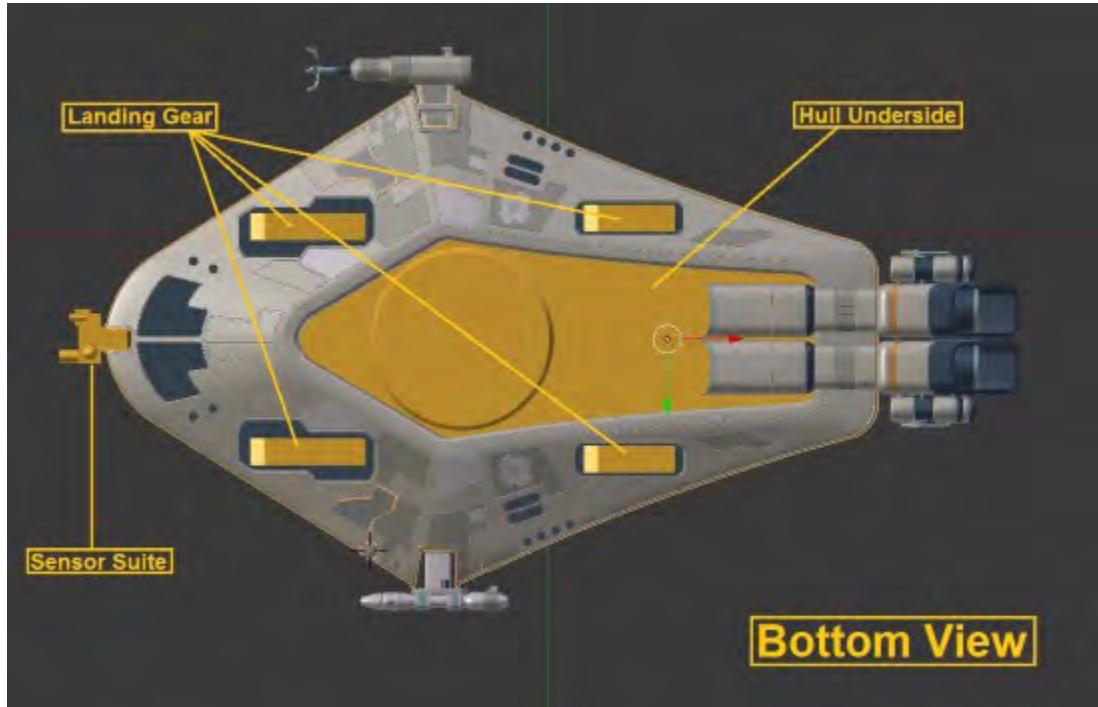
Note

I had to be very careful with the tail fins on the missiles. Because they're so close to each other, I had to use three fins and position them very carefully so that they didn't hit each other. If you want more than three fins, you may need to move them further apart. I don't know if any real military vehicle would be designed quite this way, but I thought it looked cool. Like everything else, that's up to you.

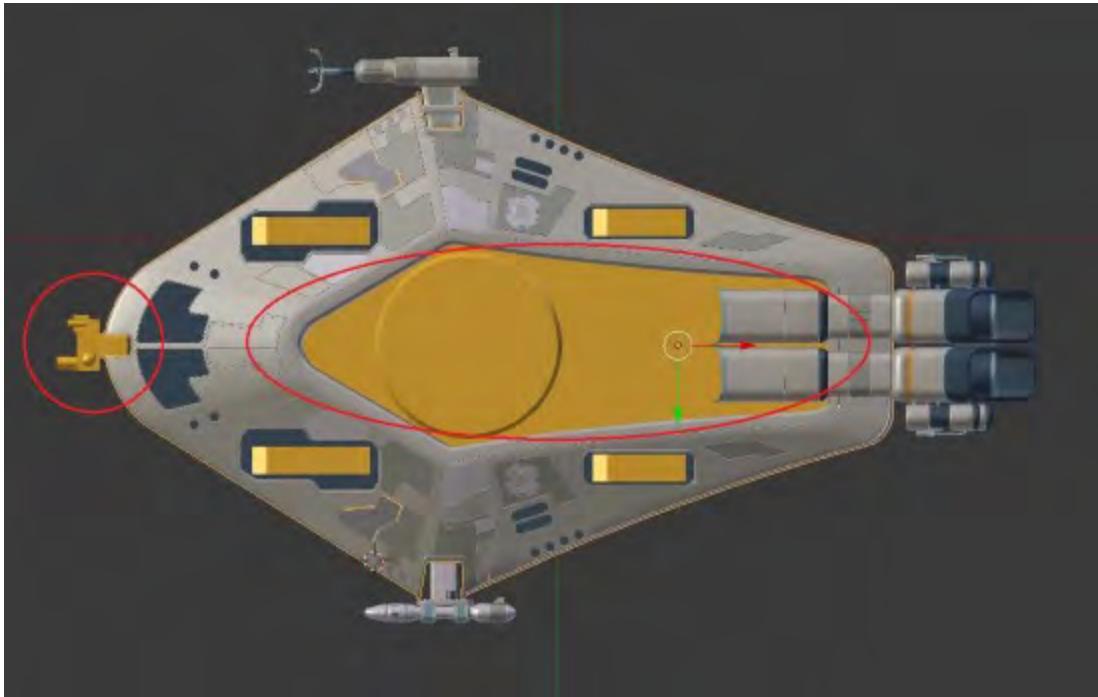
<pagebreak></pagebreak>

Do it yourself - completing the body

Next, let's take a look at the key areas that we have left to model:

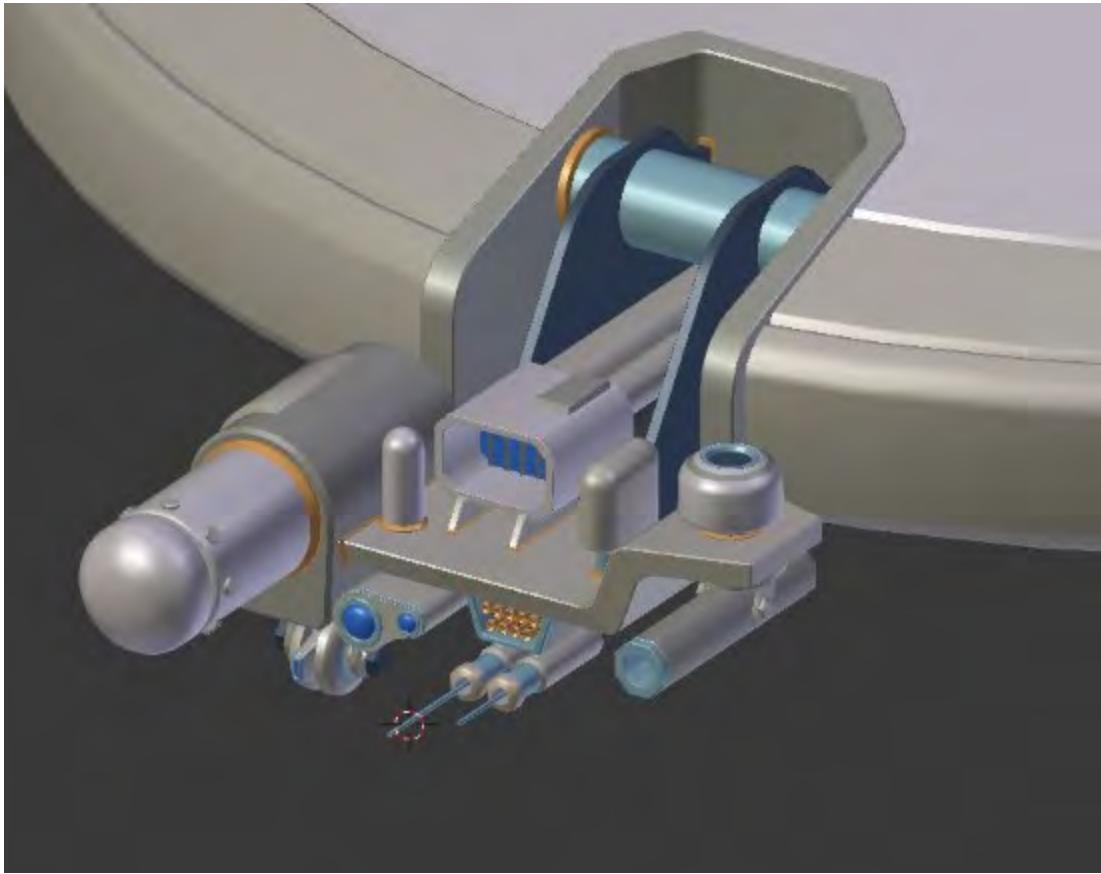


The bottom of the ship and the Sensor Suite (on the nose) are good opportunities to practice on your own. They use identical techniques to areas of the ship we've already done. Go ahead and see what you can do!



<pagebreak></pagebreak>

For the record, here's what I ended up doing with the Sensor Suite:



And here's what I did with the bottom. You can see that I copied that circular piece from the top of the engine area:



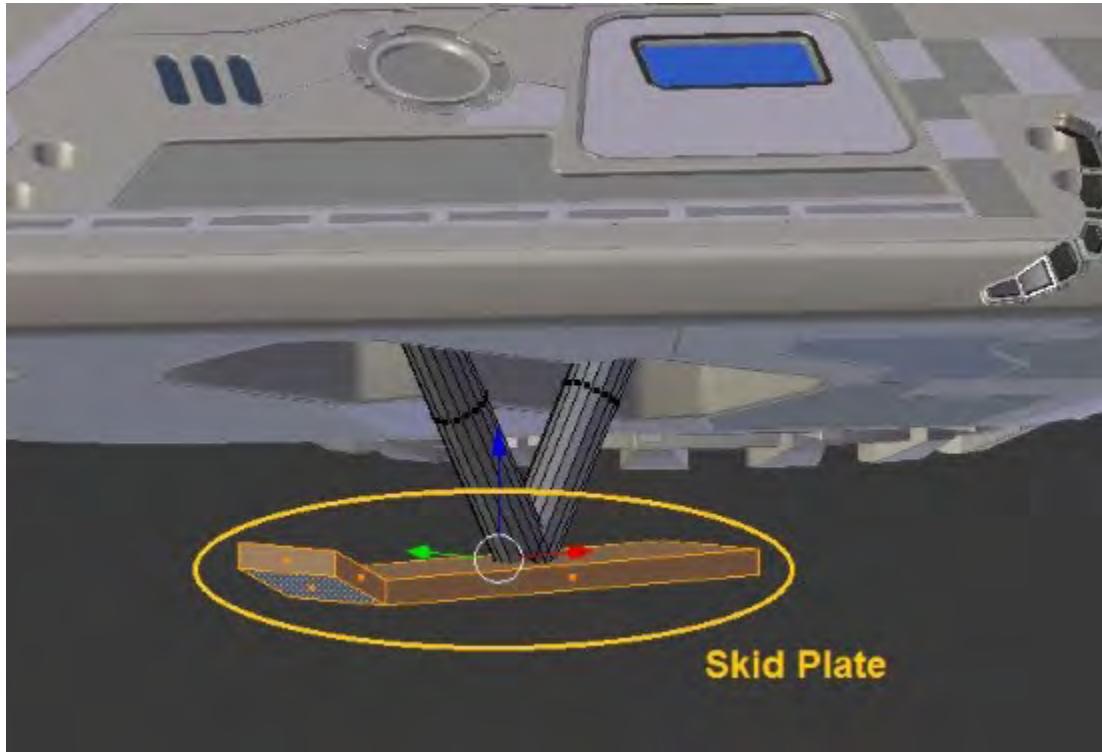
<pagebreak></pagebreak>

One of the nice things about a project like this is that you can start to copy parts from one area to another. It's unlikely that both the top and bottom of the ship would be shown in the same render (or shot), so you can probably get away with borrowing quite a bit. Even if you did see them simultaneously, it's not unreasonable to think that a ship would have more than one of certain components.

Of course, that's just a way to make things quicker (and easier). If you'd like everything to be 100% original, you're certainly free to do so.

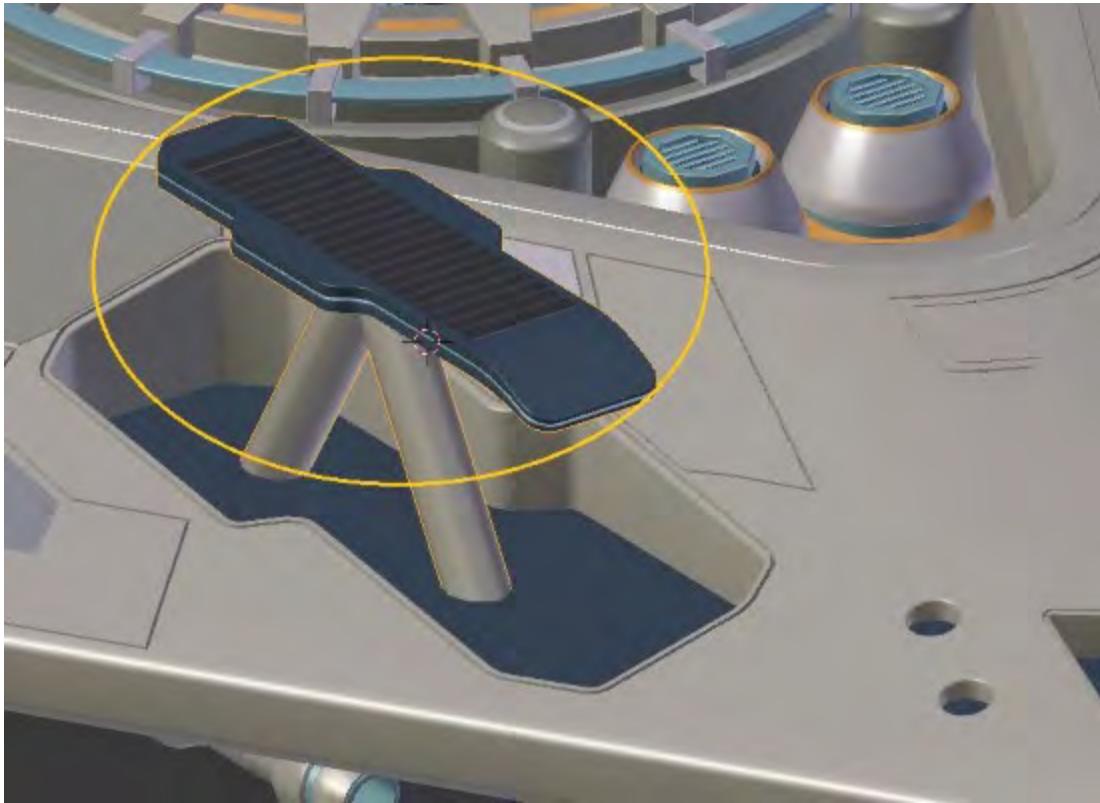
Building the landing gear

We'll do the landing struts together, but you can feel free to finish off the actual skids yourself:



<pagebreak></pagebreak>

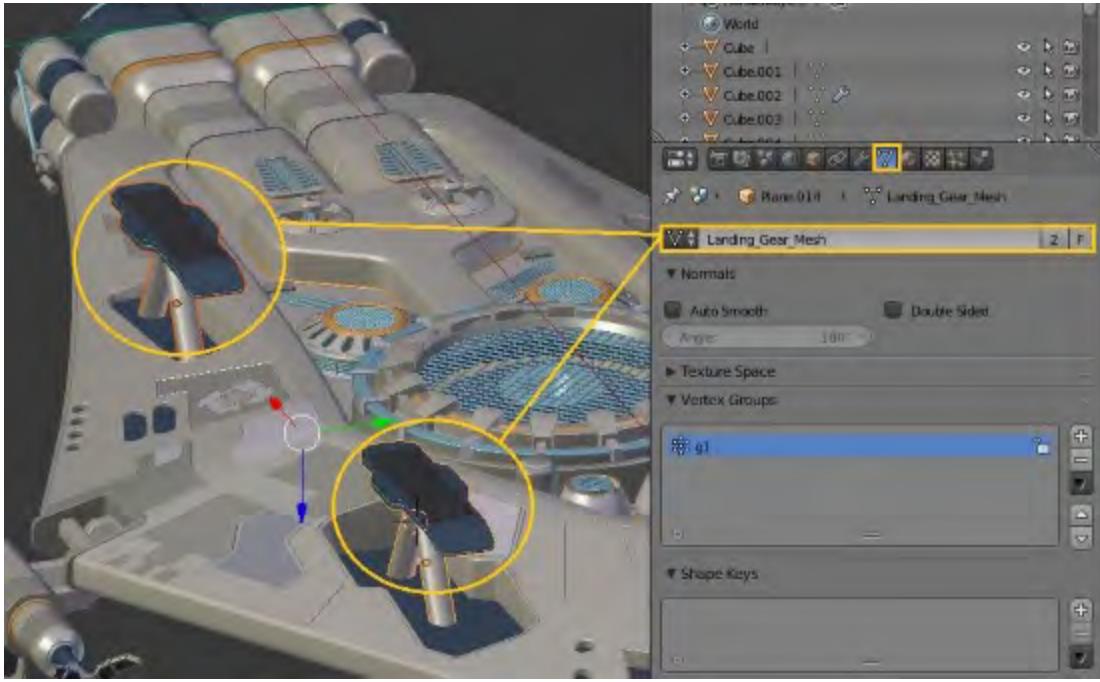
I kept mine pretty simple, compared to other parts of the ship:



Once you've got the skid plate done, make sure to make it a separate object (if it's not already). We're going to use a neat trick to finish this up.

<pagebreak></pagebreak>

Make a copy of the landing gear part and move it to the rear section (or front, if you modeled the rear). Then, under your **Mesh** tab, you can assign both of these objects the same mesh data:

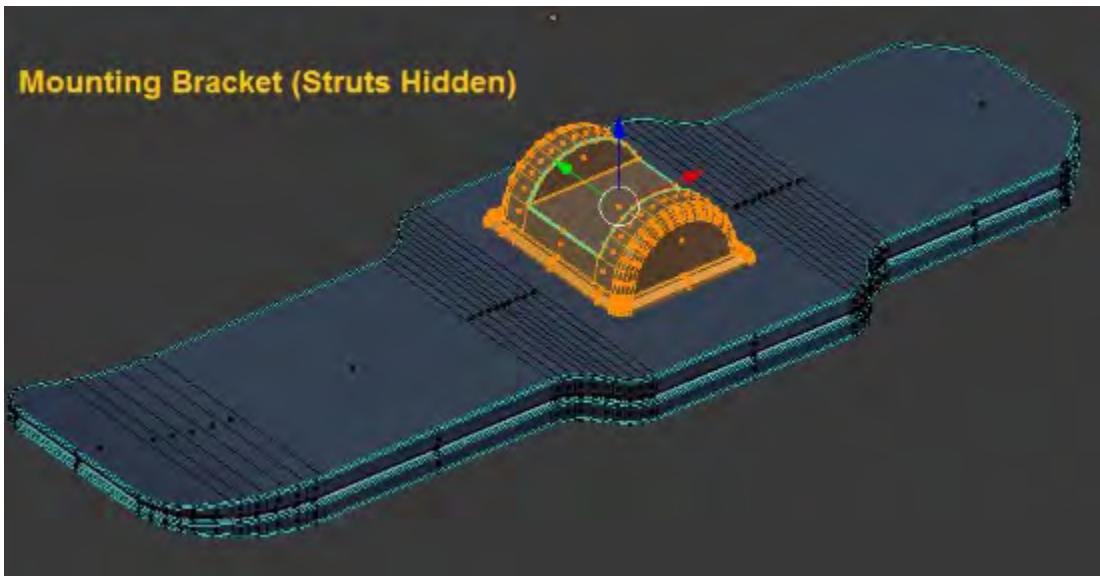


Now, whenever you make a change to one of them, the change will carry over to the other as well.

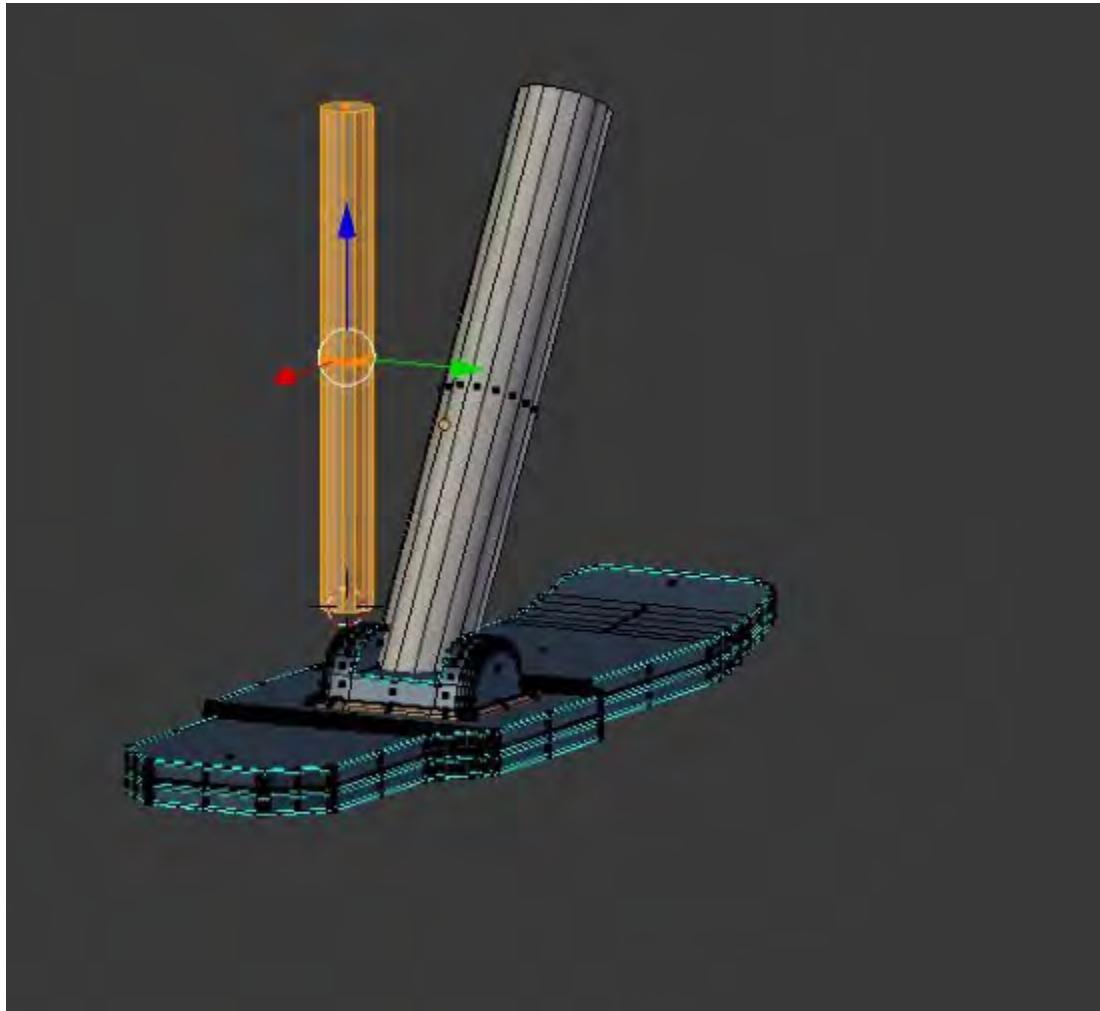
Of course, you could just model one and then duplicate it, but sometimes it's nice to see how the part will look in multiple locations. For instance, the cutouts are slightly different between the front and back of the ship. As you're modeling it, you'll want to make sure that it will fit both areas.

<pagebreak></pagebreak>

The first detail we'll add is a mounting bracket for our struts to go on:

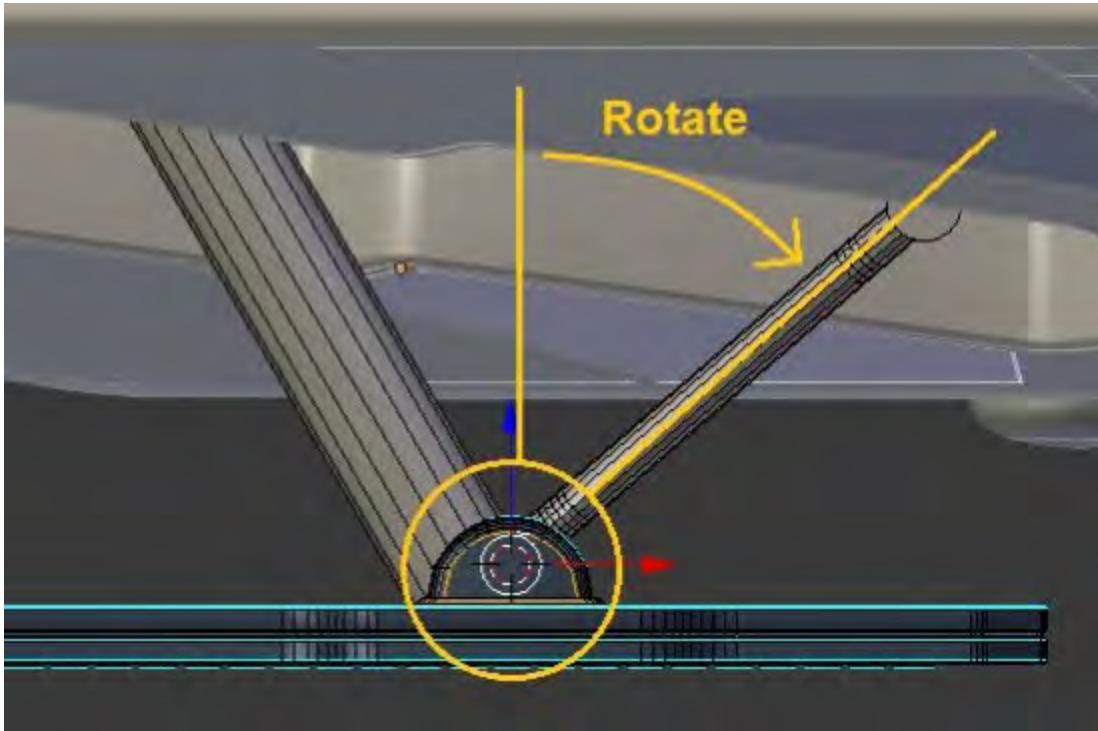


Then, we'll add a small cylinder (the large one, at this point, is just a placeholder):

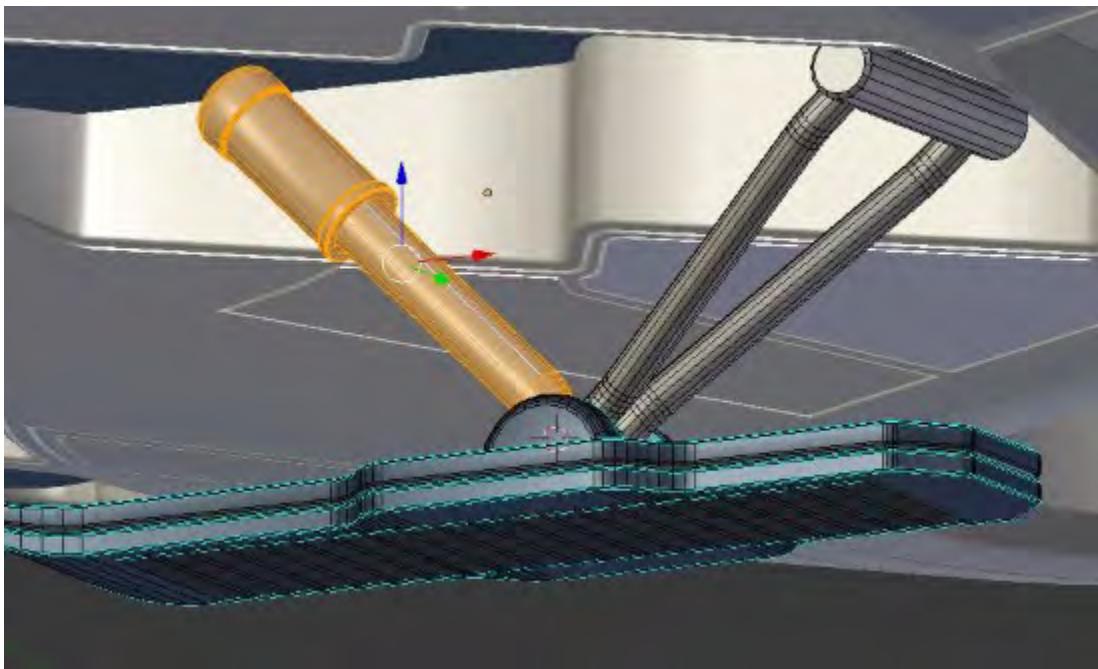


<pagebreak></pagebreak>

We'll rotate it just a bit:



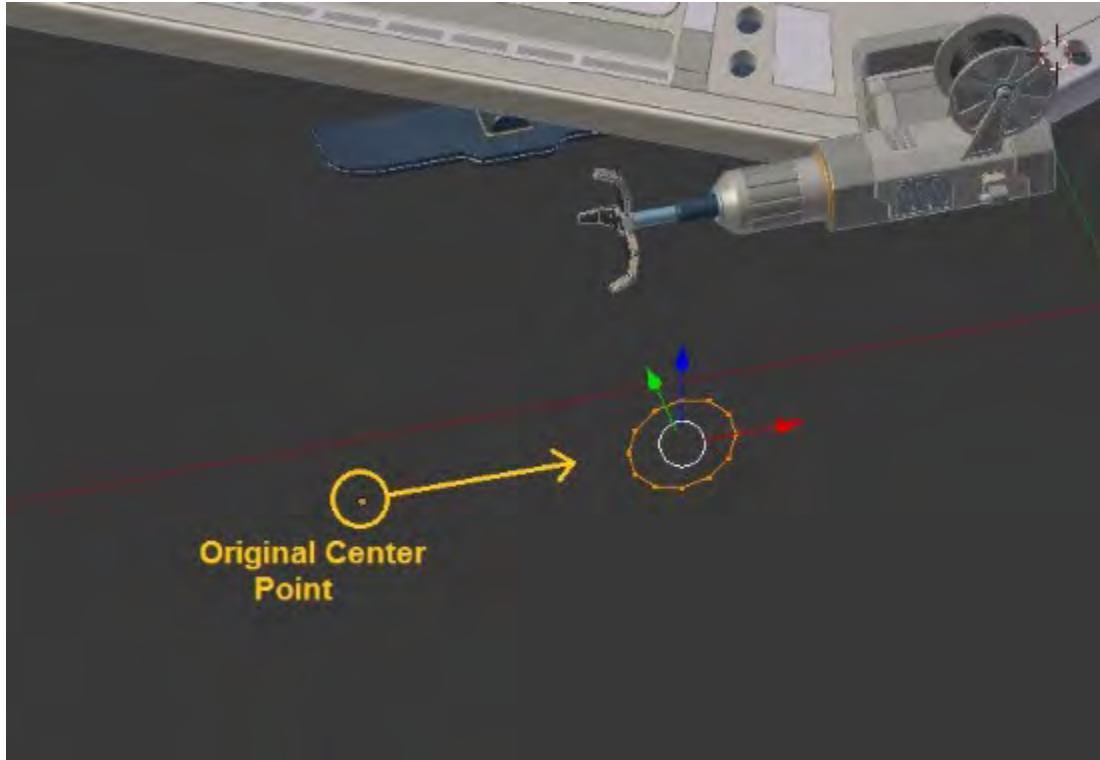
From this, it's pretty easy to create a rear mounting piece. Once you've done that, go ahead and add a shock absorber for the front (leave room for the springs, which we'll add next):



<pagebreak></pagebreak>

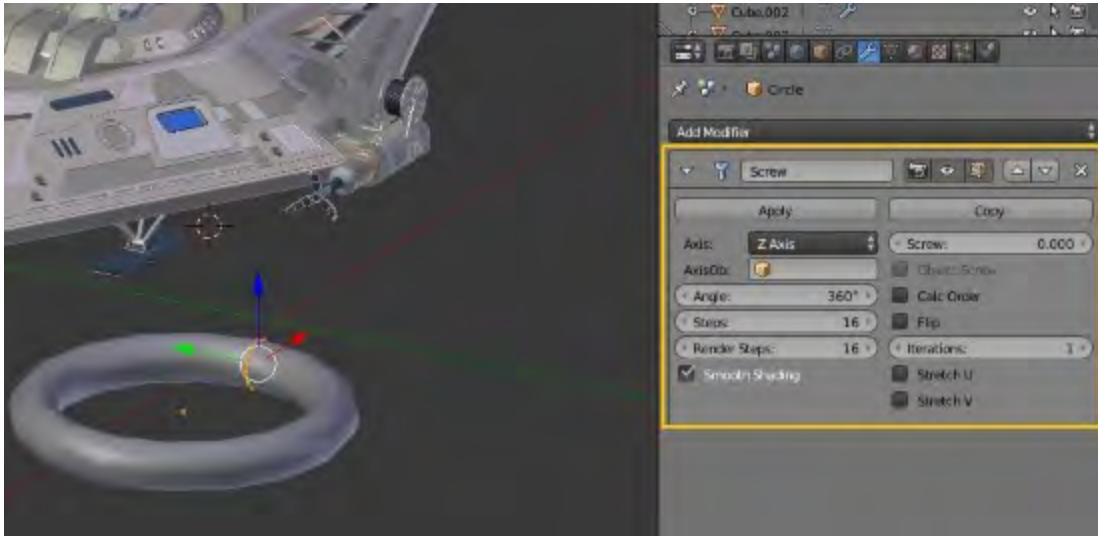
To create the spring, we'll start with a small (12-sided) circle. We'll make it that small because, just like the cable reel on the grabbling gun, there will naturally be a lot of geometry and we want to keep the polygon count as low as we can.

Then, in **Edit Mode**, move the whole circle away from its original center point:

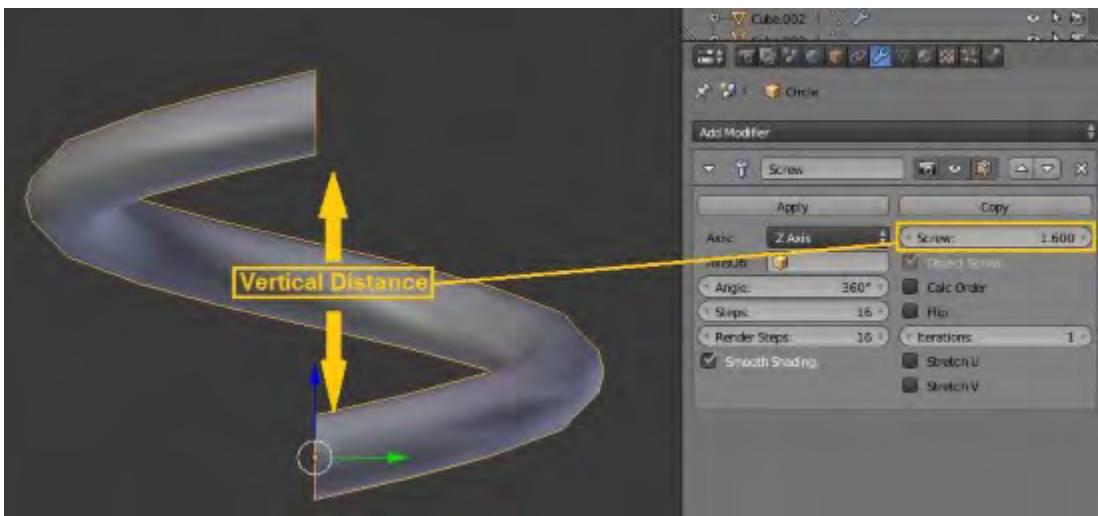


<pagebreak></pagebreak>

Having done that, you can now add a **Screw** modifier. Right away, you'll see the effect:

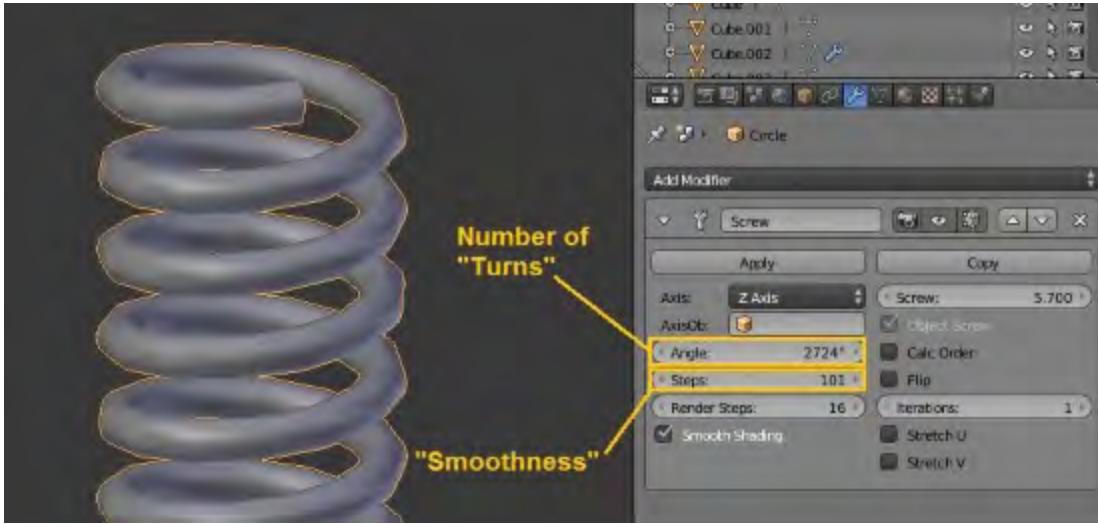


There are a couple of settings you'll want to make note of here. The **Screw** value controls the vertical gap or distance of your spring:

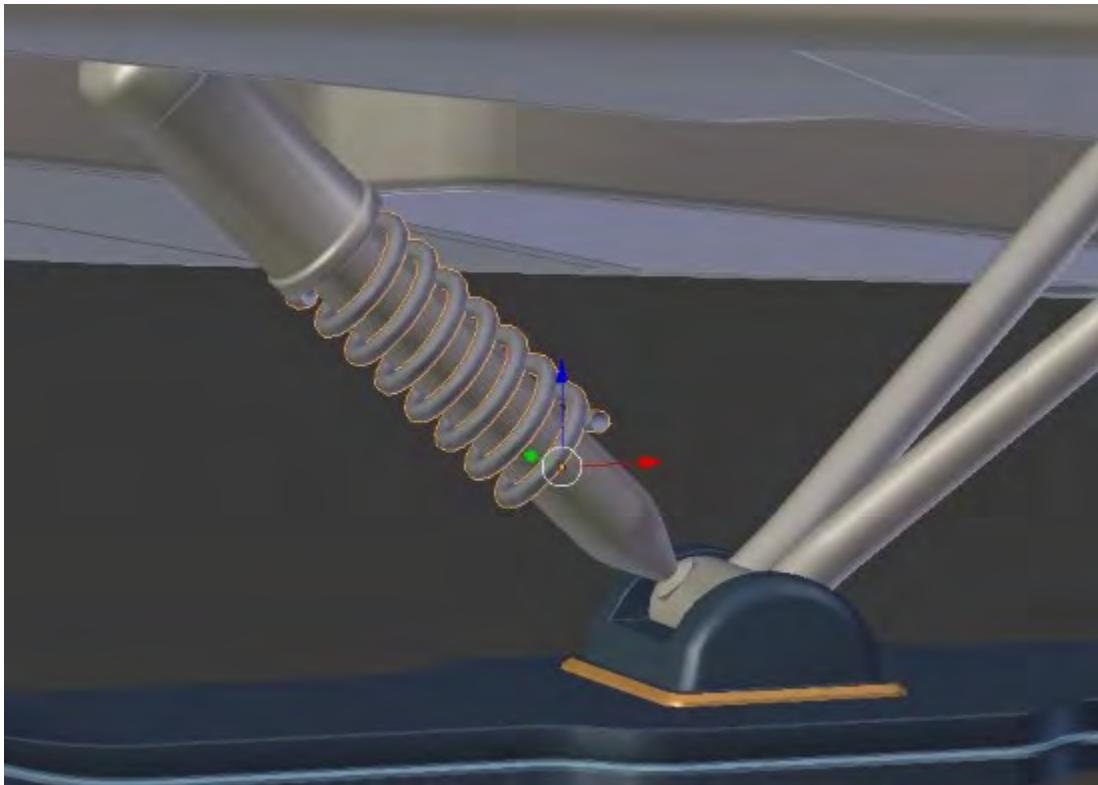


<pagebreak></pagebreak>

The **Angle** and **Steps** values control the number of turns and smoothness, respectively. If you experience any issues with **Normals** being inverted, you can use the **Calc Order** or **Flip** options on the modifier to help correct them:



Go ahead and play with these until you're happy, then move and scale your spring into position. Once it's the way you like it, go ahead and apply the **Screw** modifier (but don't join it to the shock absorber just yet):

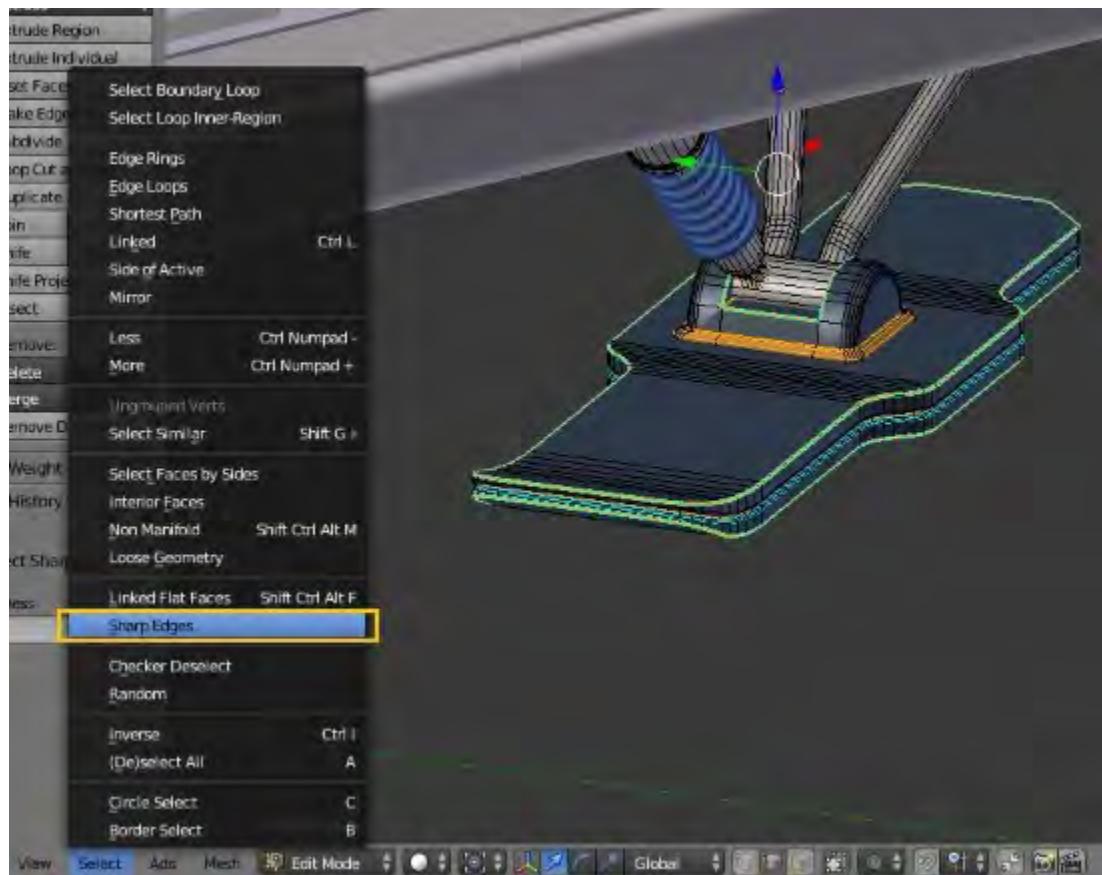


Note

None of my existing materials seemed right for the spring. So, I went ahead and added one that I called Blue Plastic.

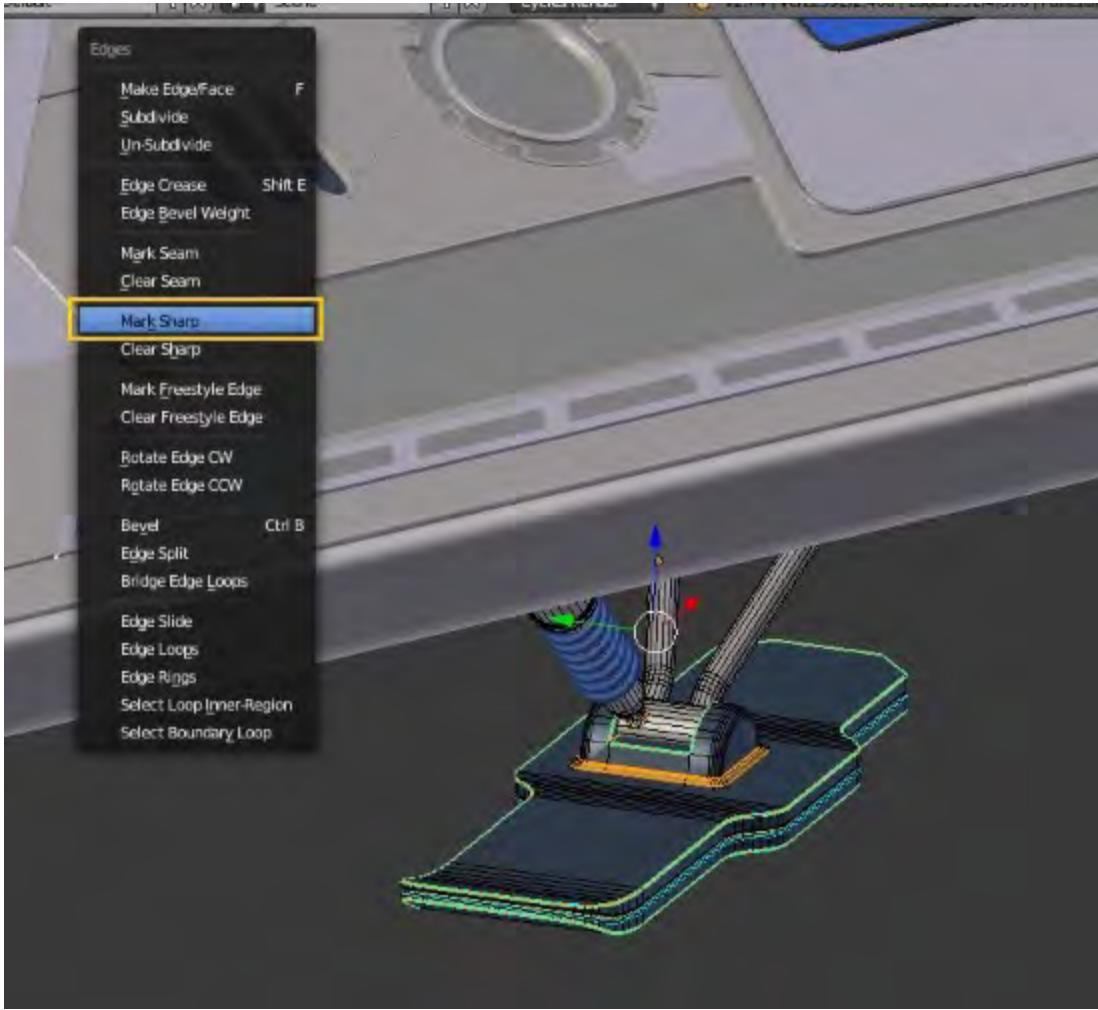
At this point, we have a bit of a problem. We want to join the spring to the landing gear, but we can't. The landing gear has an **Edge Split** modifier with a **Split Angle** value of 30, and the spring has a value of 46. If we join them right now, the smooth edges on the spring will become sharp. We don't want that.

Instead, we'll go to our shock absorber. Using the **Select** menu, we'll pick the **Sharp Edges** option:



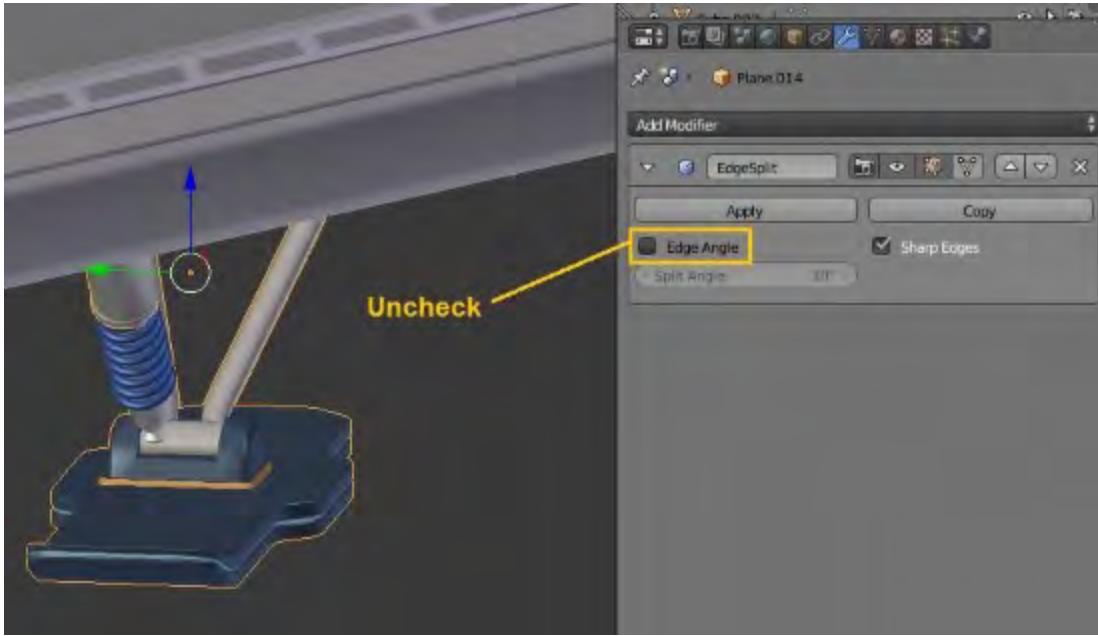
<pagebreak></pagebreak>

By default, it will select all edges with an angle of 30 degrees or higher. Once you do that, go ahead and mark those edges as sharp:

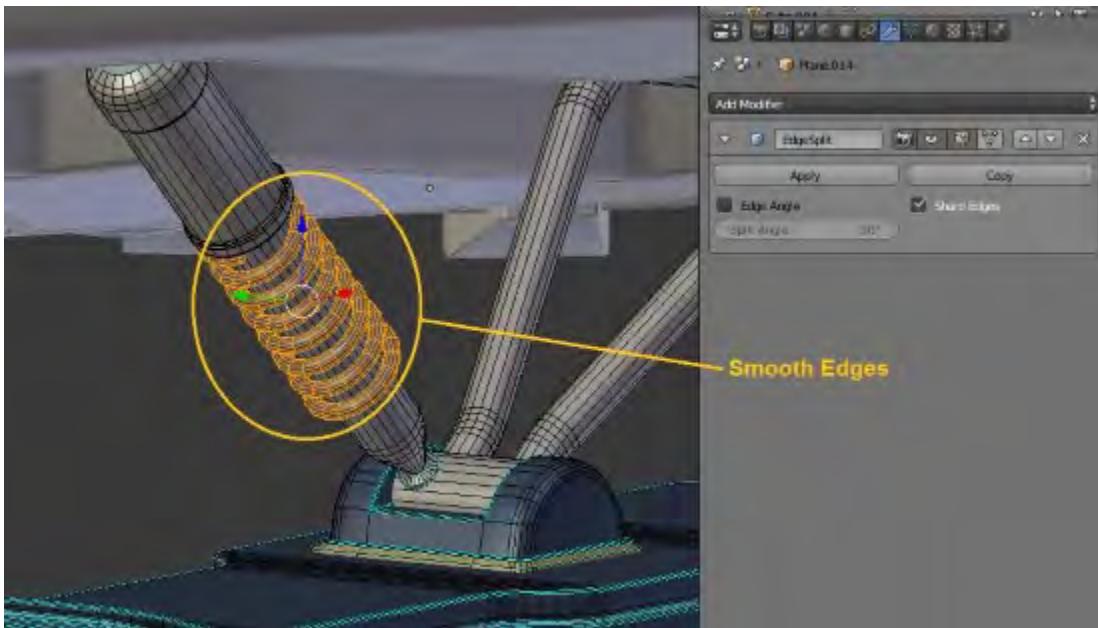


<pagebreak></pagebreak>

Since all the 30 degrees angles are marked sharp, we no longer need the **Edge Angle** option on our **Edge Split** modifier. You can disable it (unchecked) and the landing gear remains exactly the same:



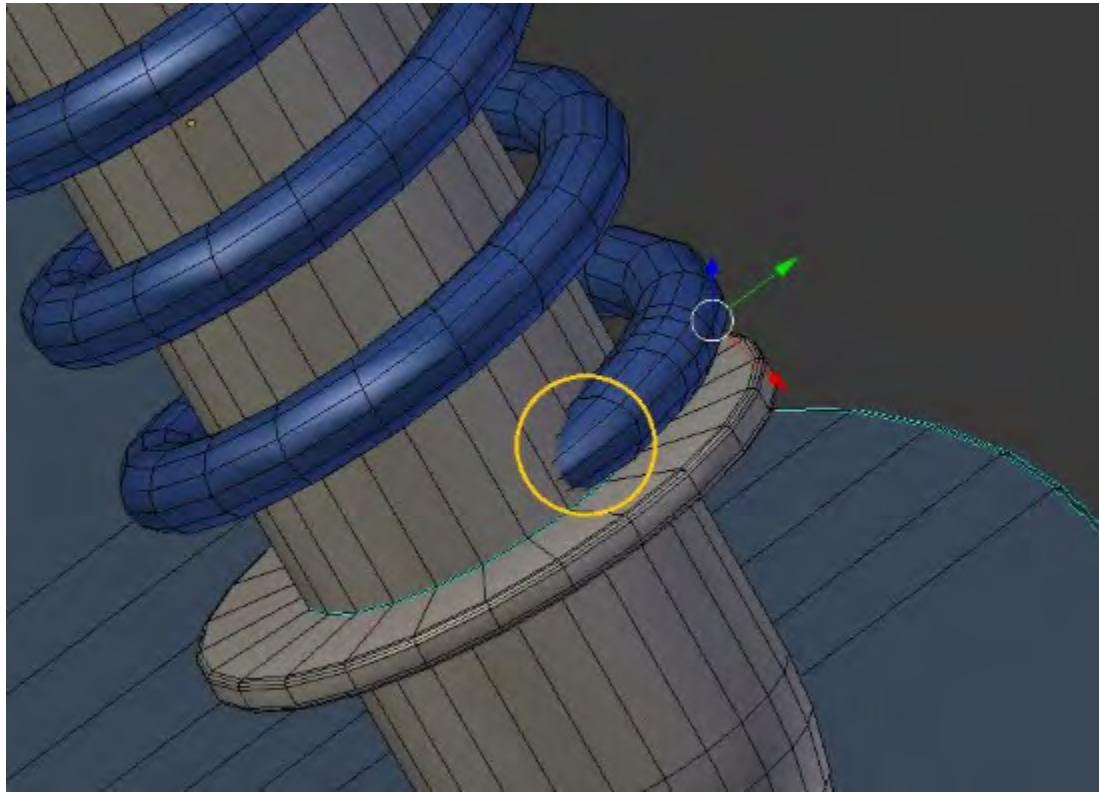
Now, you can join the spring to it without a problem:



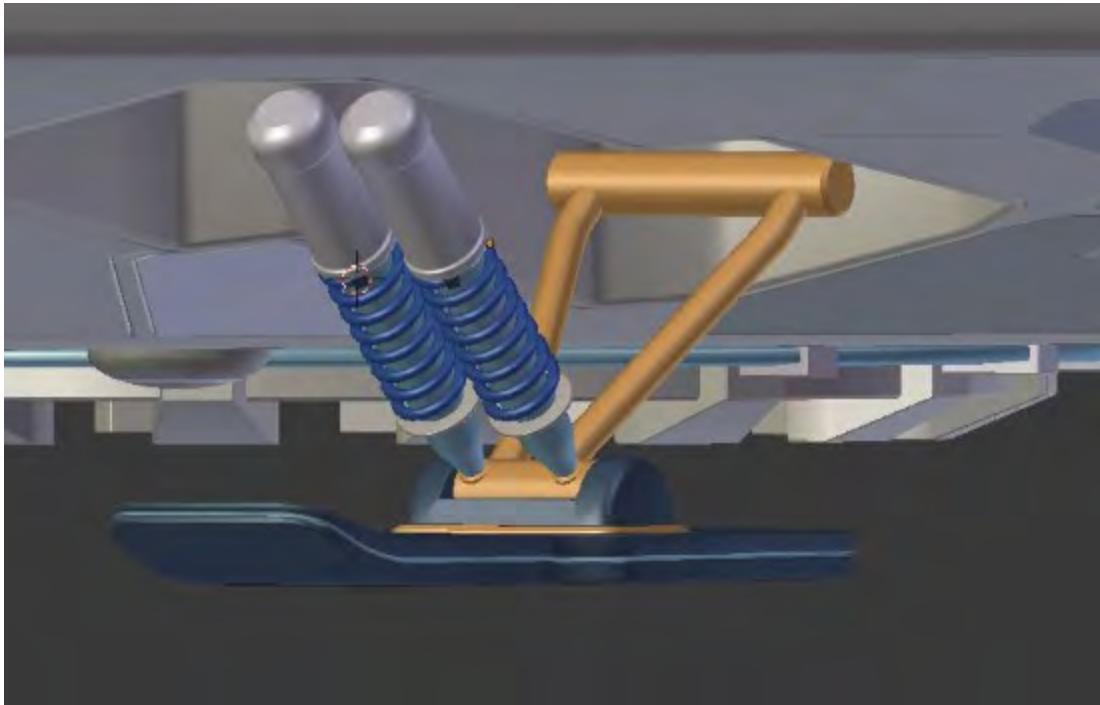
Of course, this does mean that when you create new edges in your landing gear, you'll now have to mark them as sharp. Alternatively, you can keep the **Edge Angle** option selected and just turn it up to 46 degrees—your choice.

<pagebreak></pagebreak>

Next, we'll just pull the ends of our spring in a little so that they don't stick out:



And maybe we'll duplicate it. This is a big, heavy vehicle, after all, so maybe it needs multiple shock absorbers:

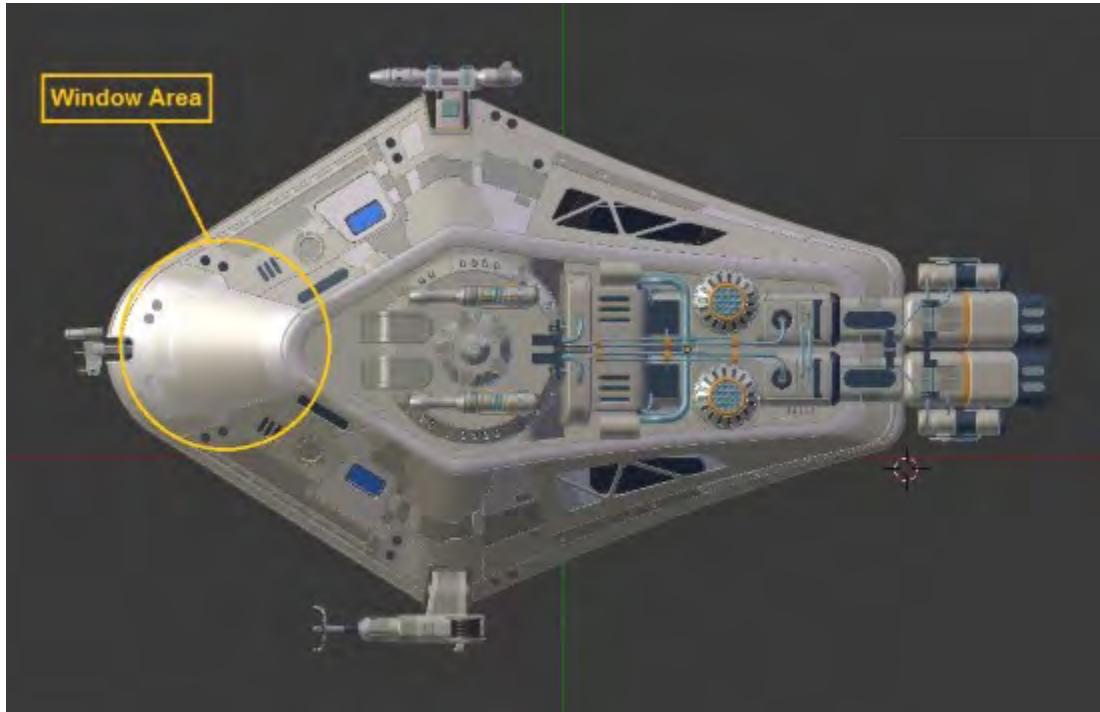


That's a good place to leave our landing gear for now, I think.

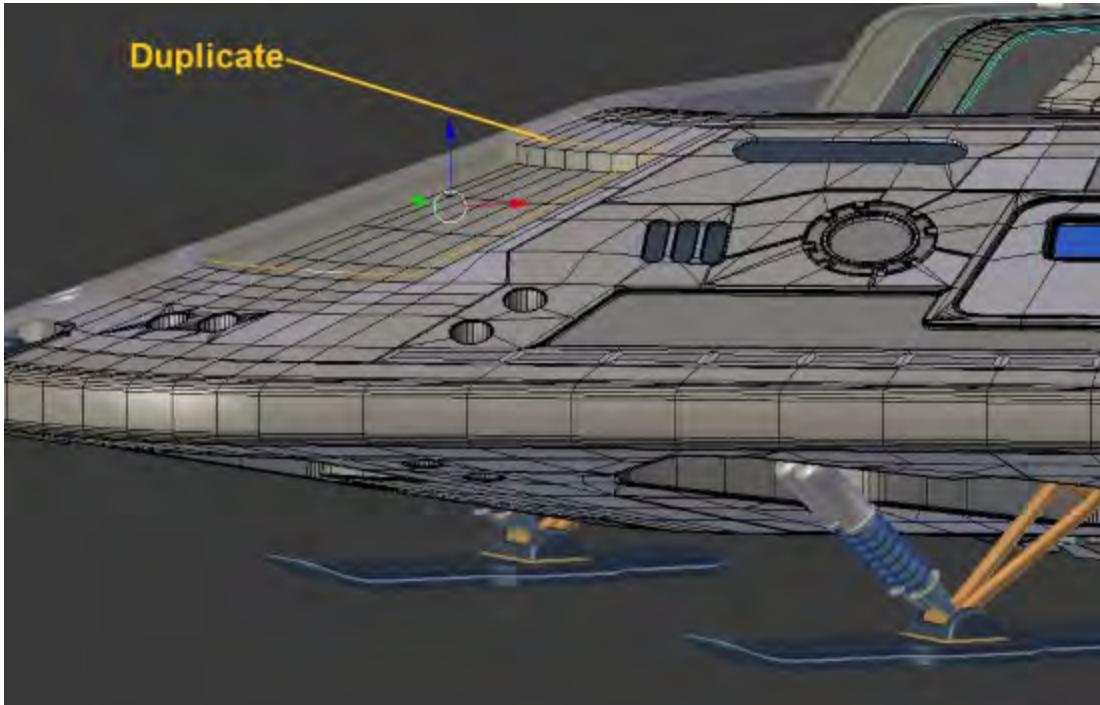
<pagebreak></pagebreak>

Adding the cockpit

One of the last major areas we need to do is our front window:

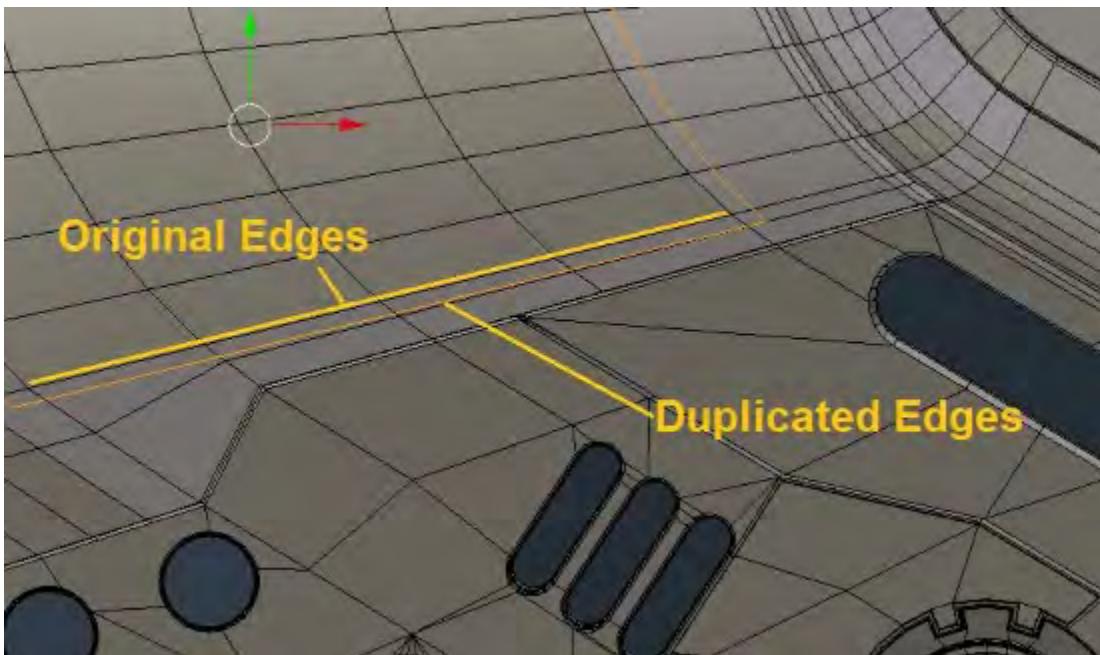


There are a number of ways we could do this, but let's try something a bit different. We can just duplicate the ring of edges that makes up the window area:

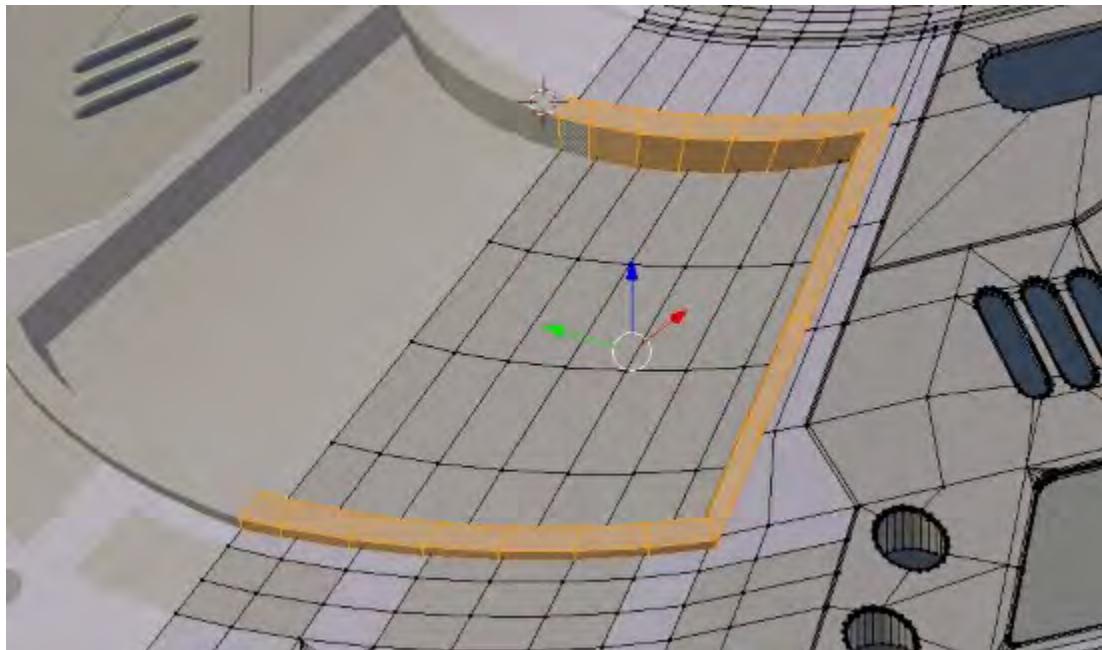


<pagebreak></pagebreak>

Then, we'll scale those duplicate edges up just a little:

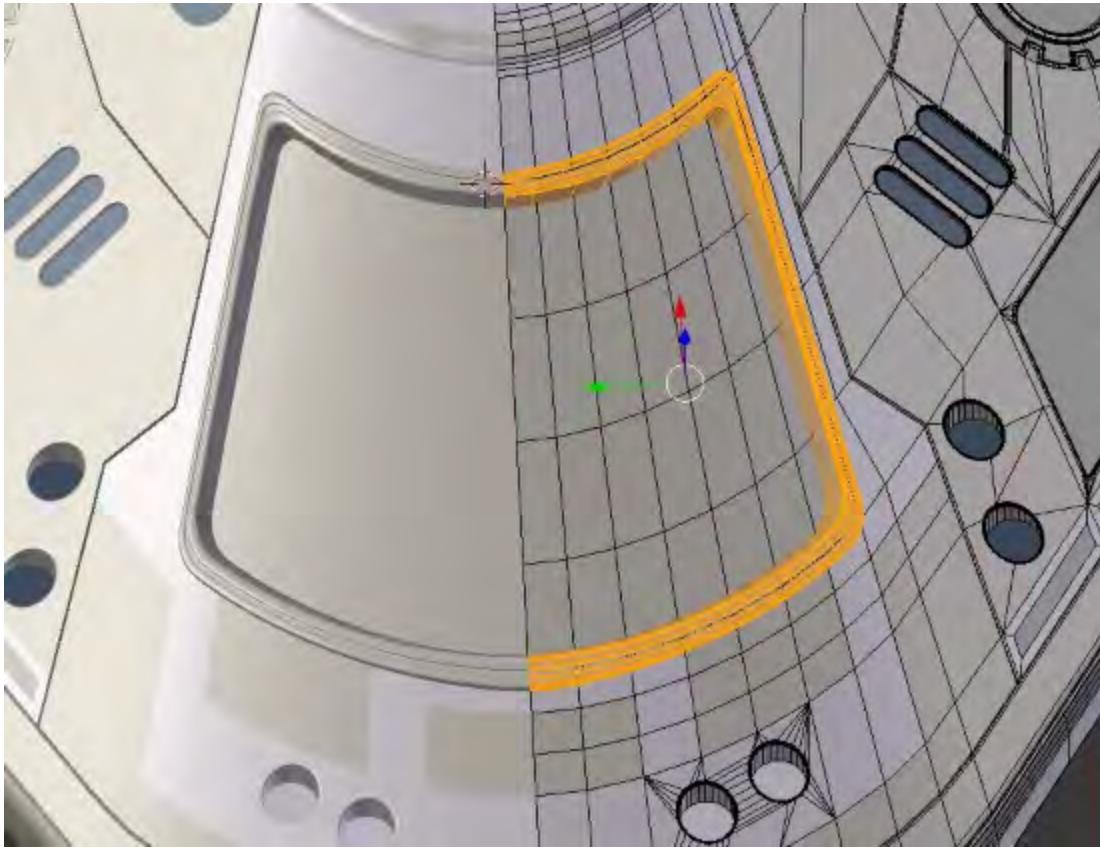


With the **Inset** and **Extrude** tools, you can now create a nice border for your window area:



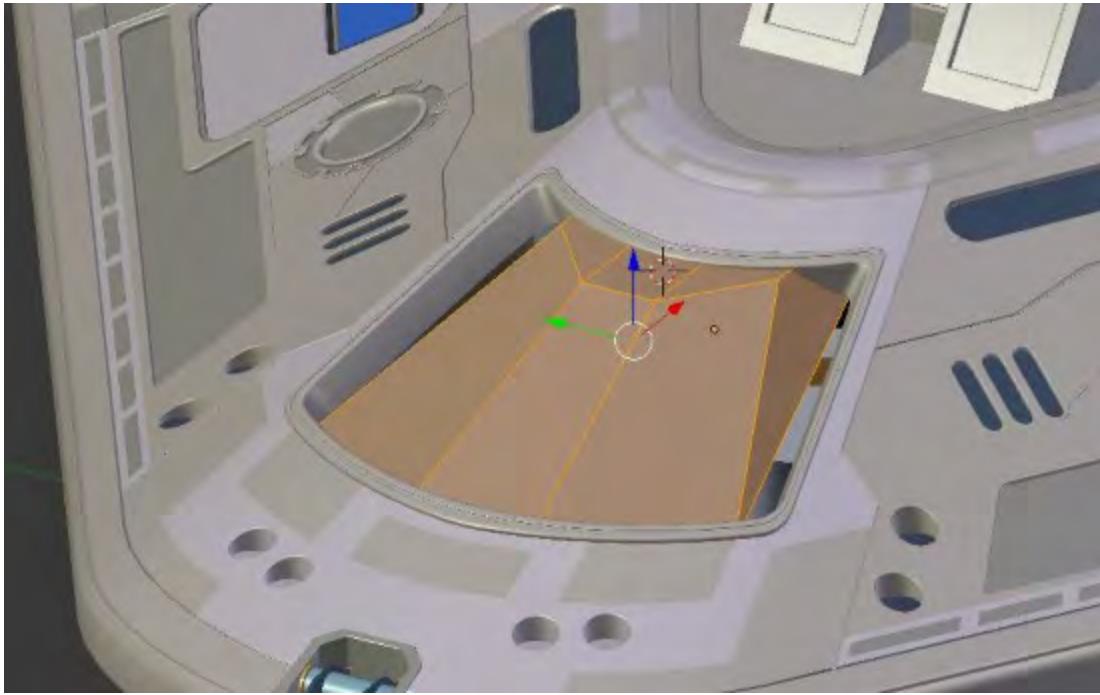
<pagebreak></pagebreak>

And we'll just bevel this so that it looks a little nicer:



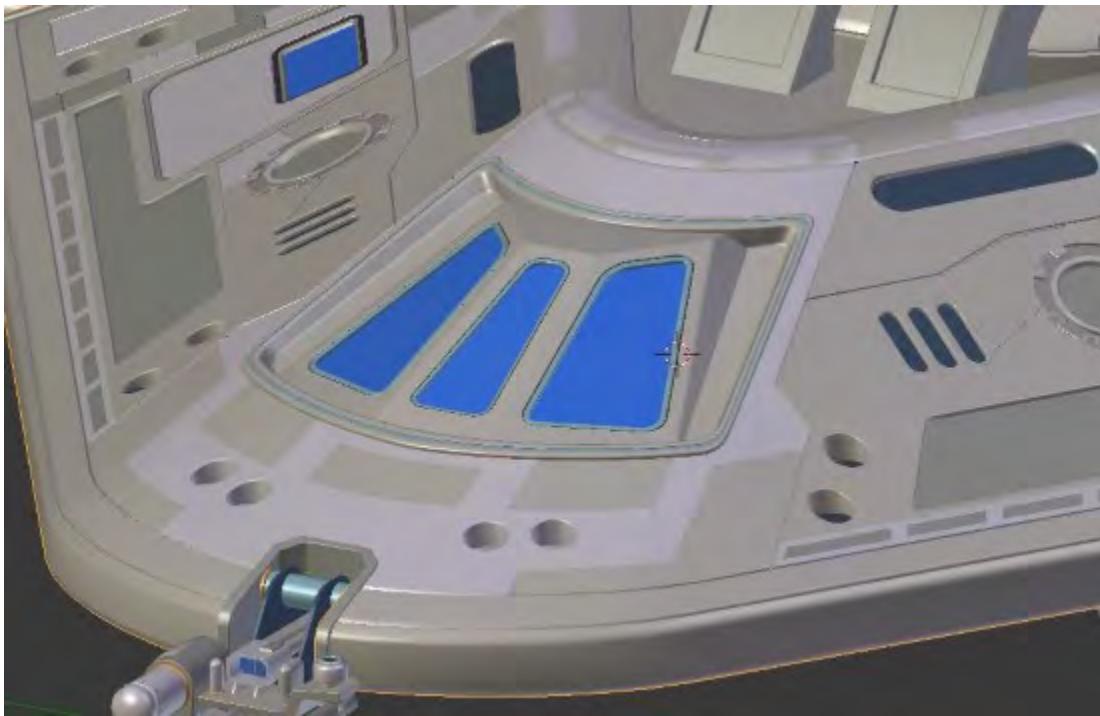
<pagebreak></pagebreak>

Then, I'll delete the back of the window faces and add a new object to serve as my cockpit:



<pagebreak></pagebreak>

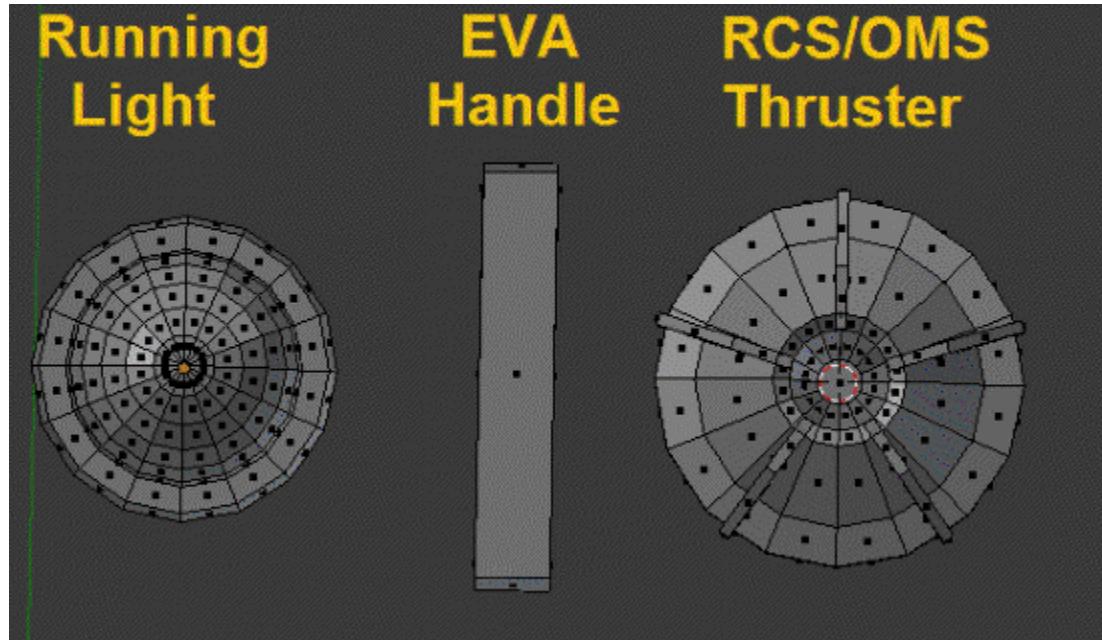
With a little beveling and extrusion, we can make it look unique:



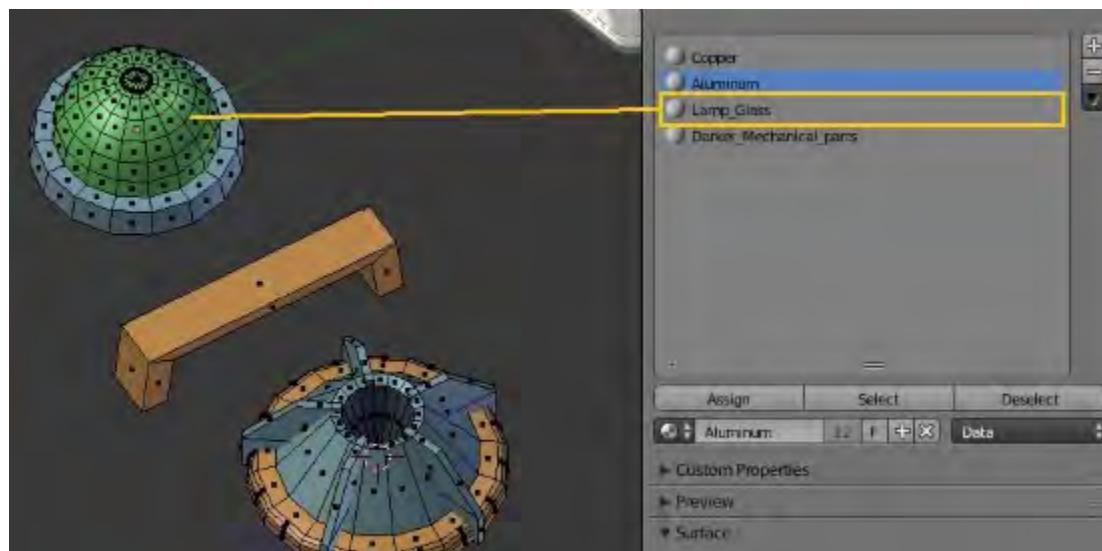
<pagebreak></pagebreak>

Creating small details

Next, let's make some small parts. These things will be duplicated and placed all over the ship, so it's easier to just model one of each and then move them into position. The more copies of each one that you plan to make, the lower you should keep the polygon count:

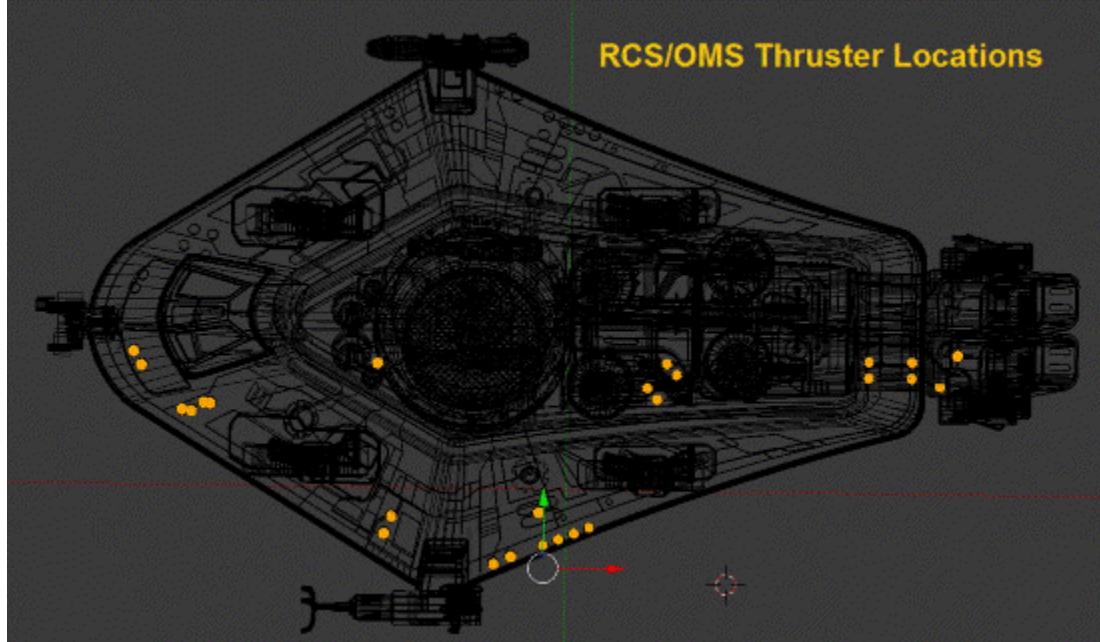


Before duplicating, of course, we'll want to add our materials:



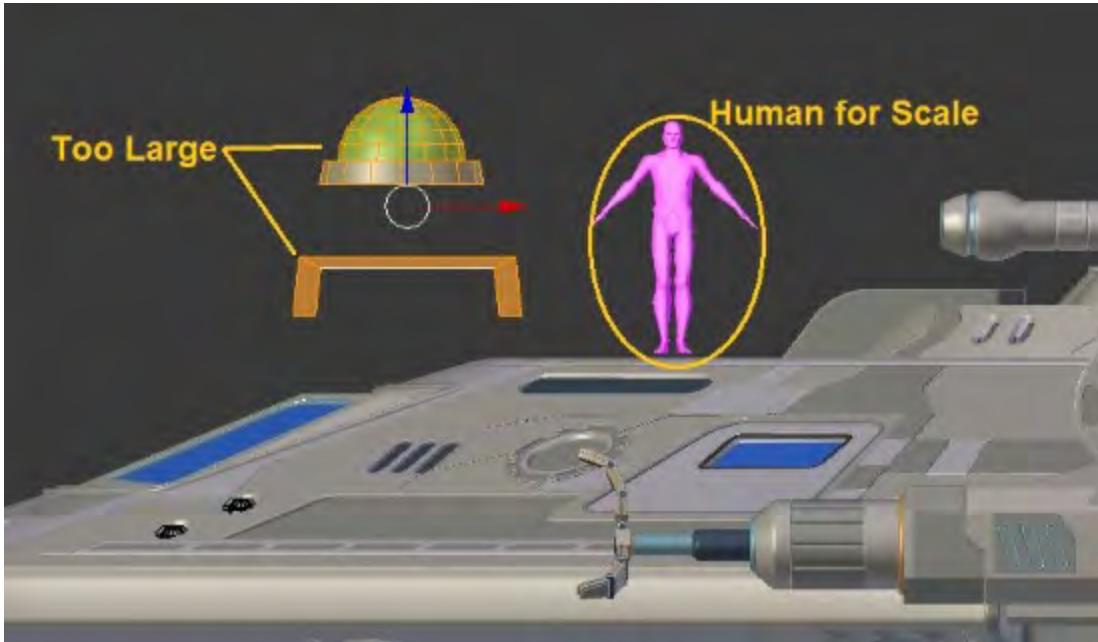
<pagebreak></pagebreak>

Here's where I ended up putting the RCS/OMS thrusters (on 1/2 the ship):



<pagebreak></pagebreak>

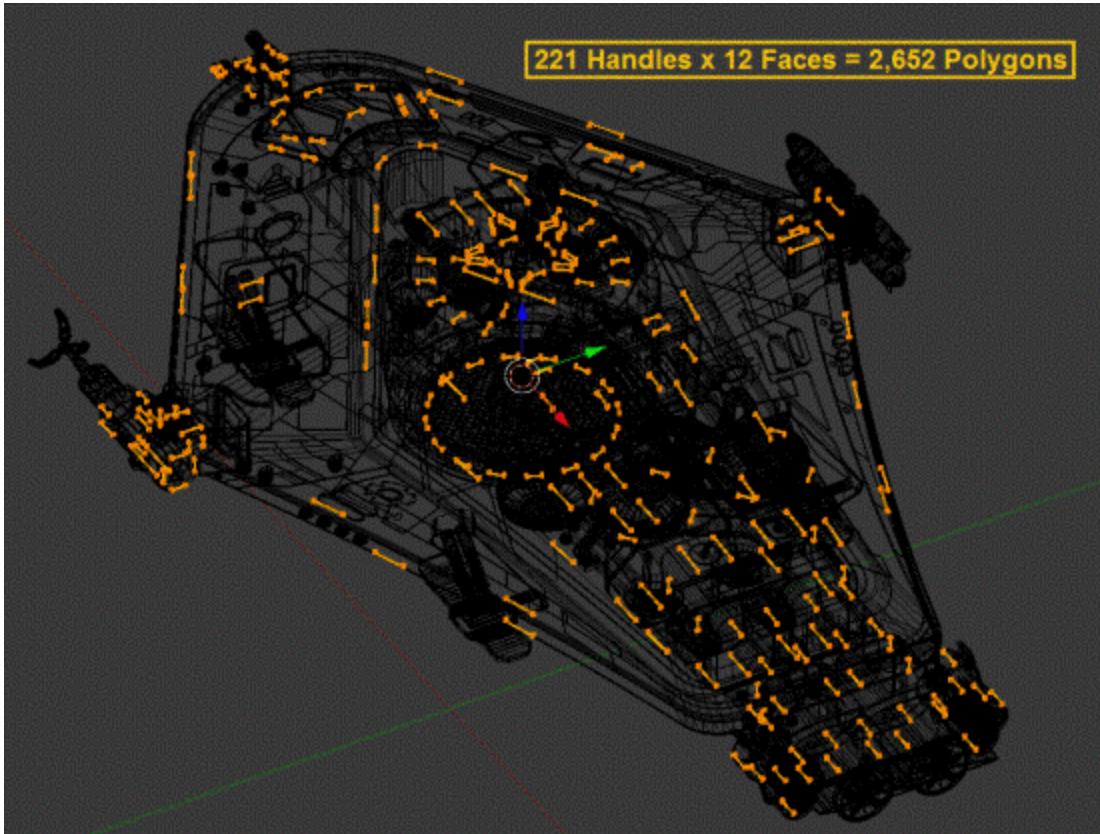
The thrusters can be any size (scale) that they need to be, but the handles and lights are a little different. These should be related to the size of a human. To make sure we get this right before making a lot of copies, we can just add a quick human model to our scene (or at the very least, just a cube that's as tall as a person would be):



This will give you a good sense of scale and allow you to get your little details right. For instance, it wouldn't make sense if the EVA handles were larger than a human. Nor would it make sense if the running lights were larger than a bathtub. So, we can scale these down to the correct size.

<pagebreak></pagebreak>

I knew I was going to make a lot of EVA handles, which is why I kept them so small:



Another thing I wanted to do was add some very small mechanical parts to the front of the ship. It's easier to model these things when they're flat (aligned with the X/Y/Z axes), so we'll use duplicate meshes again.

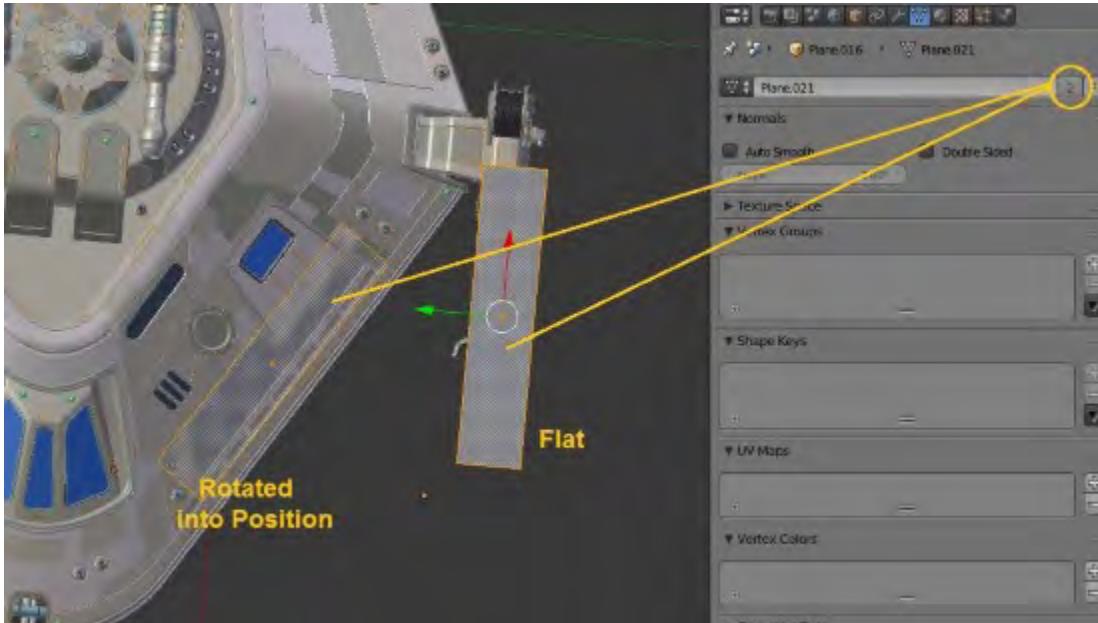
<pagebreak></pagebreak>

First, add two **Planes**:



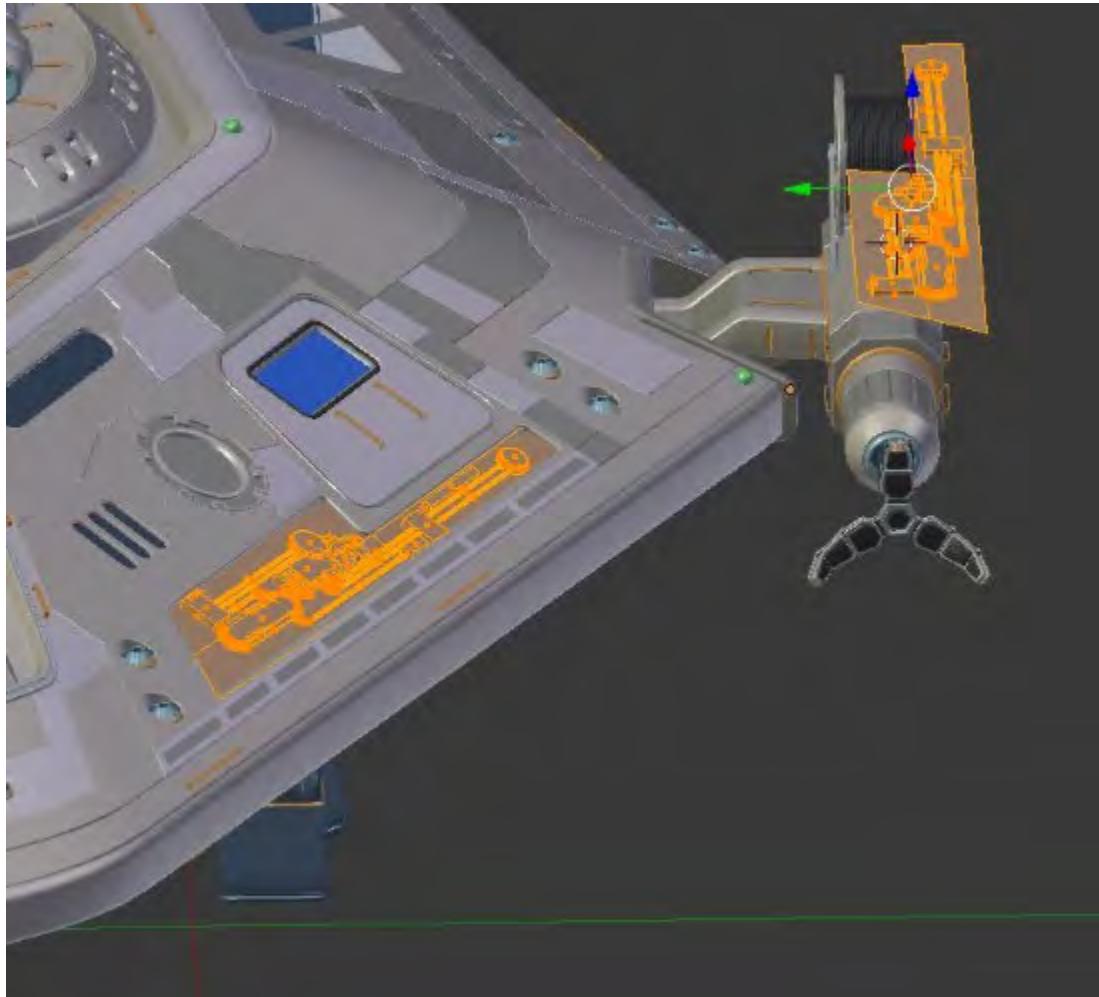
<pagebreak></pagebreak>

Then, assign them the same mesh data, just like we did with the landing gear. After you do that, you can rotate one of them into position:



<pagebreak></pagebreak>

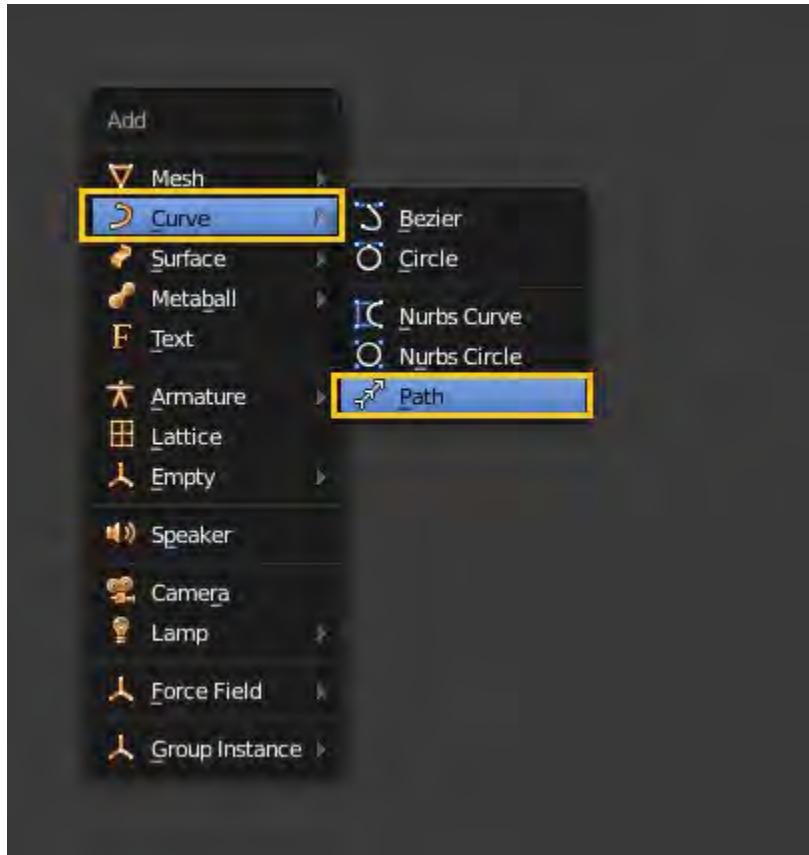
Now, you can model the other one, which is quite a bit easier, and the detail will show up on both:



<pagebreak></pagebreak>

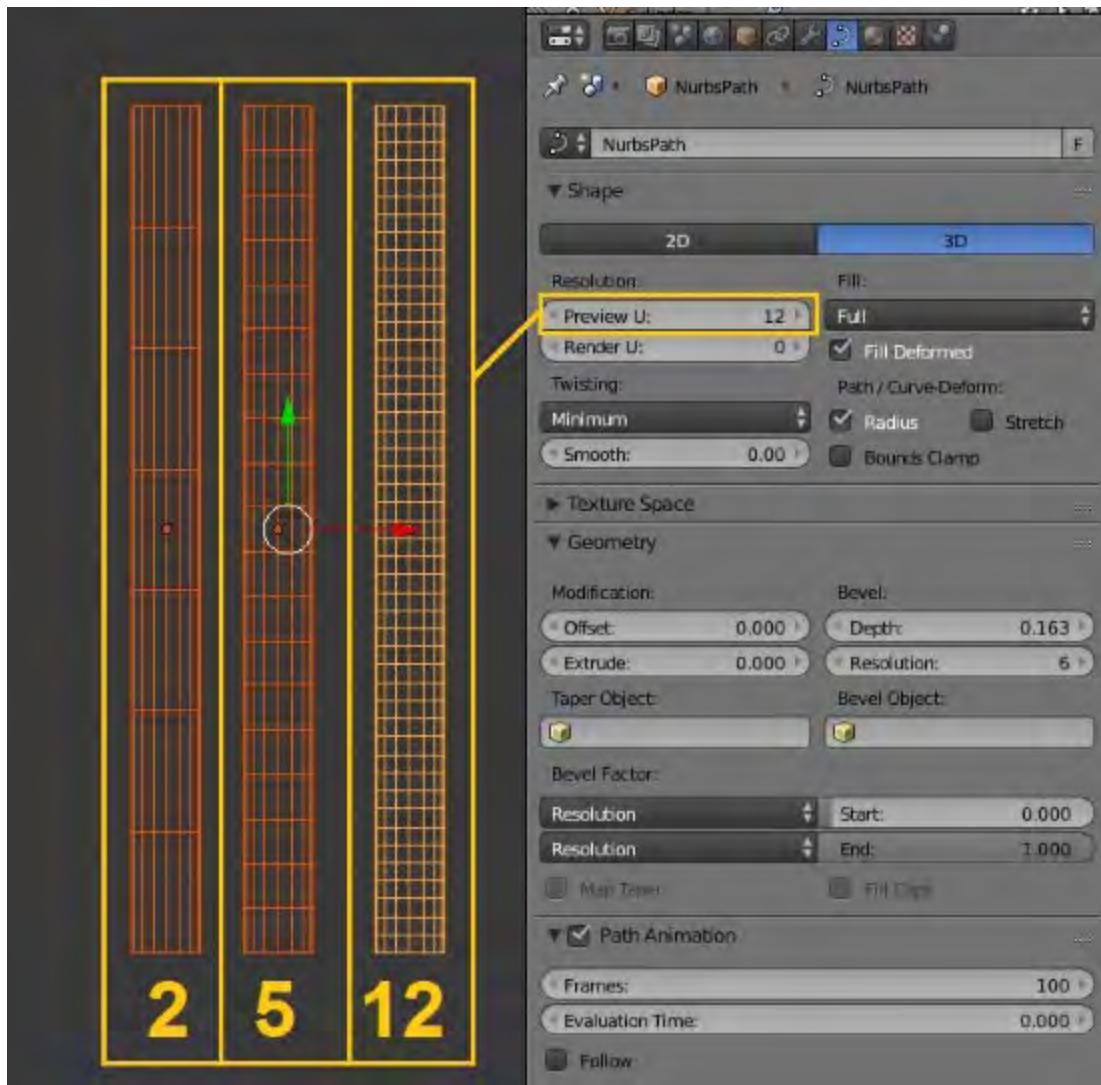
Working with Path objects

The last technique we'll cover in this chapter is using Path objects. These are ideal for making wires, cables, hoses, and more. To get started, just add one from the **Curve** section of your **Add** menu:



<pagebreak></pagebreak>

From the **Path** section of your **Properties** panel, you can see that we have a number of options that we need to look at. The **Preview Resolution** shows you how many segments your **Path** will be broken down into:



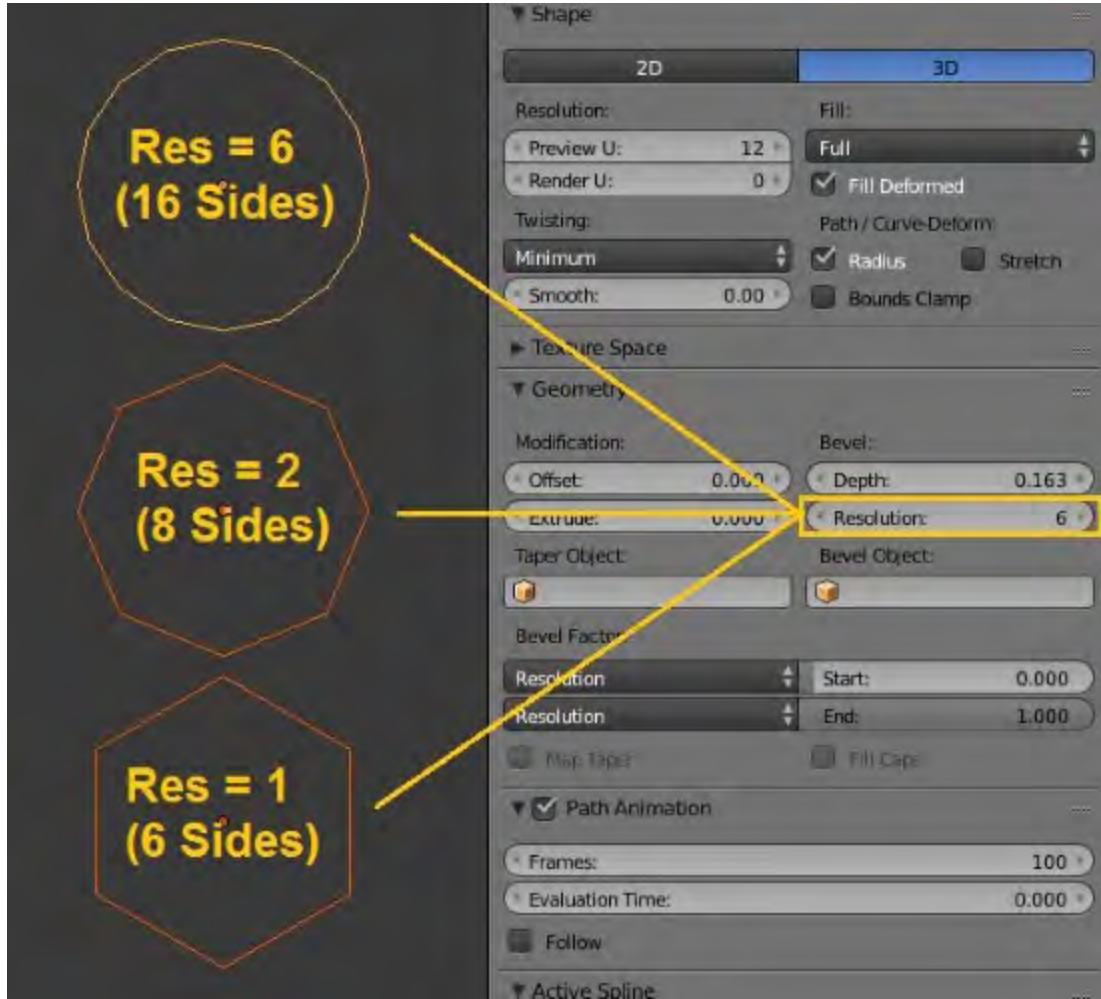
This is very important. If your **Path** is going to have a lot of curves to it, you'll need a decent resolution to make it look smooth. On the other hand, it should be obvious that the higher the resolution, the more polygons you'll eventually be using.

Also, don't be fooled by the concept of **Preview Resolution**. This is (or will be) your actual resolution. The **Preview** term just means that if you were to leave it as a **Path** object and render it, you could choose a different resolution for render time than in the 3D viewport.

Eventually, we'll want to turn our **Paths** into real mesh objects. It means that the **Preview Resolution** will be reflective of our final model.

<pagebreak></pagebreak>

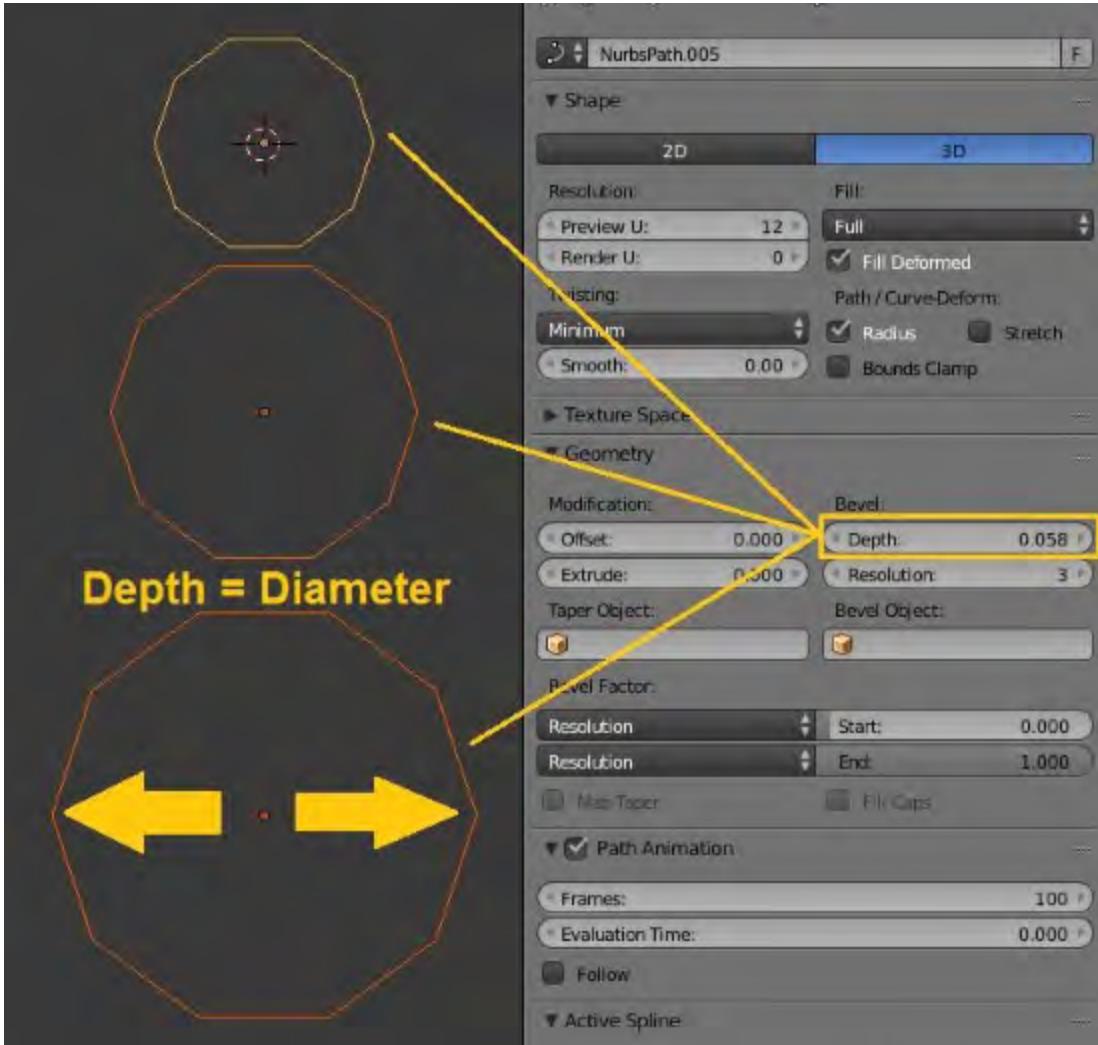
Under the **Geometry** section, there's another box labeled **Resolution**. This is important, too, but it's different. This value specifies the number of sides to your Path object:



How many sides should your **Path** have? Well, it's really a question of how many you can afford, in terms of polygon count. If you're going to put in one or two **Path** objects, you can probably turn the resolution way up. But if you're going to be making a whole lot of them (like I am), you probably need to turn it down a bit. The great thing is that you can adjust this setting at any time. So, once you've added all of your **Path** objects, you can come back and play with these settings until you find a balance between detail level and polygon count.

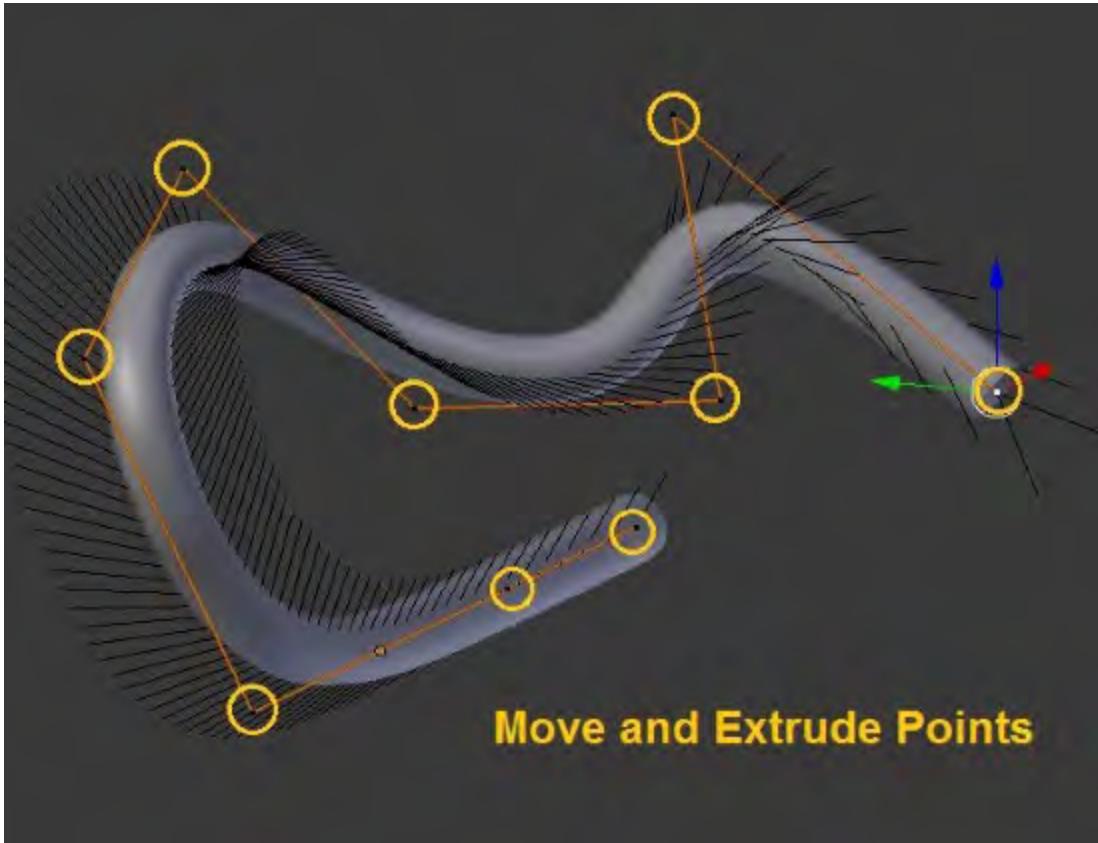
<pagebreak></pagebreak>

The next setting we'll want to take a look at is **Depth**. This is the diameter of your **Path**:

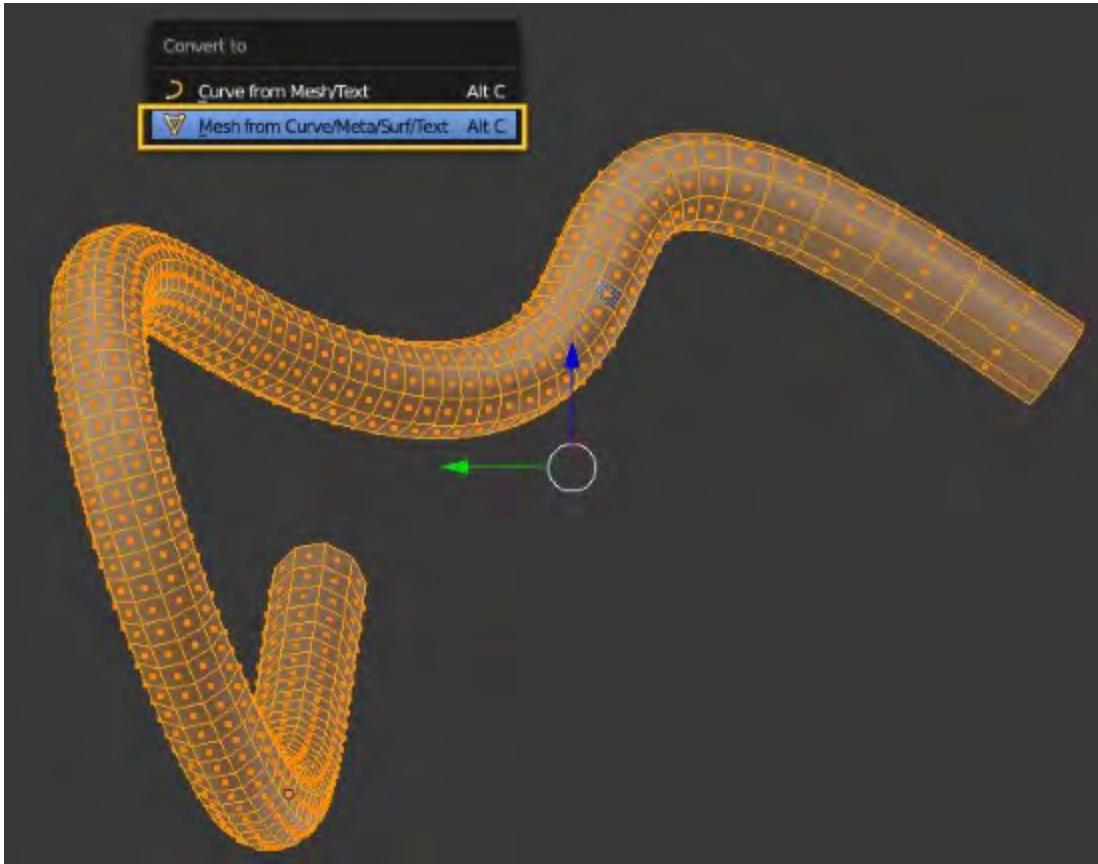


<pagebreak></pagebreak>

Once you've got those settings the way you want them, you can start to work on your **Path**. In **Edit Mode**, you can move or extrude the individual points that make up the **Path**. These aren't vertices, they're just temporary points that let you control the shape:



Once you have your **Path** the way you want it, you can use Alt + C, **Mesh from Curve/Meta/Surf/Text**. This will make your **Path** a regular mesh object. Be careful, though—don't do this until you're sure it looks the way you want it to.



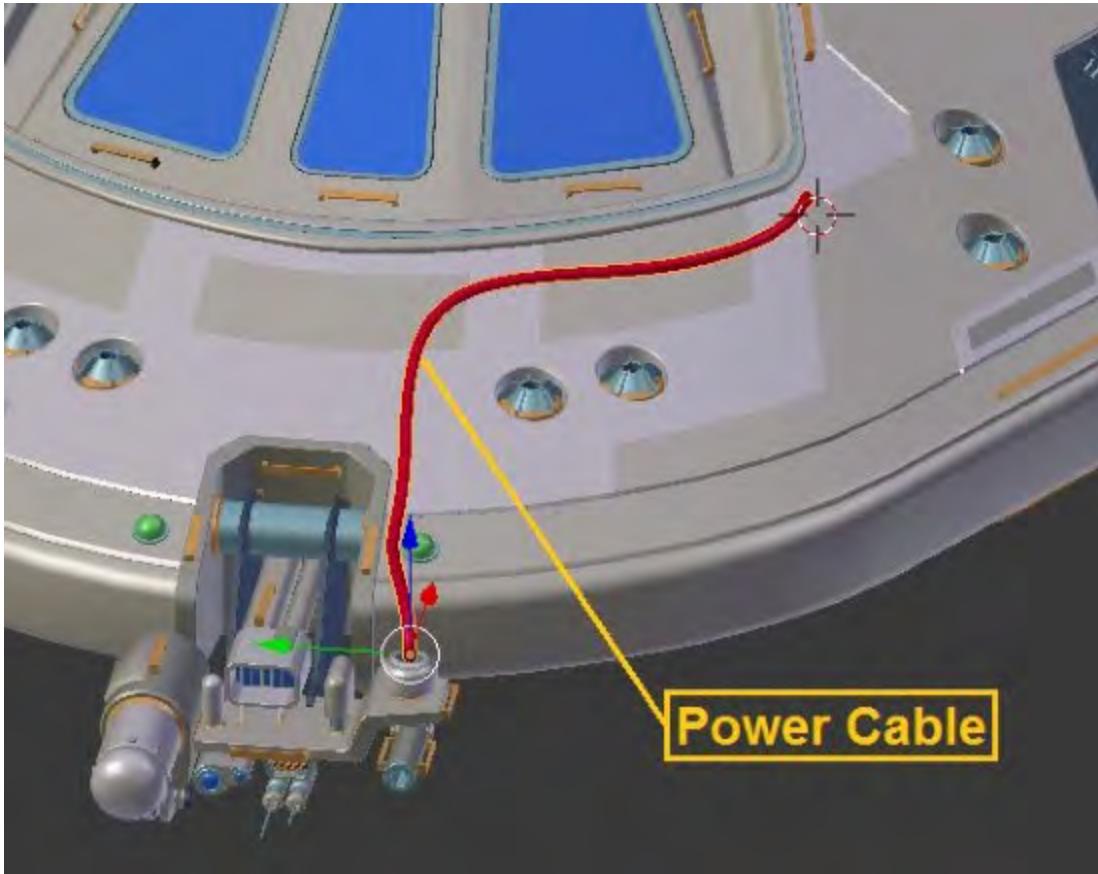
<pagebreak></pagebreak>

Maybe you're wondering why we need to convert at all. Can't we just leave these as **Paths**?

Sure, you can. There are a couple of downsides, though. For example, you can't do any UV unwrapping on a **Path** (we haven't covered this yet, but it's how you assign an image texture).

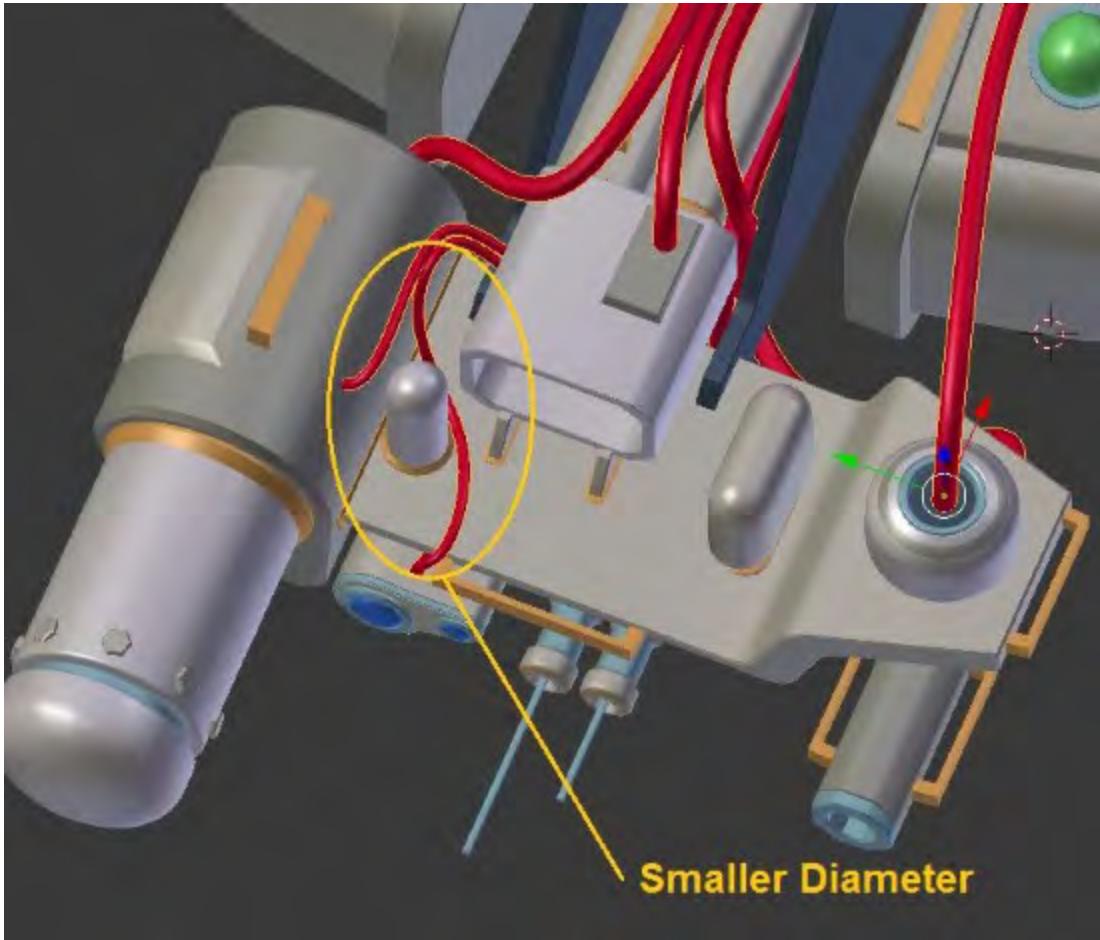
Also, you may want to make some small modeling changes to your **Paths**. For instance, you can go in and delete unneeded loop cuts (on straight sections of the **Path**) to save geometry. So, it's really a personal choice. If you like, you can make a copy of your **Paths** before you convert them. That way, you can always go back and modify their characteristics later.

Here's an example of a **Path** object that I've added to the front of the ship to make a power cable:



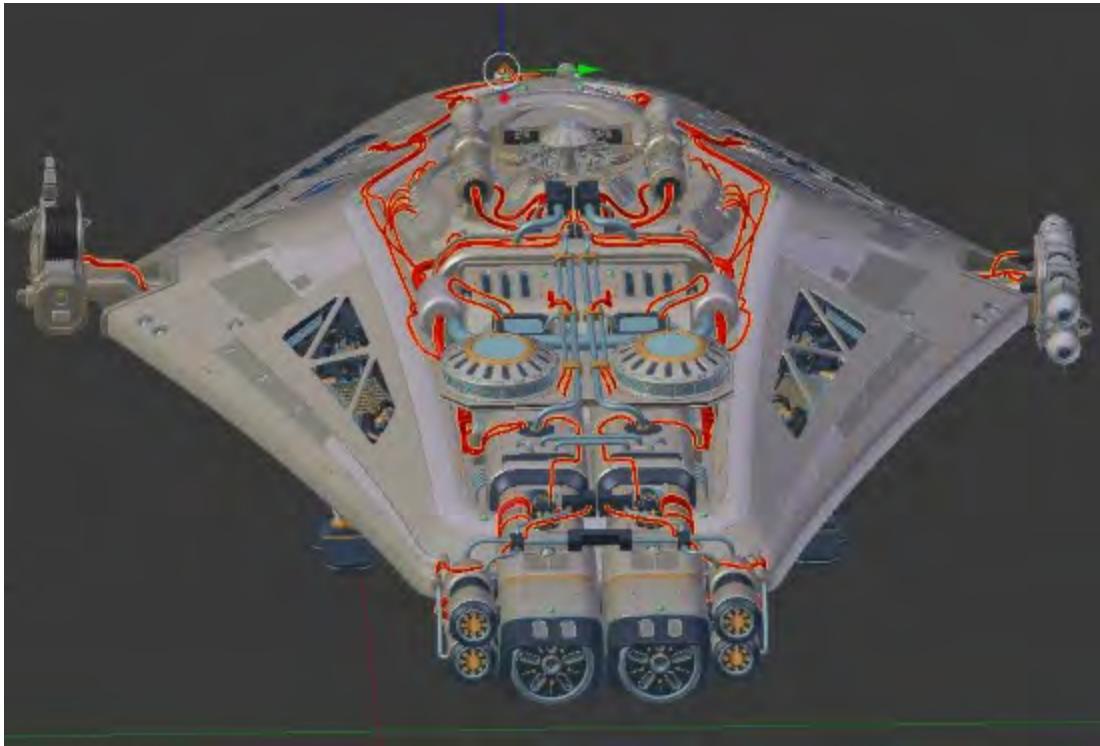
<pagebreak></pagebreak>

By the way, you don't need to add multiple **Path** objects to your scene (generally). Within **Edit Mode**, you can copy an existing **Path** and just move it to a new position. That way, any changes you make to its characteristics (diameter, resolution, and so on) will apply to all of your Path objects. If you want to change the diameter of just one or two sections, you can use Alt + S to adjust their diameter in **Edit Mode**:



<pagebreak></pagebreak>

Here's what I ended up doing with my **P a th** objects:



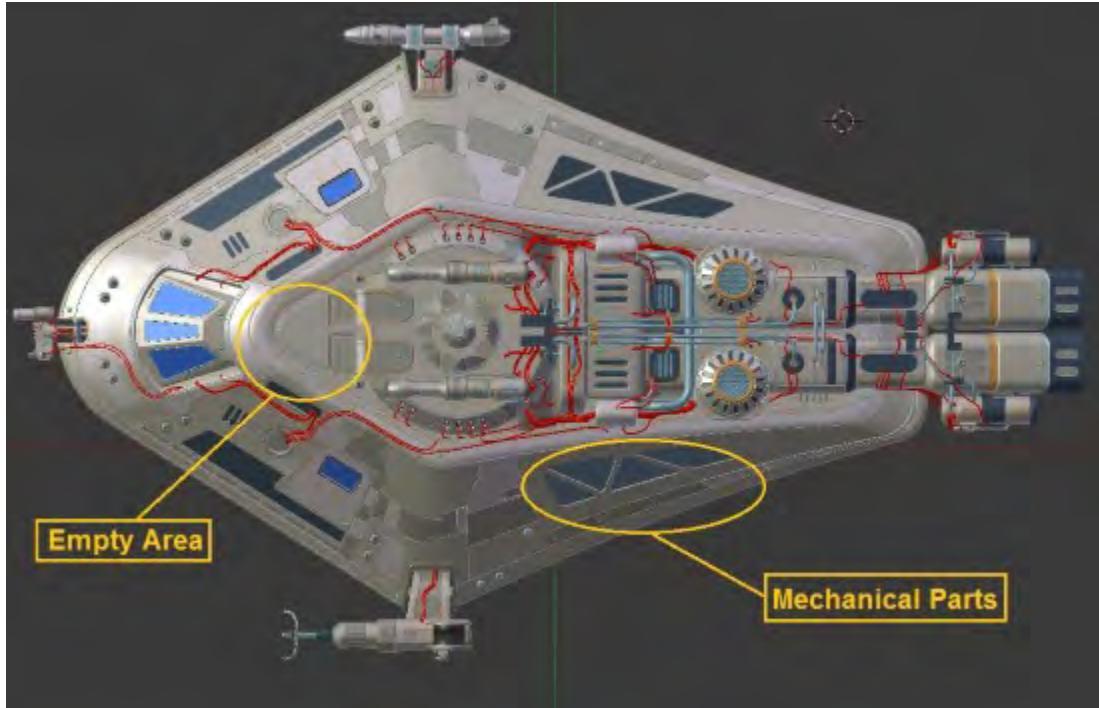
You may think that this is too many **Paths**. After all, it does appear a bit cluttered and messy. For obvious reasons, engineers tend not to design things with random cables and wires all over the place. If they're exposed, they tend to be tied down in fixed cable guides or rails.

On the other hand, maybe you want your ship to be a little messy. Maybe it's not a brand new vessel straight from the factory, but something that's been modified and haphazardly repaired over the years. Maybe a lot of that equipment wasn't in the original design, so it's been wired up over time in an unsafe manner. This is really an artistic choice, so I leave it to you.

<pagebreak></pagebreak>

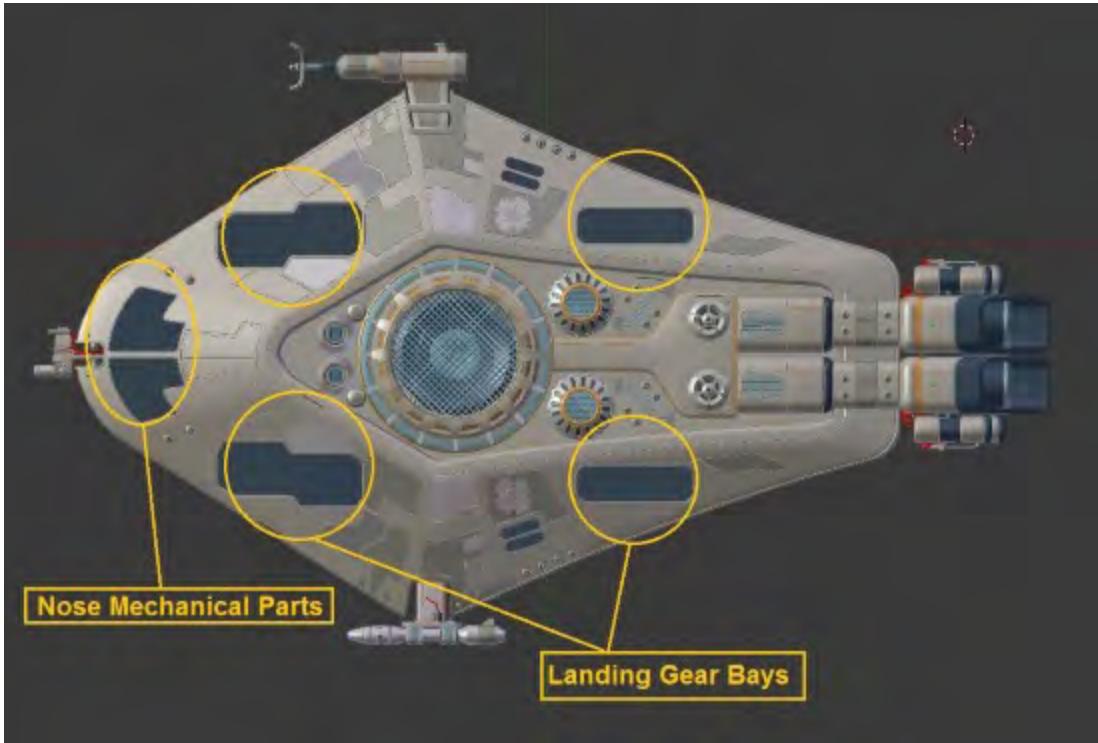
Finishing touches

At this point, you have just a few key areas left to finish on your own. On the top, we have the recessed mechanical area and the empty spot by the nose:



<pagebreak></pagebreak>

On the bottom, we have the rest of the landing gear bays and (in my case) the extra set of mechanical areas in the nose:



By this point, you should have no trouble completing those areas on your own. You can duplicate parts from other areas of the ship, or create completely new objects—it's up to you.

<pagebreak></pagebreak>

Here's what my final ship looks like (with no materials):

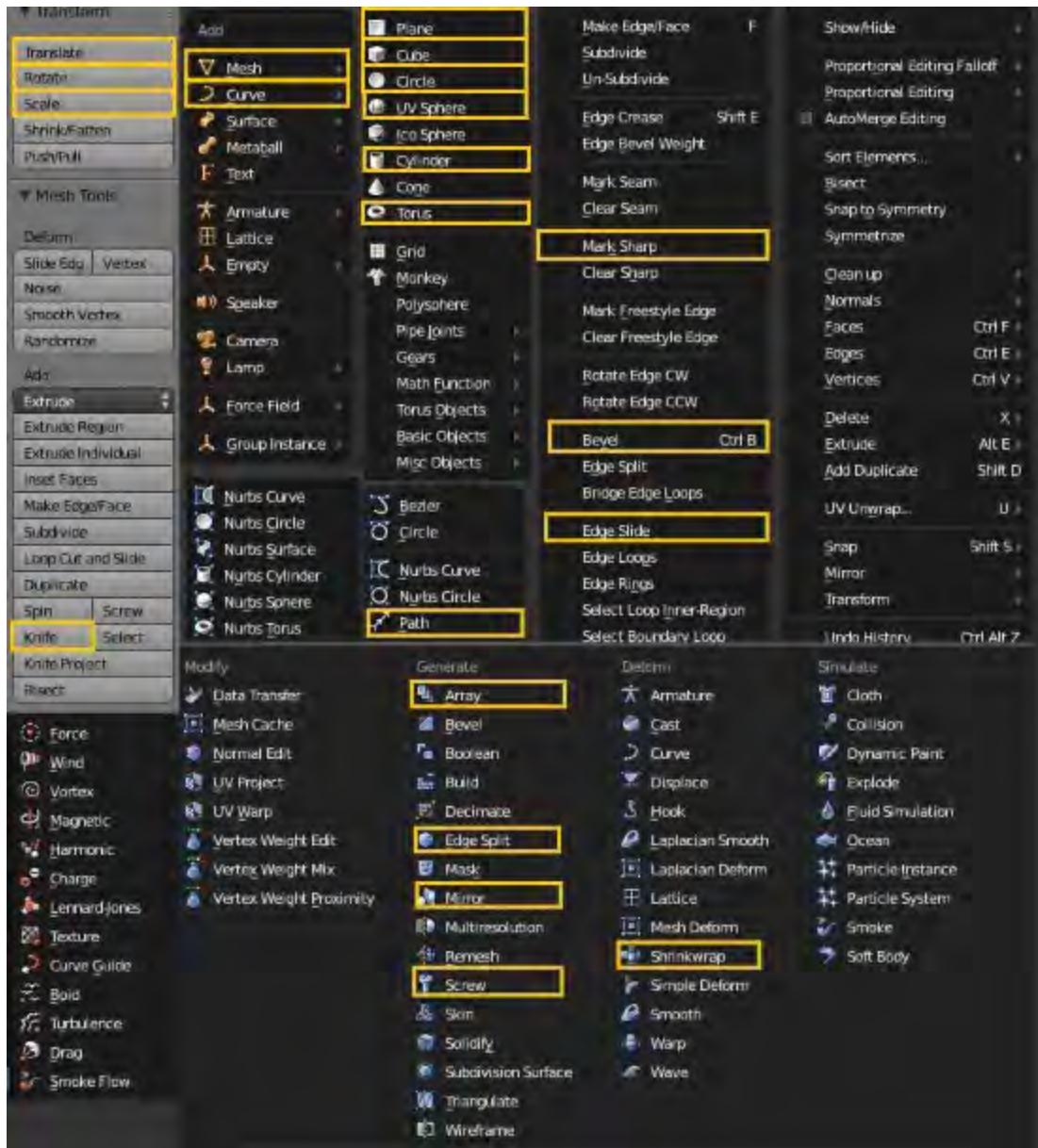


So that's it—we're finally done with our ship model! I know this was a very complex project, but it did let us practice a lot of key skills. By this point, you should be very comfortable with the techniques we've been using.

Before moving on, there's an important observation we can make.

<pagebreak></pagebreak>

One of the hardest parts of learning Blender is the sheer number of tools and options available. But consider this—we were able to build this entire (complex) spacecraft using only a handful of these menu items:



Everything in Blender has its purpose, but many options aren't related to hard surface (mechanical) modeling. After doing this last project, you can get a sense of what your key tools and techniques are. I recommend that you get comfortable with those first, then move on to others.

<pagebreak></pagebreak>

Summary

In this chapter, we finished modeling our spaceship. We used a few new tools within Blender, but mostly we focused on workflow and technique. Before you move on, take as much time as you'd like to get your ship just right. Once you're satisfied with the model, we'll bring it to life with materials, textures, and rendering in the next chapter!

Chapter 6. Spacecraft – Materials, Textures, and Rendering

In this chapter, we'll add materials, textures, and basic setup to our spacecraft. This will complete the project. The following are the topics covered in this chapter:

- Preparing the model and scene
- Creating materials
- Adding decals with UV maps
- Other possibilities

Now that we've finished our model (finally), it's time to add some materials and textures to bring it to life. There are many options for doing this, and we'll look at a number of different materials here.

These materials are fairly simple, but they lay the groundwork for creating much more complex materials on your own. Let's get started!

<pagebreak></pagebreak>

Preparing the model and scene

Before adding materials, let's prepare our model. First, make sure that path objects have been converted to meshes with Alt + C.

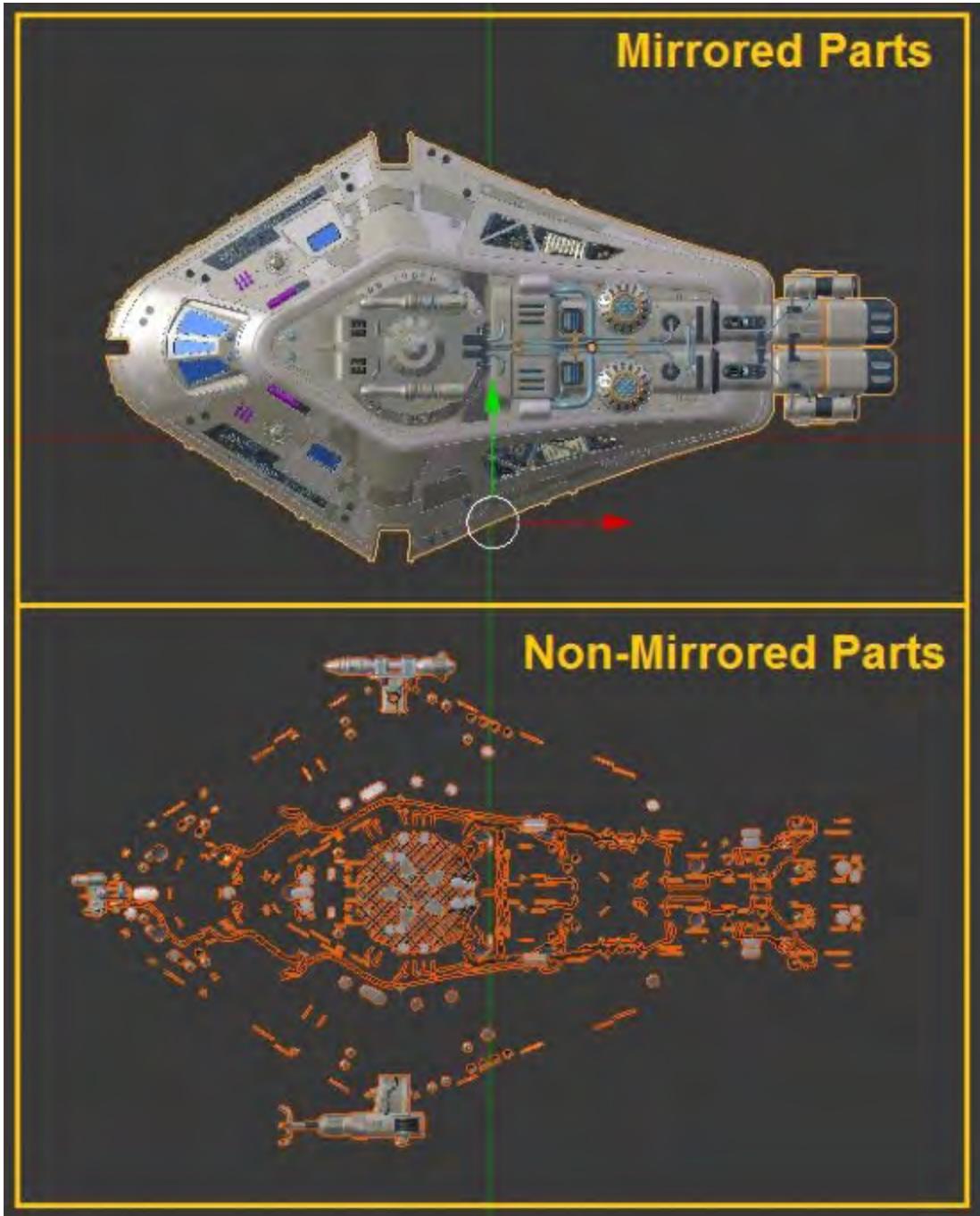


Then, we'll combine our various parts together. In the end, I'm just going to be left with two objects: mirrored parts and non-mirrored parts. This will make the texturing a bit easier.

<pagebreak></pagebreak>

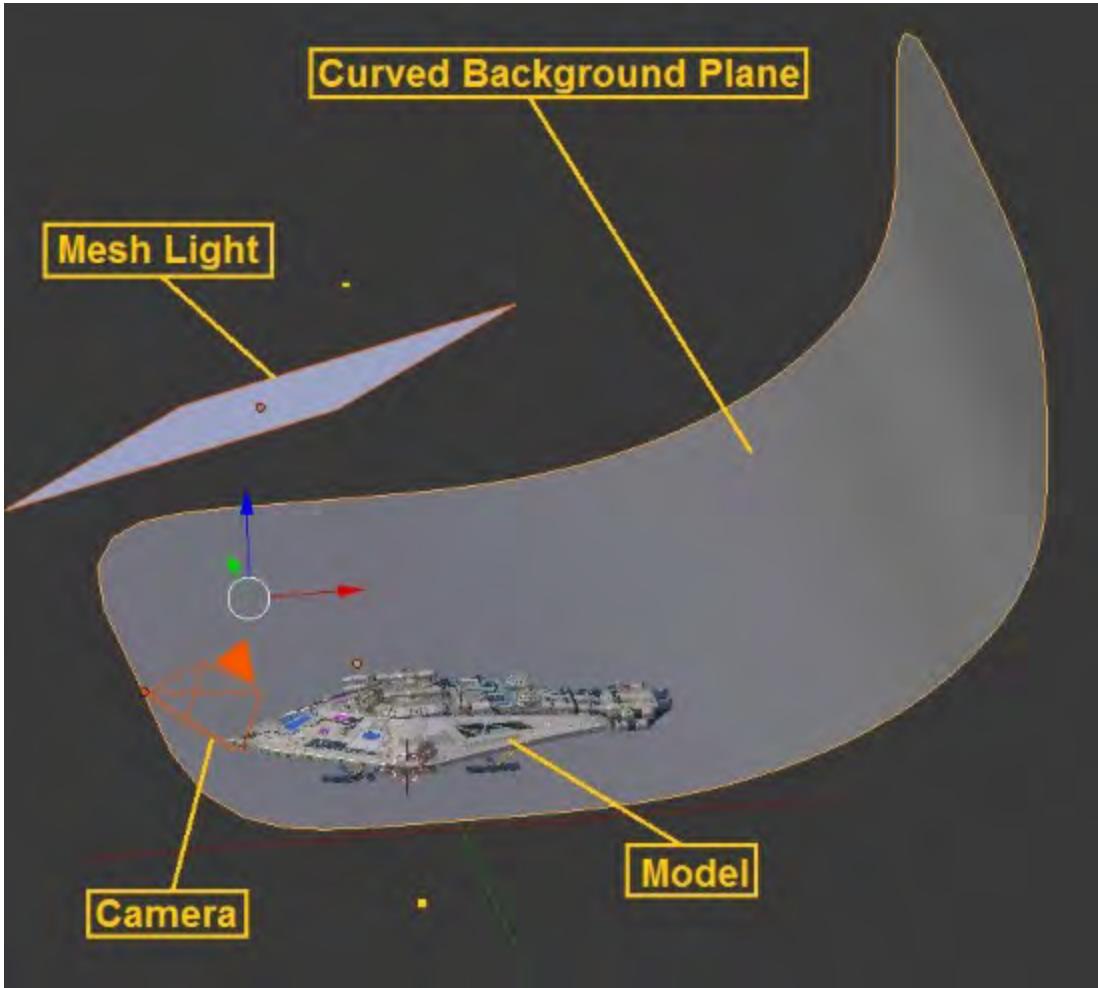
Note

In general, it's good practice to have the minimum number of separate objects as necessary.



<pagebreak></pagebreak>

Now, we'll add a basic scene to render our spacecraft. I've added a curved background here, along with a mesh lamp and a camera:



Obviously, this is a very simple scene. You could get far more complex with it, but this will work to show off the model.

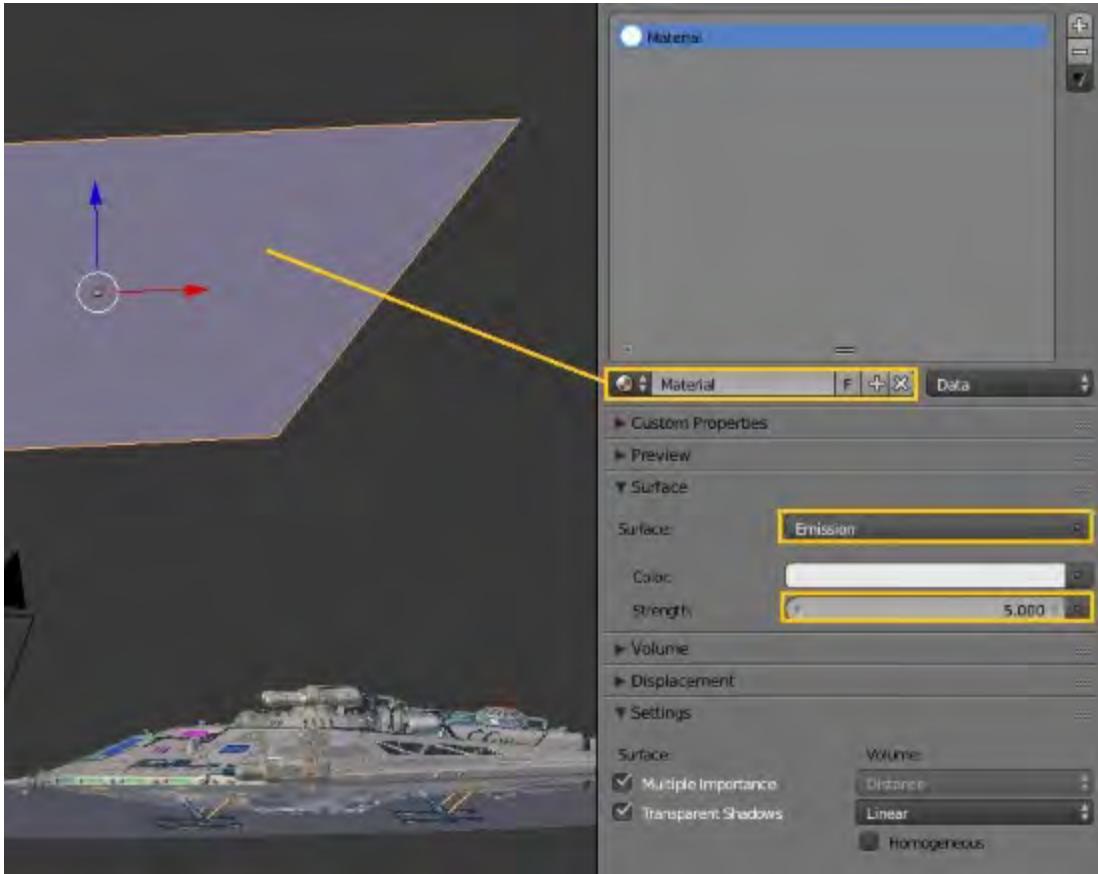
By pressing 0 on the number pad, you can now look through your camera.



Make sure that your background is large enough and that you see exactly what you want. You may need to adjust the position of various objects.

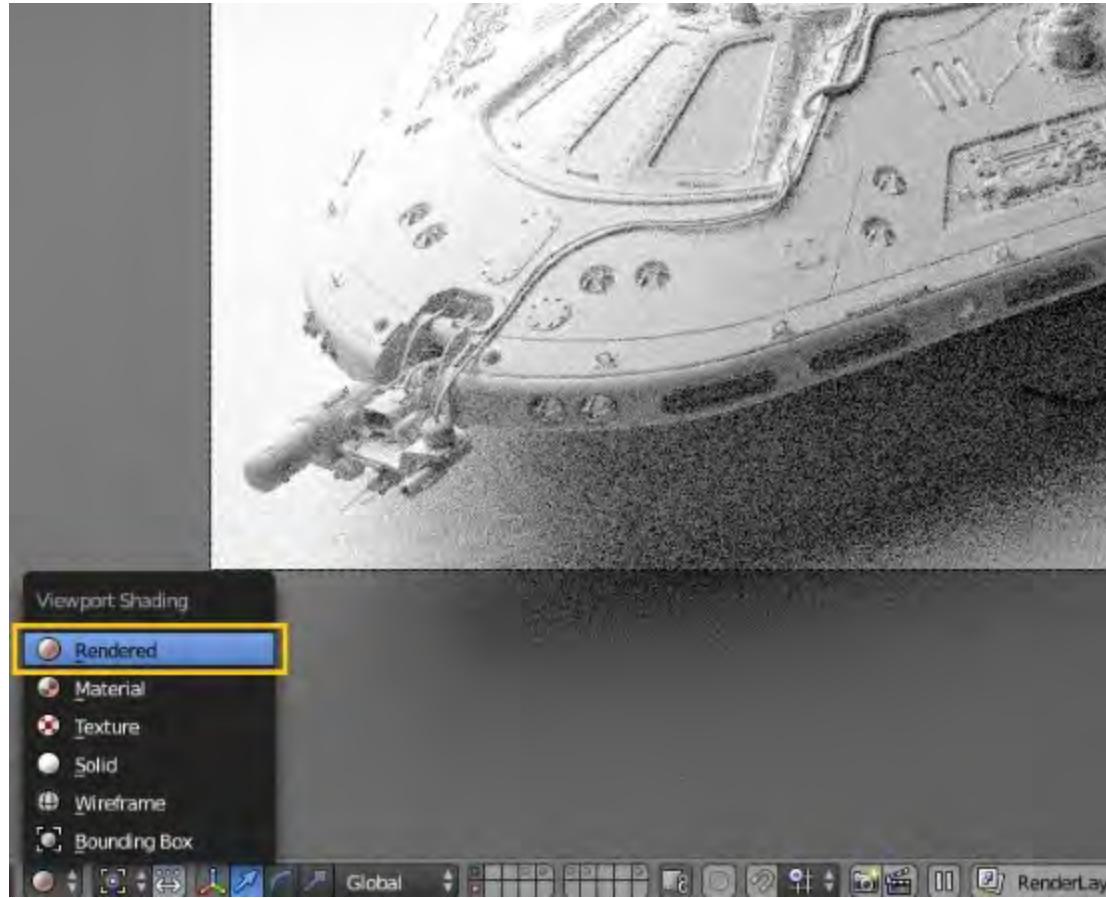
<pagebreak></pagebreak>

Now, we'll add an **Emission** material for our light:



<pagebreak></pagebreak>

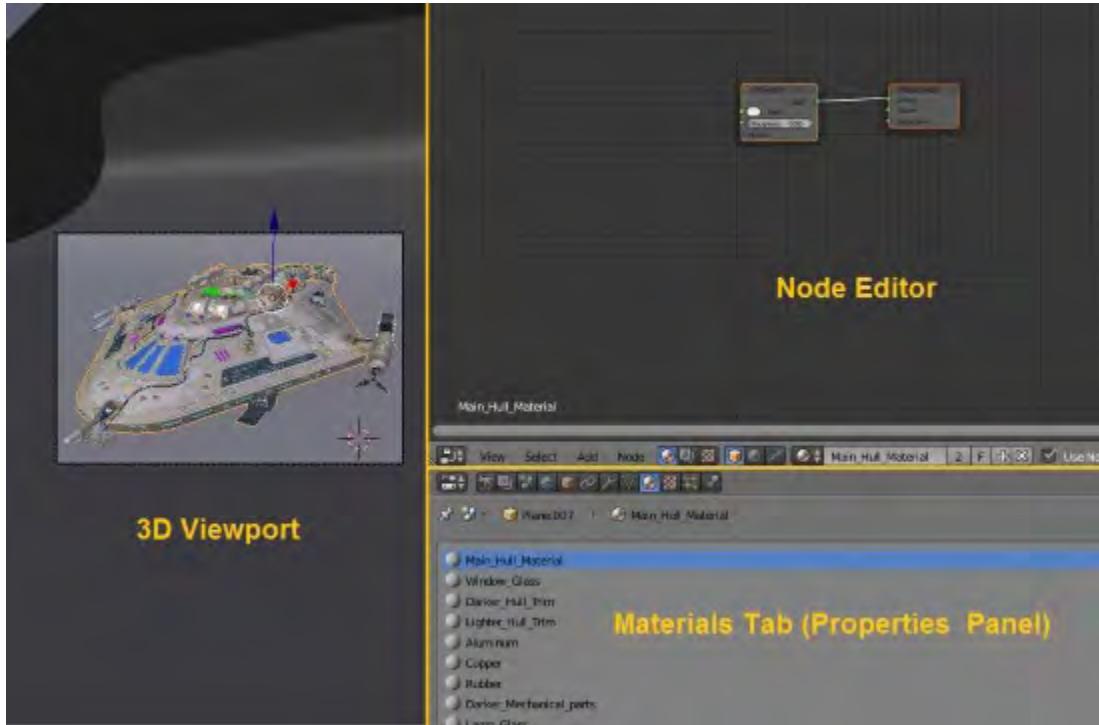
When we switch to the rendered view, we can see that there's now illumination in our scene:



<pagebreak></pagebreak>

Creating materials

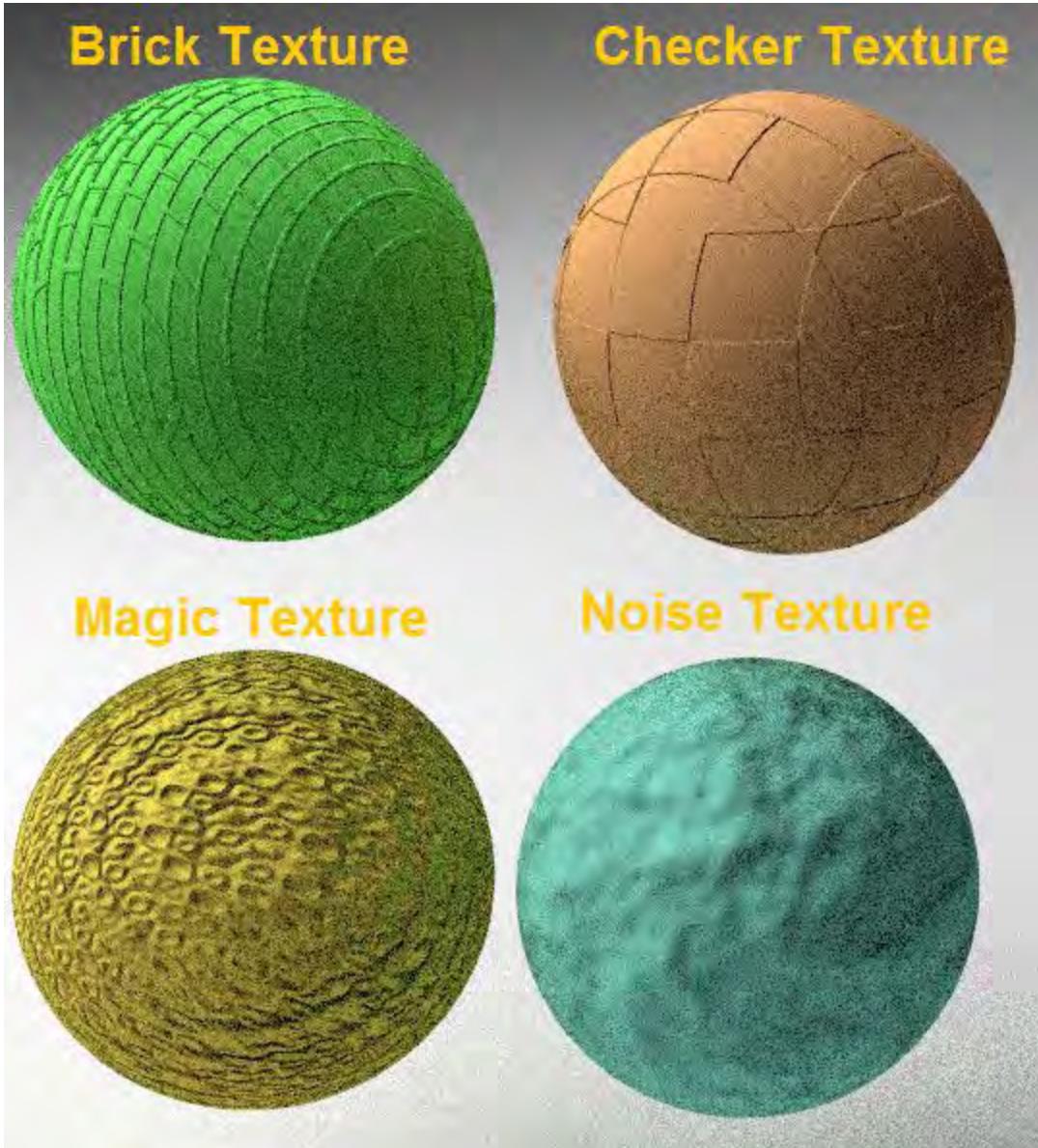
Now, it's time to set up our materials. The first thing I'll do is split my screen into three parts—the **3D Viewport**, **Node Editor**, and **Materials Tab**:



This is just a personal preference, but it does give you access to everything you'll need in order to adjust your materials.

<pagebreak></pagebreak>

Before going further, let's take a look at some of the basic procedural textures that come with Blender:



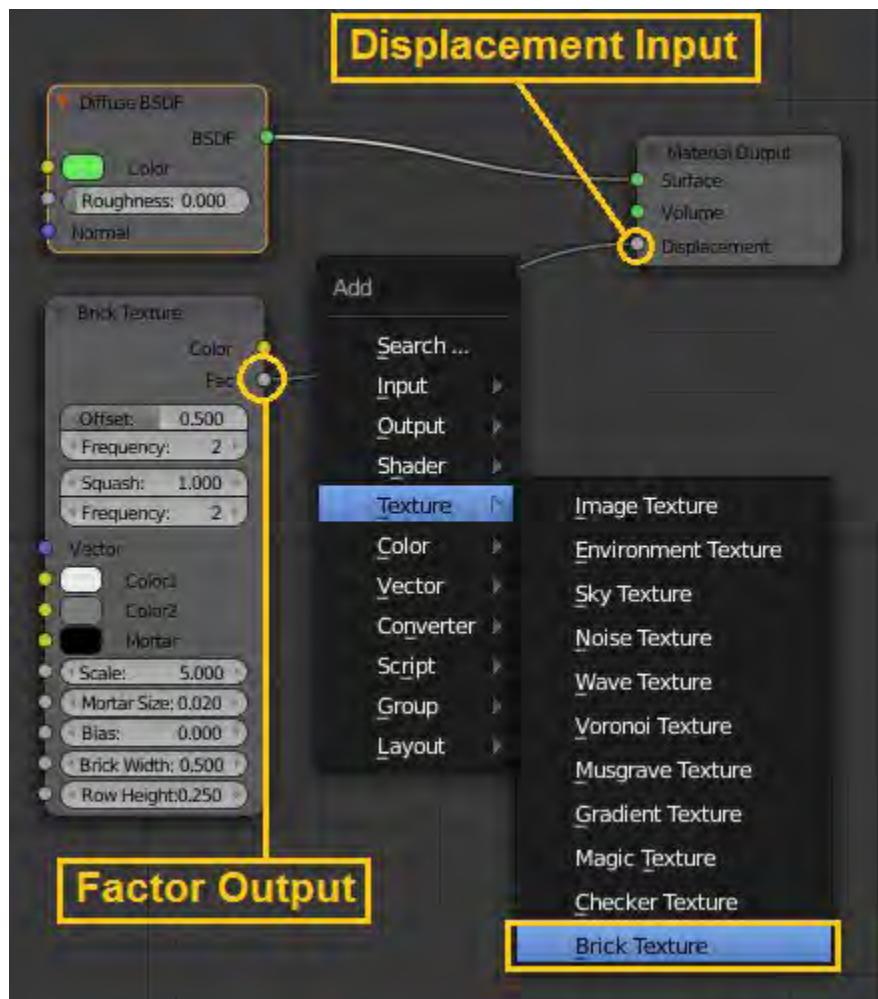
As you can see, I've used these textures to affect the displacement or normal of the material. The textures are basically acting as **normal maps**.

Note

A normal map is a way to add additional detail to the model without physically building it. For example, if you wanted a cloth pattern, it would not be practical to model a cloth with all of the tiny folds and stitching. Normal mapping allows us to simulate the light and shadows of a complex surface without adding all of the geometry to model it.

<pagebreak></pagebreak>

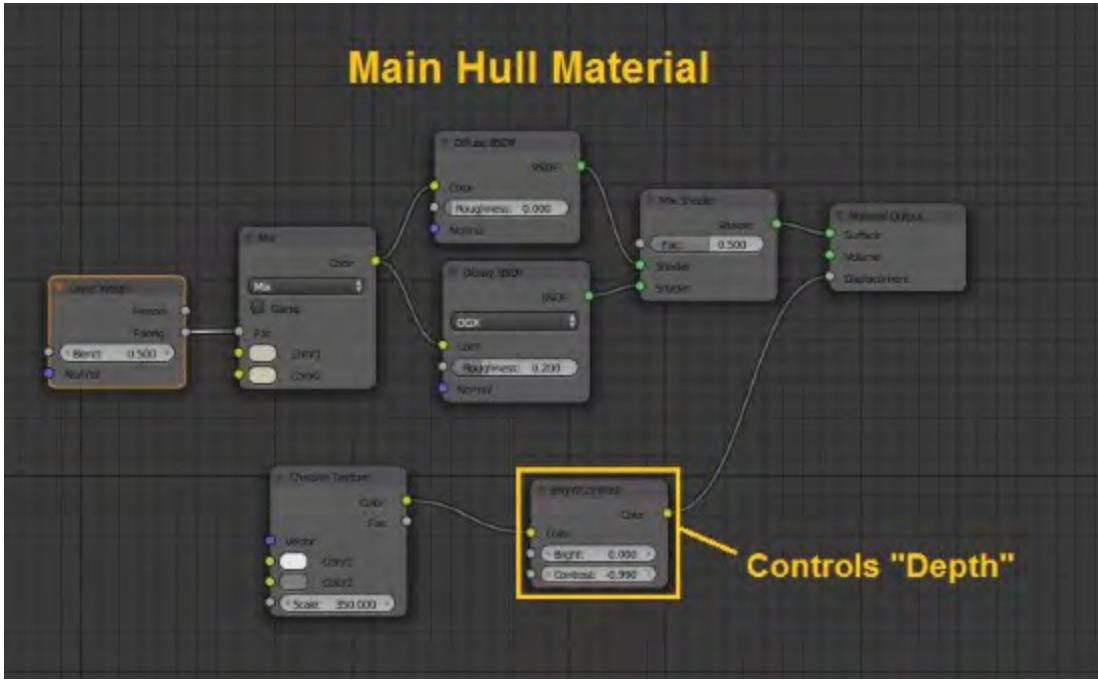
You can use a texture to control the normal map by plugging its output into the **Displacement** input of the material node:



For our main hull material, I'd like to create a canvas or thermal blanket. We already did a metal material for our gun, so let's do something a little different this time. I'm going to set up some basic colors here, similar to what we've already done.

<pagebreak></pagebreak>

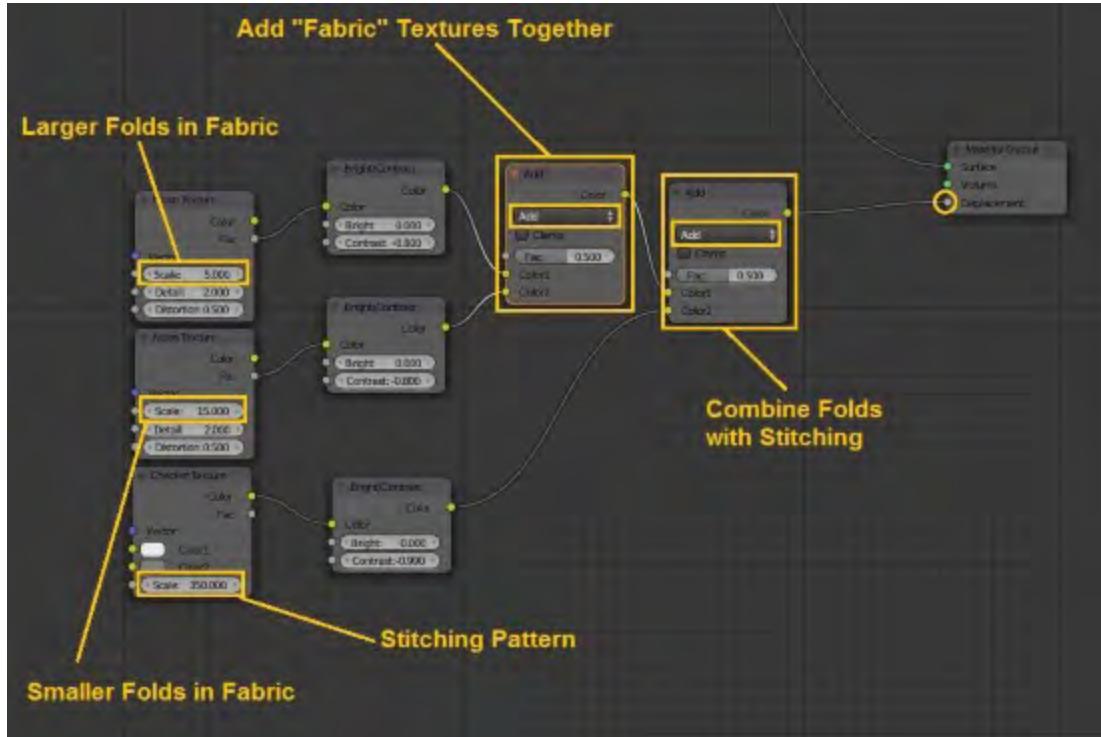
I'm also going to use a **Checker texture** to create the appearance of small stitching in the fabric. The **Bright/Contrast** node allows us to control the apparent depth of the fabric.



You can also combine a number of different textures for a more complex normal map. Here, I've combined a couple of **Noise textures** with the **Checker texture** to get the overall normal or displacement that I want.

Note

Technically, this is a "bump map" rather than a normal map. They will generally have the same effect, but there is an additional node called normal map that includes additional options for texturing. We don't need it here, but feel free to explore it on your own!



<pagebreak></pagebreak>

You can see what that looks like by either rendering your model or switching to the rendered view in the viewport:

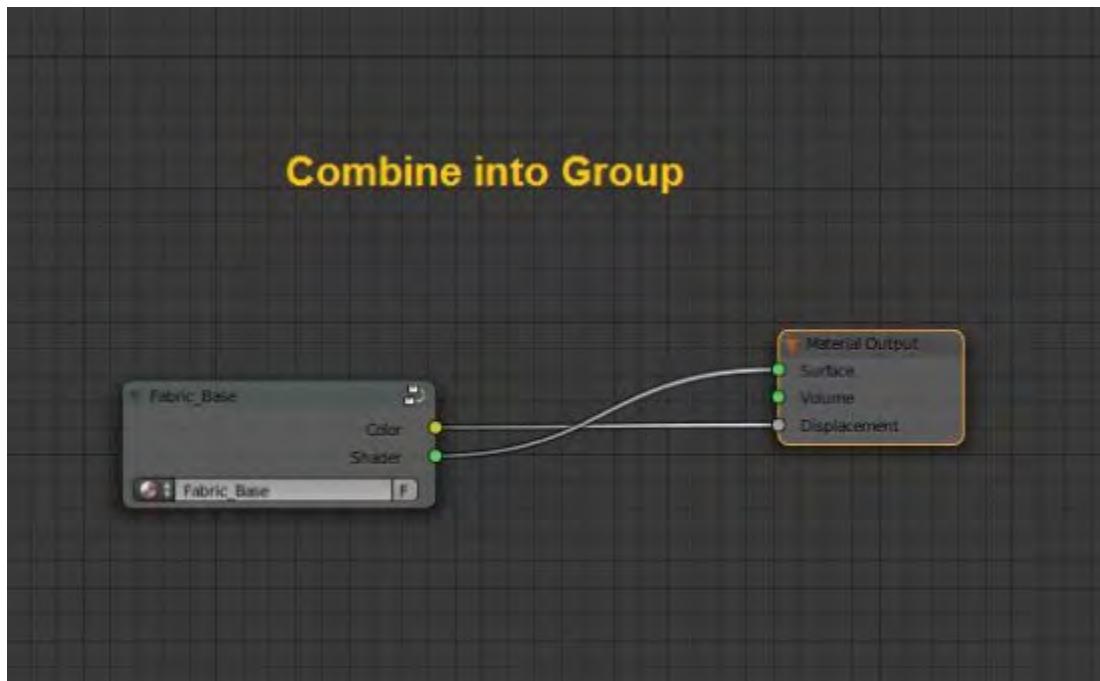


Once you're satisfied with the material, we'll need to copy it. Other parts of the hull will use similar materials, so there's no sense in recreating all of those nodes from scratch.

A quick way to do that is to just combine all of your nodes into a **node group**. This group can be plugged into multiple materials. It's also a good way to "copy and paste" between materials.

<pagebreak></pagebreak>

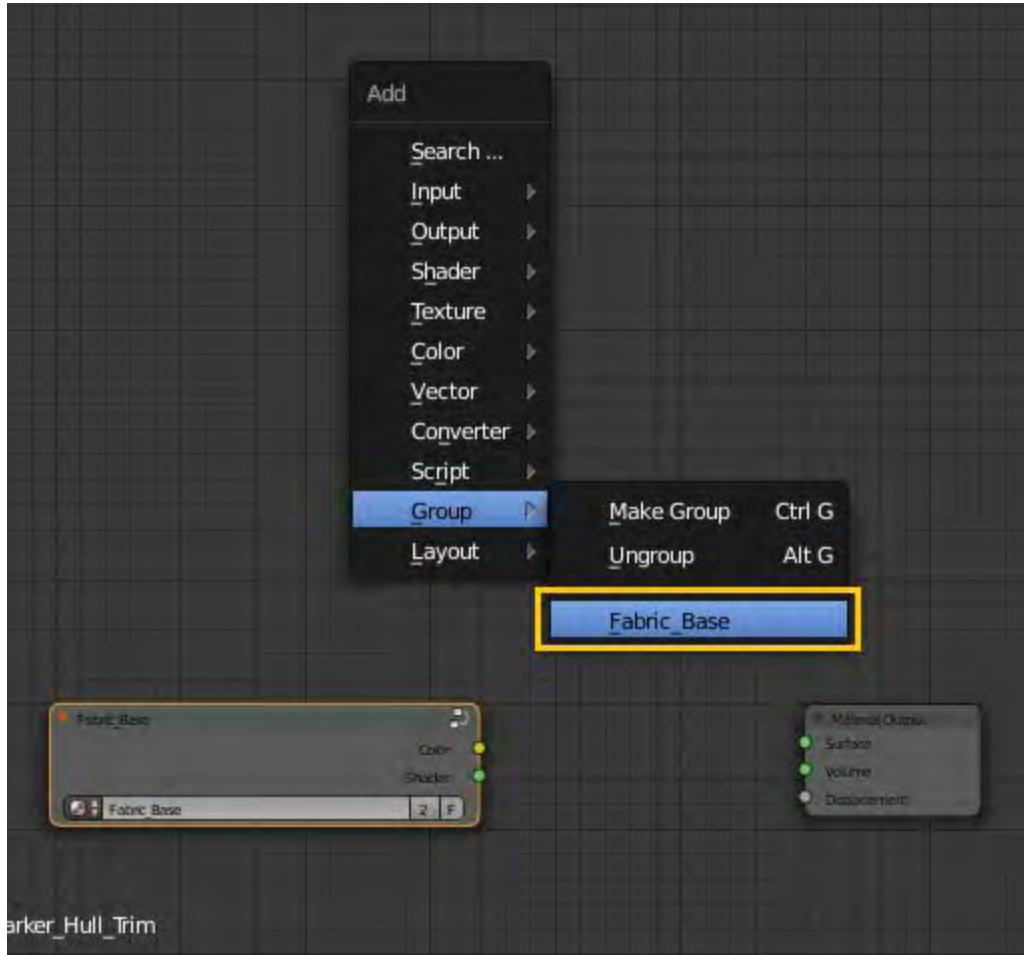
To add all your nodes to a group, simply select all of them and hit Ctrl + G. This will create the group. You can then use the Tab key to collapse the group, and all of your nodes will be hidden within the displayed node group.



I've named this group **Fabric_Base**, but it doesn't matter what we call it. We're only using the group temporarily.

<pagebreak></pagebreak>

Now, I can go into another fabric material and add that node group:



Using Alt + G, we can ungroup the nodes. We've basically copied everything from our first material. Now, you can go in and make a few minor changes, so the material looks different from the first one.

<pagebreak></pagebreak>

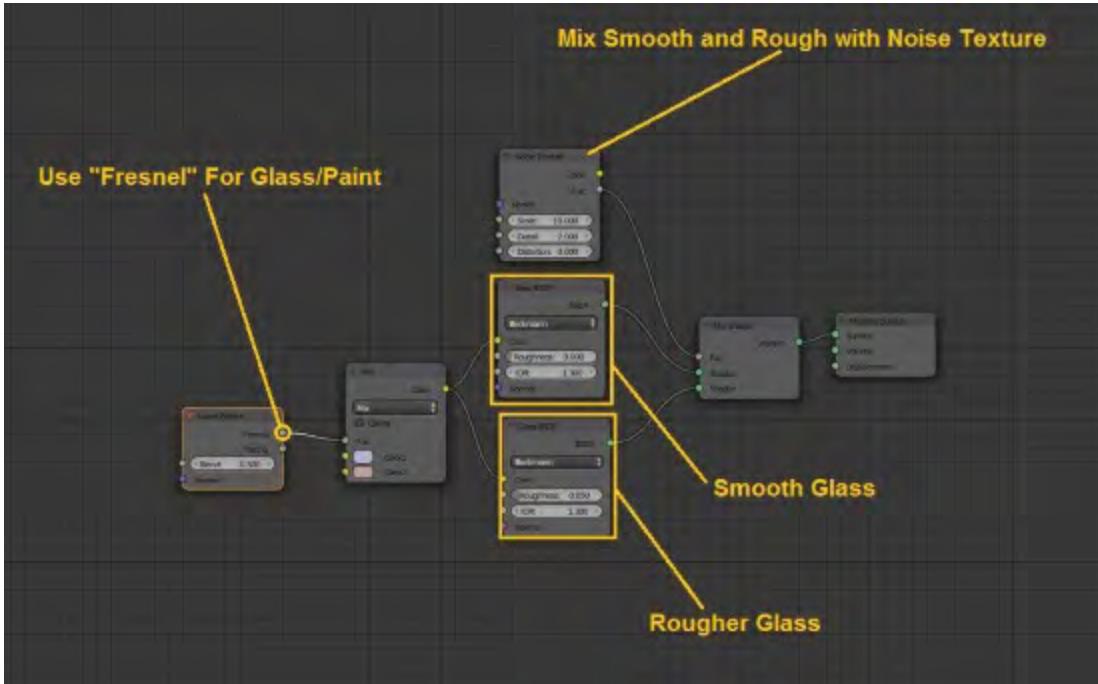
Repeat this process for our various fabric materials:



The next thing we'll tackle is the window glass. For this, I'm using a combination of two glass shaders. One is a bit rougher than the other, and I'm combining them with a texture, to give the appearance of glass that's not perfectly consistent.

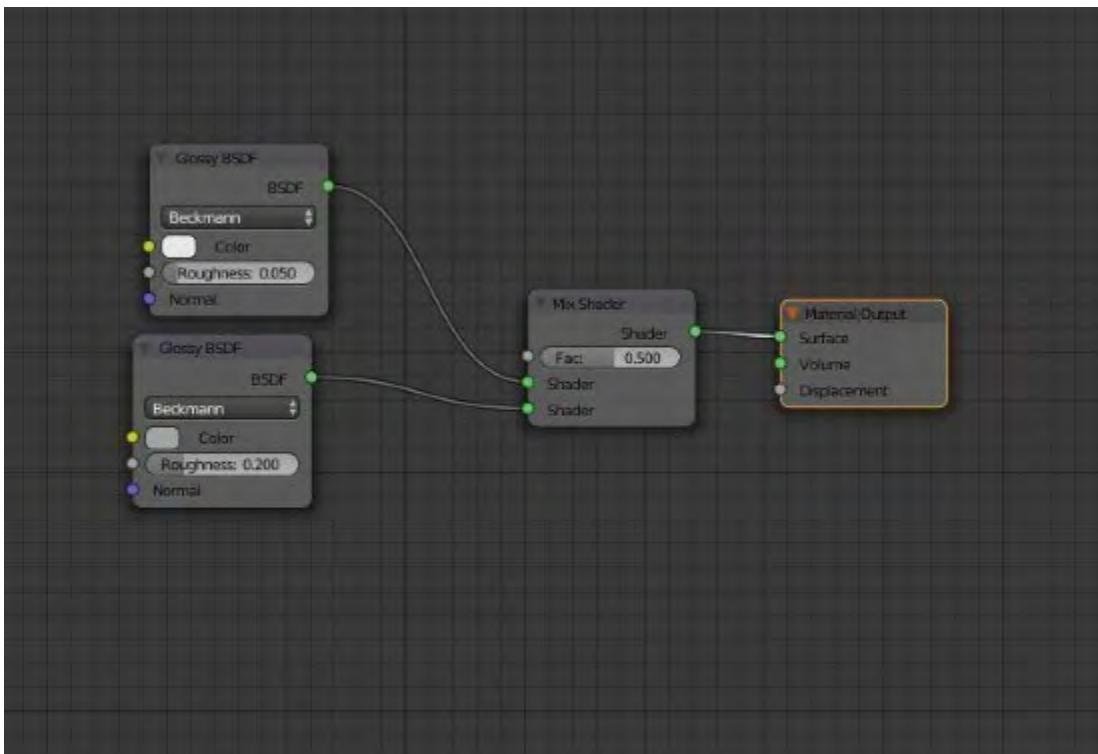
<pagebreak></pagebreak>

For my color input, I'm going to use another **Layer Weight** node. Typically, we'd use the facing output to control the color with glass, paint, and similar materials; however, sometimes it's better to use the **Fresnel** output. Fresnel replicates the way light reflects off extremely shiny materials or materials that have multiple transparent (or translucent) layers.



<pagebreak></pagebreak>

For my aluminium material, I'm going to use a simple combination of two different glossy shaders:



It's easy to get carried away with this. I've seen metal materials before that have over 100 different nodes. Entire books have been written on creating materials in Blender, as it's a huge and multifaceted topic. Generally speaking, the more work you put into a material, the better it will look.

<pagebreak></pagebreak>

However, this is a point of diminishing returns. More complexity isn't always better. The simple setup we're using will work a large portion of the time. As always, of course, you're free to take this further.

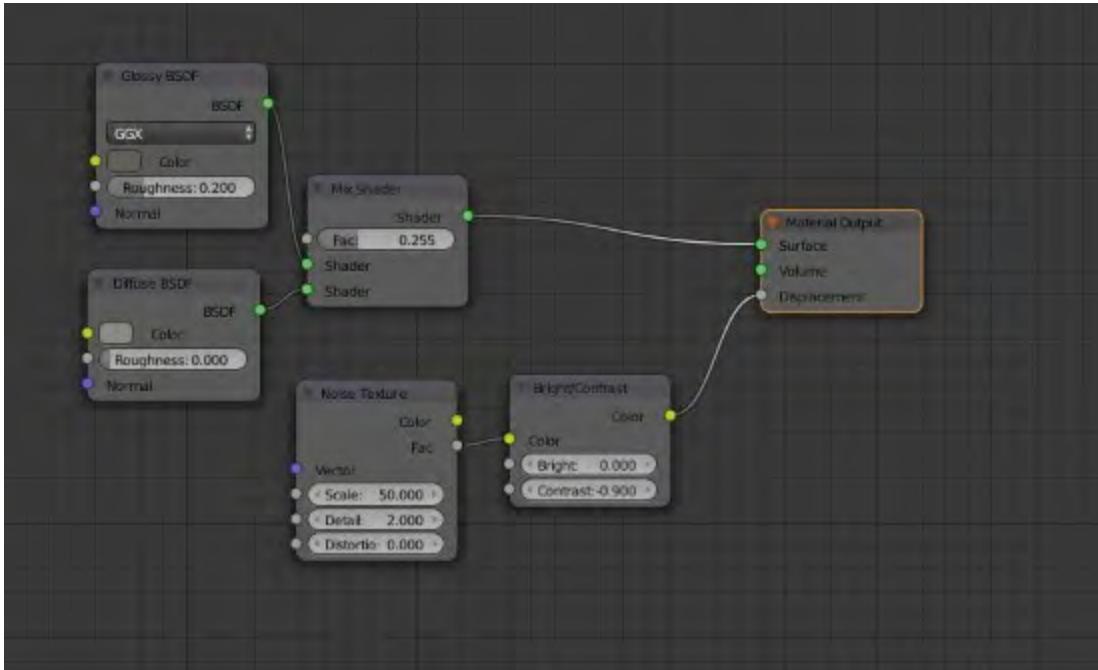
For our copper material, we'll just combine a couple of colors into a glossy shader:



For our rubber material, we'll just combine a couple of darker colors. Note that we're using the GCX setting for the glossy shader-this is better suited to plastic or similar materials that aren't as reflective as metal.

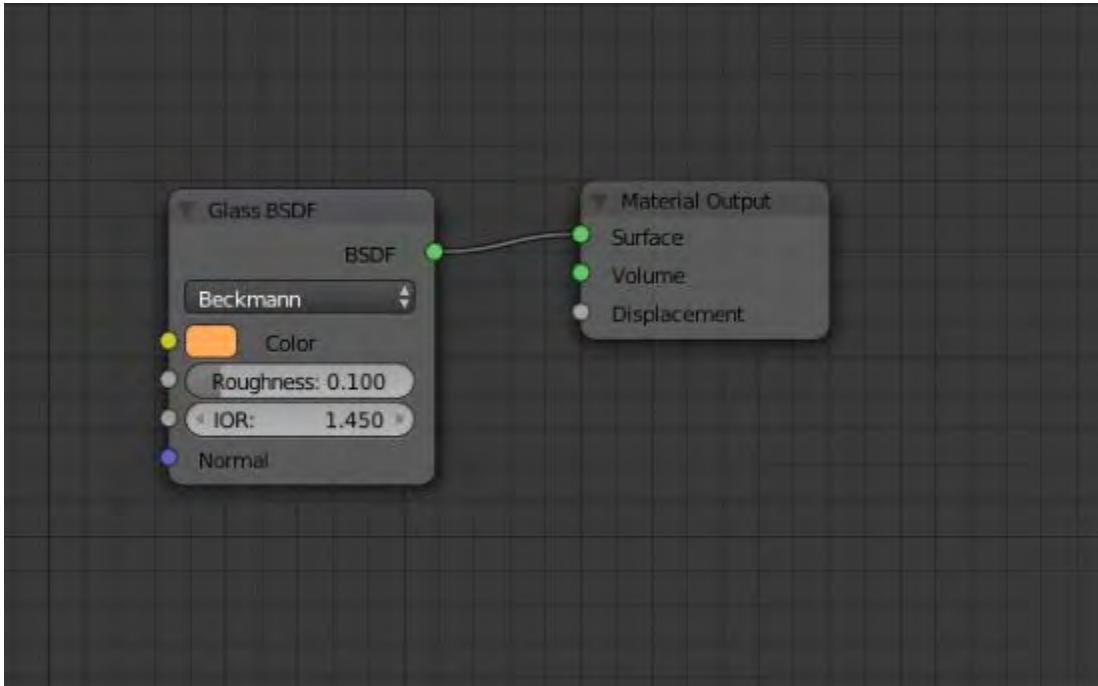
<pagebreak></pagebreak>

We'll also add a slight noise texture to give us a little distortion in the surface of the rubber:



<pagebreak></pagebreak>

For our running/navigation lights, we'll add a colored glass material. I'd like mention that the **Roughness** value is misleading. I've set mine to just **0.100** here, which doesn't seem like a lot. In reality, that makes the glass pretty rough. A setting of 0 will make the glass perfectly smooth, and a setting of 1 will make it perfectly opaque. Adjust the roughness until you're happy with it.



Those are the basic types of materials we'll be using on the spacecraft. You can add more of course, or change the ones we already have.

<pagebreak></pagebreak>

Adding decals with UV maps

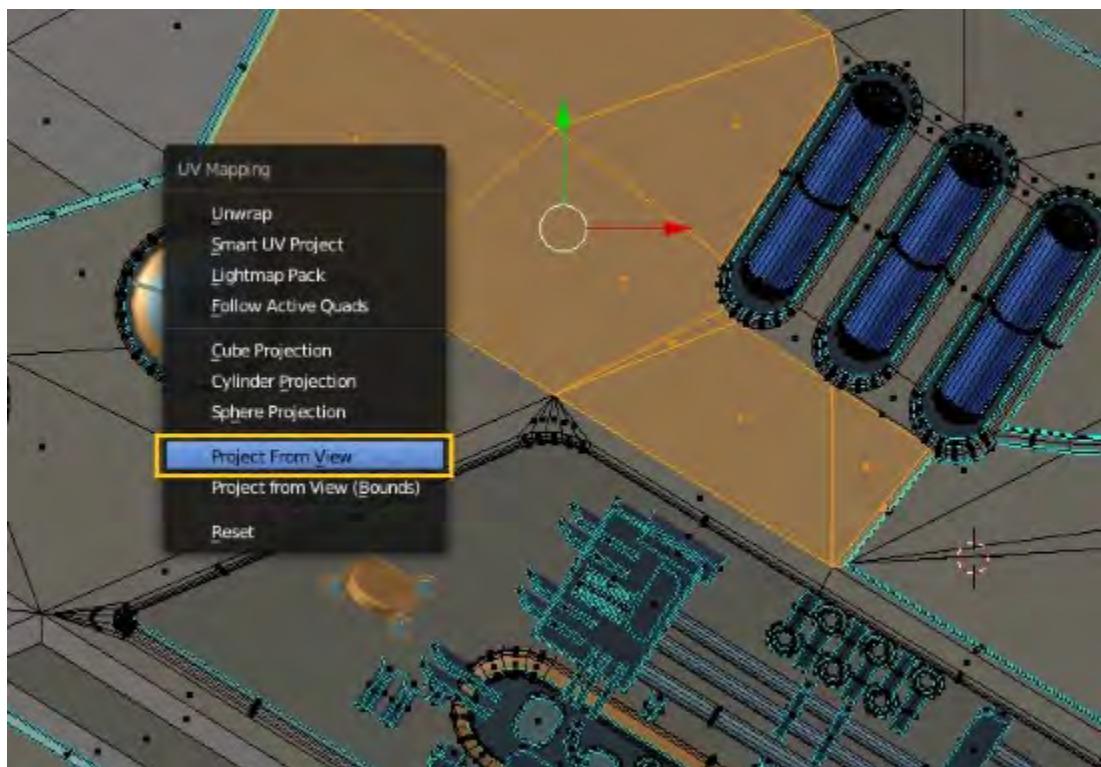
Next, let's add a few decals. Most large vehicles tend to have words, symbols, or logos on them. We can do this easily using **UV maps**.

Note

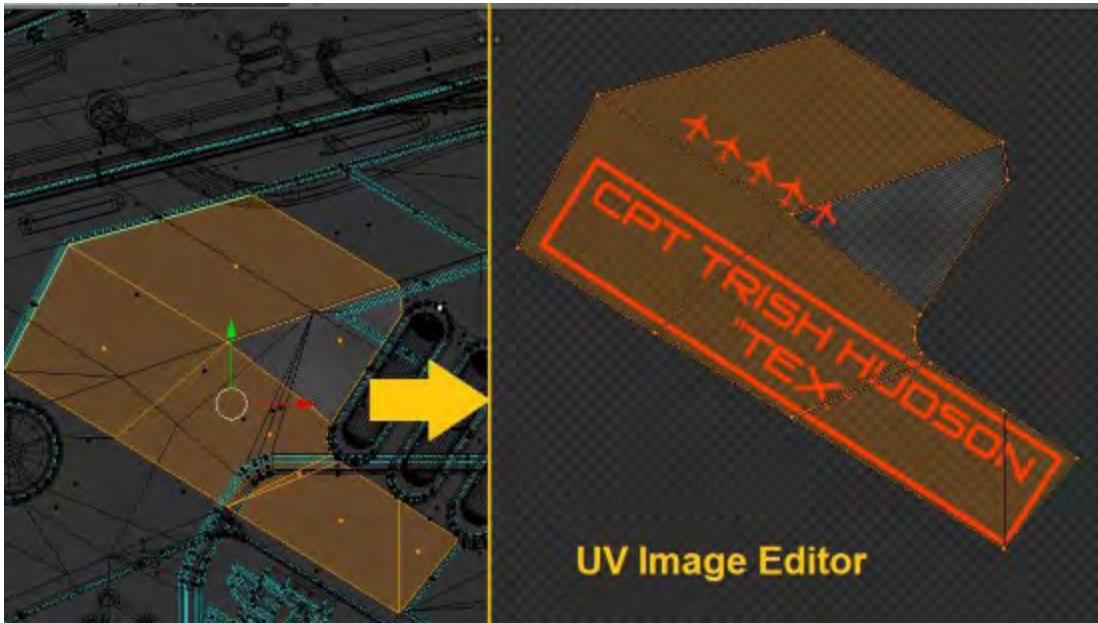
UV mapping tells Blender how to apply a two-dimensional image to three-dimensional objects.

Later in the book, we'll use UV mapping to unwrap and texture an entire model, and we'll look at the different settings and options. In this case, however, we're just using it selectively to apply a few decals to the model.

We can pick an area of the ship where we'd like to have writing or a decal. The easiest way to unwrap areas for decal use is to line up the faces so that they're roughly flat in the 3D viewport (you can also use Shift + 7 to precisely align the view to your faces). Then, you can press U to bring up the **Unwrap** menu; select **Project from View**:



In the **UV / Image Editor**, you can now align those faces with an image of your choice:

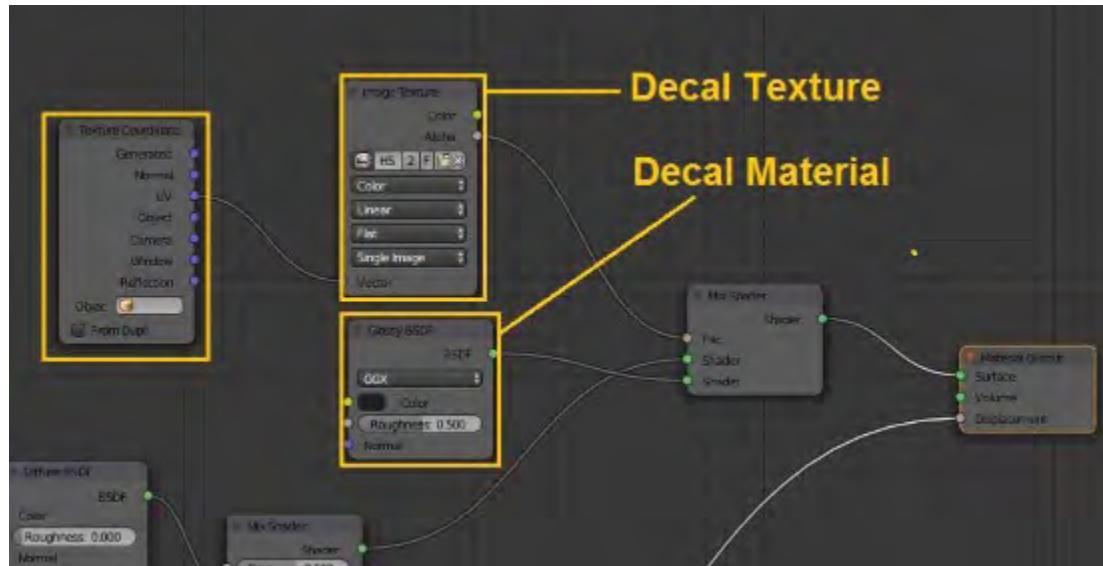


You can assign multiple parts of the model to the same area of an image. If a decal is repeated, it's easy to use one image and replicate it.

Once you've lined up your UV maps with an image, we'll need to add that image to our material. In this case, we'll ignore the color of the image. Instead, we'll just use its alpha or transparency value to determine which areas of the model will be affected by it.

<pagebreak></pagebreak>

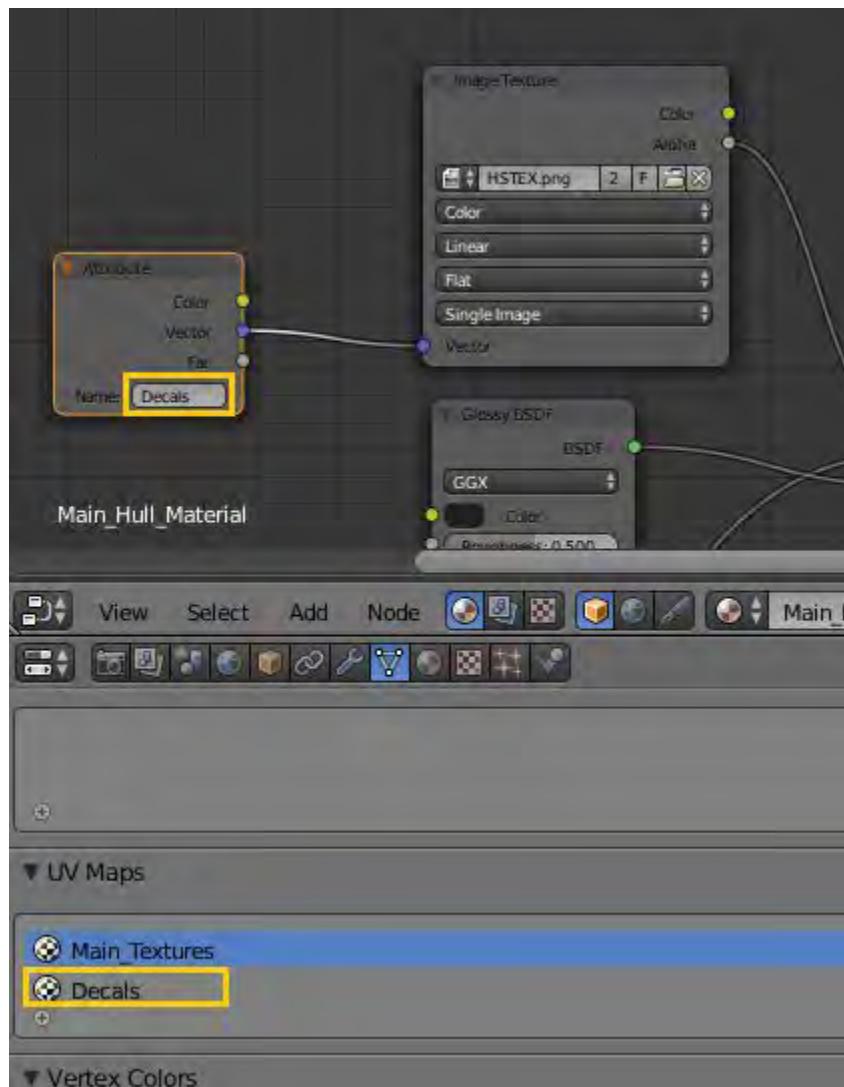
For our decal color, I'm just using a low-gloss gray color.



You can see here that I'm using a **Texture Coordinate** node (found under the **Input** submenu). This tells Blender to use the UV coordinates to align the image.

<pagebreak></pagebreak>

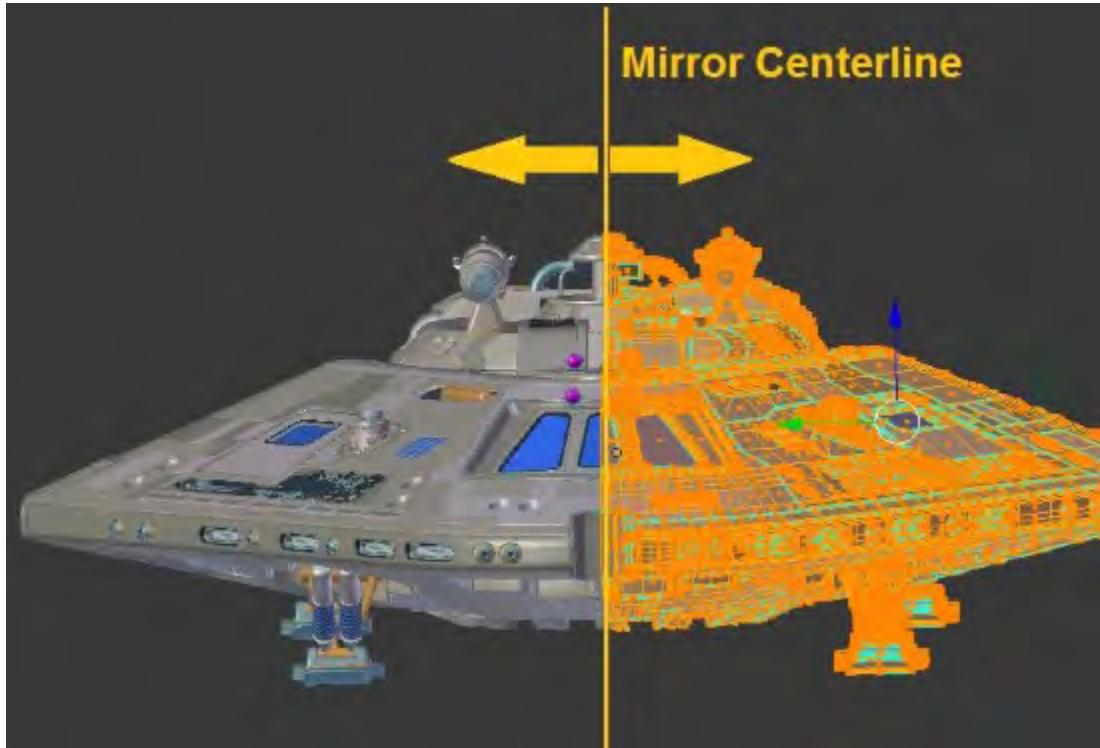
Note that it's possible to have more than one UV map in a model. You may wish to use two or more images that are mapped to the model differently. In this case, you'll need to use the **Attribute** node (also under the **Input** submenu) to specify which UV map you want to use.



<pagebreak></pagebreak>

One problem you may run into when doing decals is that UV maps (just like the mesh itself) are mirrored between the two sides. This means that if you unwrap some words or letters on one side, they will be "flipped" (mirrored) on the other side.

It can be helpful to separate areas from the model that will have decals on them. Then, you can apply the mirror modifier to those parts and join them with your "non-mirrored parts" object. That way, you'll be able to unwrap them separately.



<pagebreak></pagebreak>

Other possibilities

With a few decals, here's what my final ship ended up looking like:

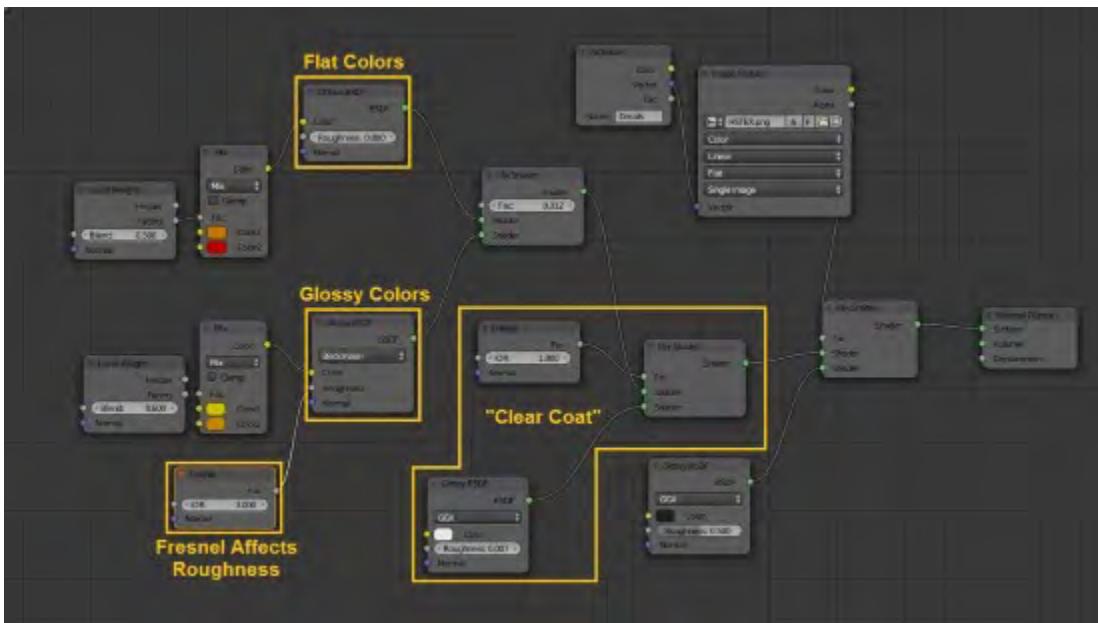


<pagebreak></pagebreak>

Of course, that's hardly the only option we have for materials. If you'd prefer, you could add shiny paint materials instead:



Basic paint isn't particularly difficult. It's just a combination of diffuse and glossy shaders. If you want to have a clearcoat layer to your paint, you can use the **Fresnel** input to add it as well. The basic paint setup shown will get you started.



As you can see, there are many options for adding materials and textures to our spacecraft. Take your time and play with the various options!

Summary

In this chapter, we've added materials and textures to our spacecraft and created a basic scene to render it. That wraps up our spacecraft project, though of course you can do plenty of other things with it if you'd like.

In the next section, we'll look at a different type of modeling and rendering—a stylized type known as Freestyle, which can be used for line drawings, blueprints, and other non-realistic renderings.

Chapter 7. Modeling Your Freestyle Robot

In this chapter, we'll look at another application of Blender: creating non-photorealistic (NPR) renders using Freestyle. We'll take a look at a way to optimize a model for this purpose, and then model a simplified robot. The following topics will be covered in this chapter:

- Modeling for Freestyle
- Blocking out your robot
- Creating the body with Subdivision Surfacing
- Finishing the robot

Modeling for Freestyle

Before modeling our robot, we need to briefly talk about what Freestyle is. Freestyle is a non photorealistic rendering engine that works as an addition to **Blender Internal** or **Blender Cycles**, which is intended to produce procedural lines on the top of a render. These could be line drawings, anime-style images, cartoons, comics, blueprints, or anything similar. The first thing we'll do is switch from **Cycles Render** to **Blender Render**:



Note

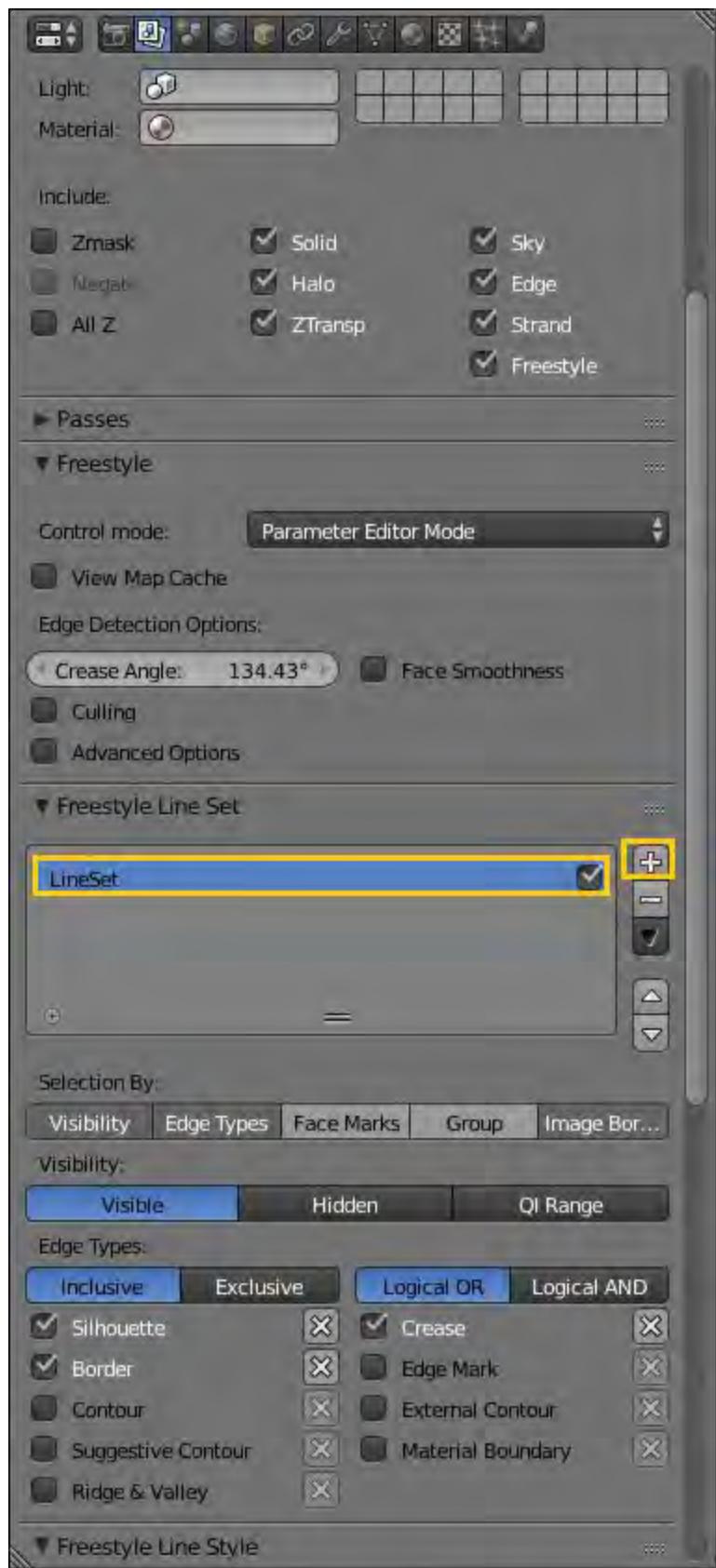
Freestyle is now supported by Cycles as well, but for a long time, that wasn't the case. You could make the argument that Blender Internal (or "BI") is more suited to NPR work. Besides, we haven't used the internal render engine yet in this book, so this is a good opportunity to play with it.

Next, you'll have to check **Freestyle** under your rendering tab:



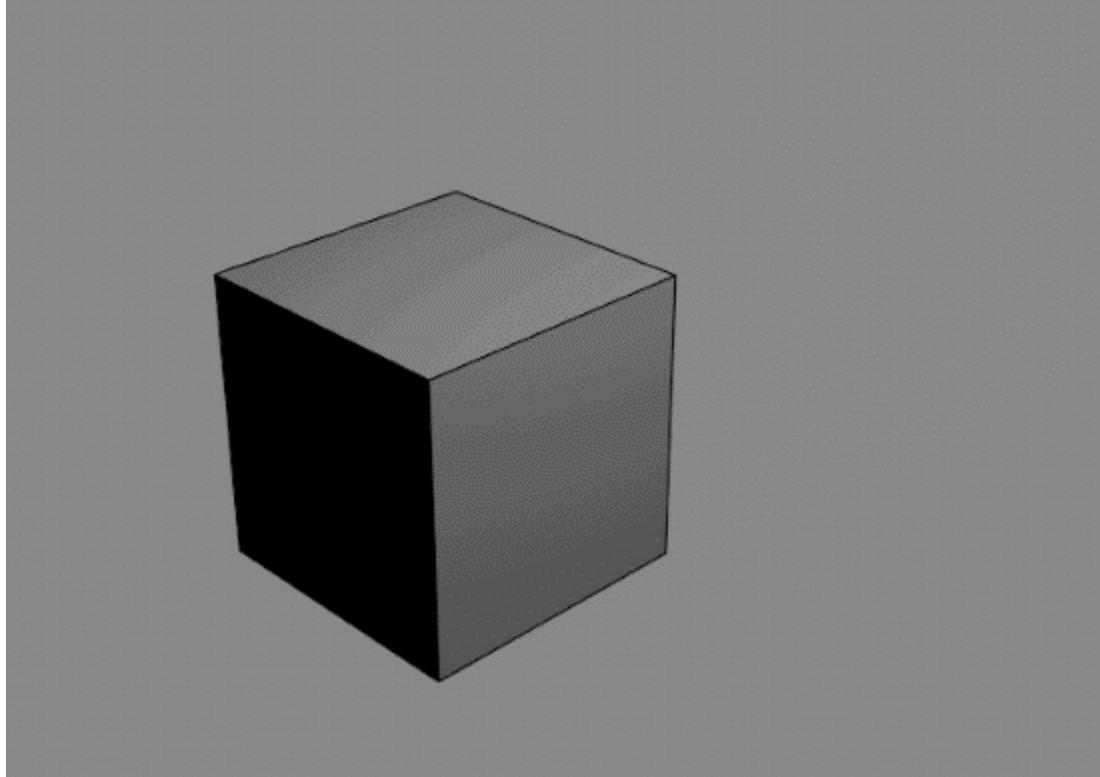
<pagebreak></pagebreak>

Then, under the **Render layers** tab, you can add a new **LineSet**:

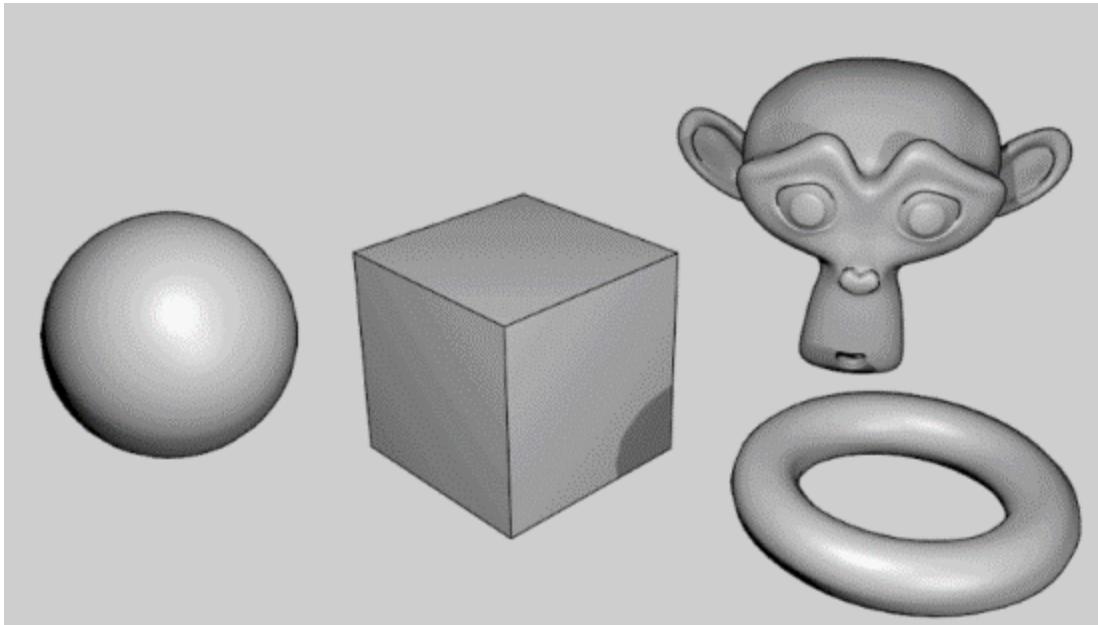


<pagebreak></pagebreak>

Now, when you render a scene, you will see the freestyle edges:



For simple objects like a cube, this is pretty straightforward. Blender knows where the edges are, and quickly fills them in with freestyle lines. Try this with various objects and see how it looks:



<pagebreak></pagebreak>

But now, let's do something different. We'll take the gun model from our first project and attempt to render it with **Freestyle**. Here's how it looks:



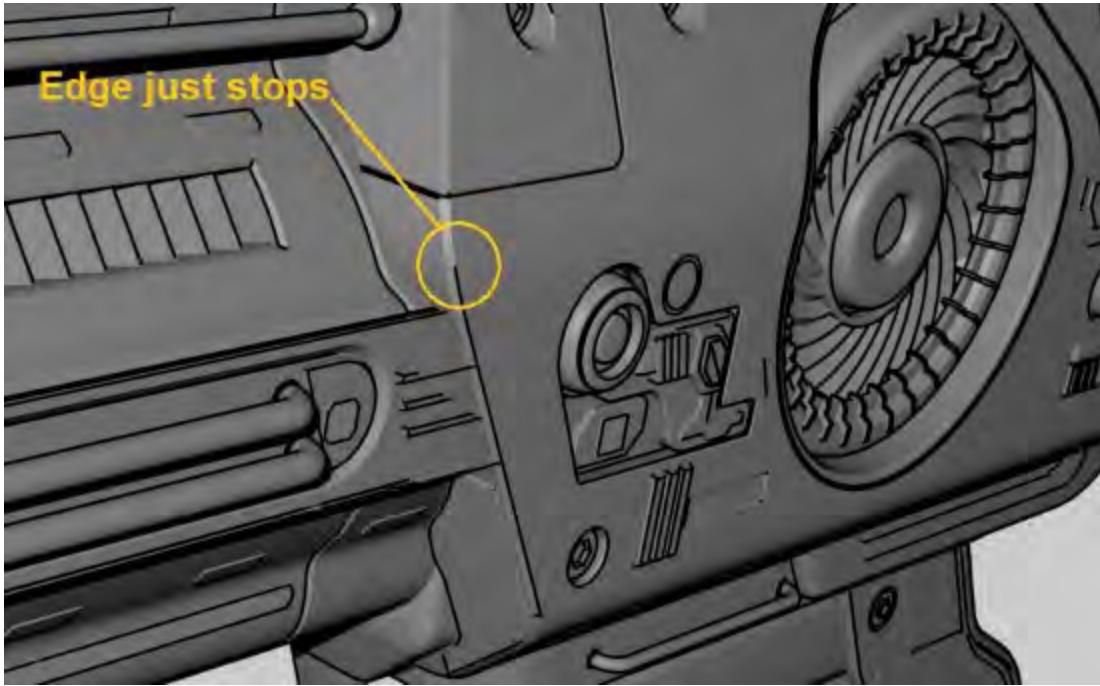
It does not look great. First of all, you can see one of the key problems here:



<pagebreak></pagebreak>

It looks like we're missing lines where the small detailed pieces join with the main body. That's because there's no actual connection there—the detailed piece is just sitting on (and slightly penetrating through) the main mesh. That works fine for normal rendering, but it confuses Blender's Freestyle function.

There are other problems as well. For instance, beveled edges don't work well with Freestyle either:

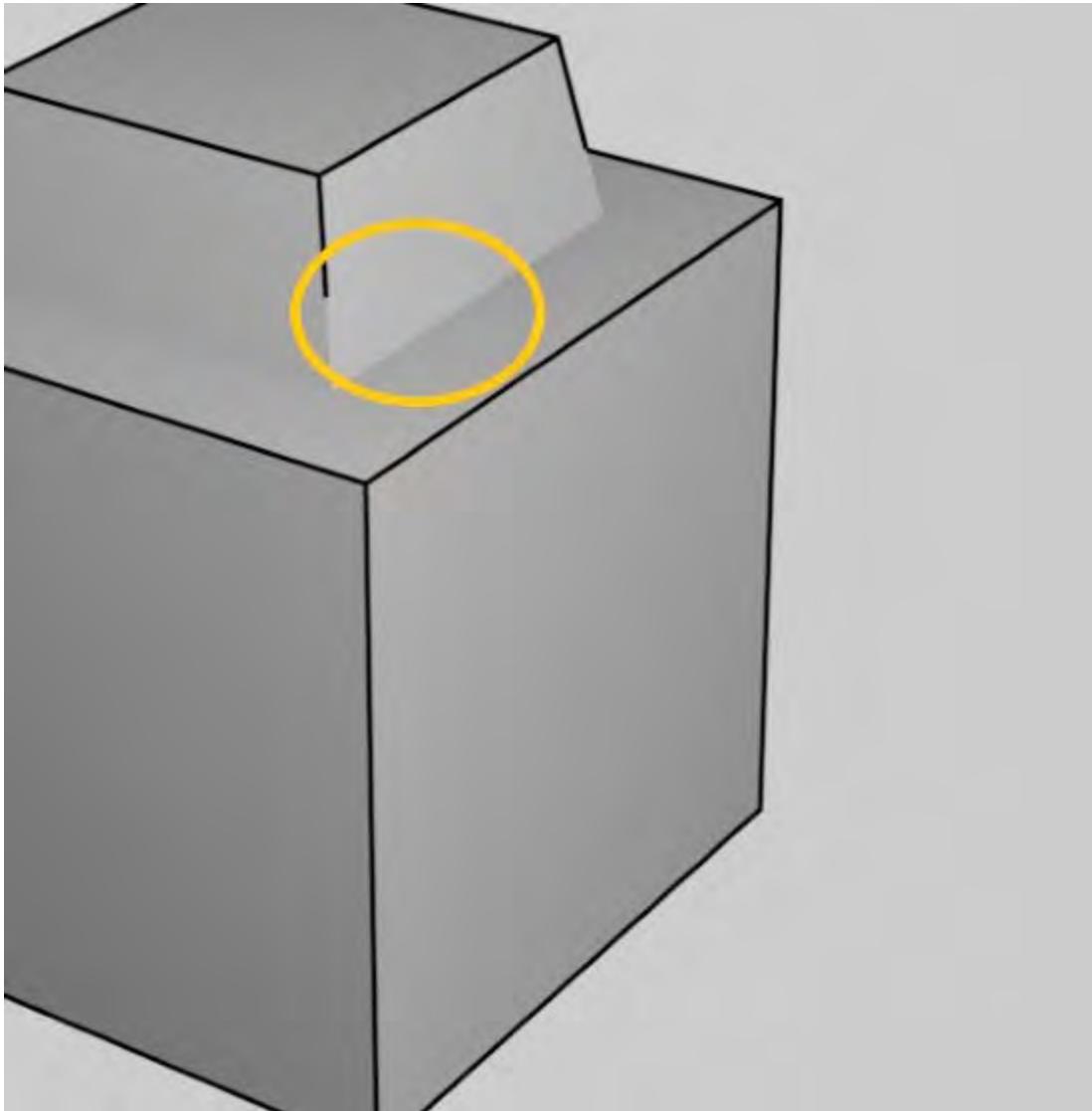


This leads us to a key point:

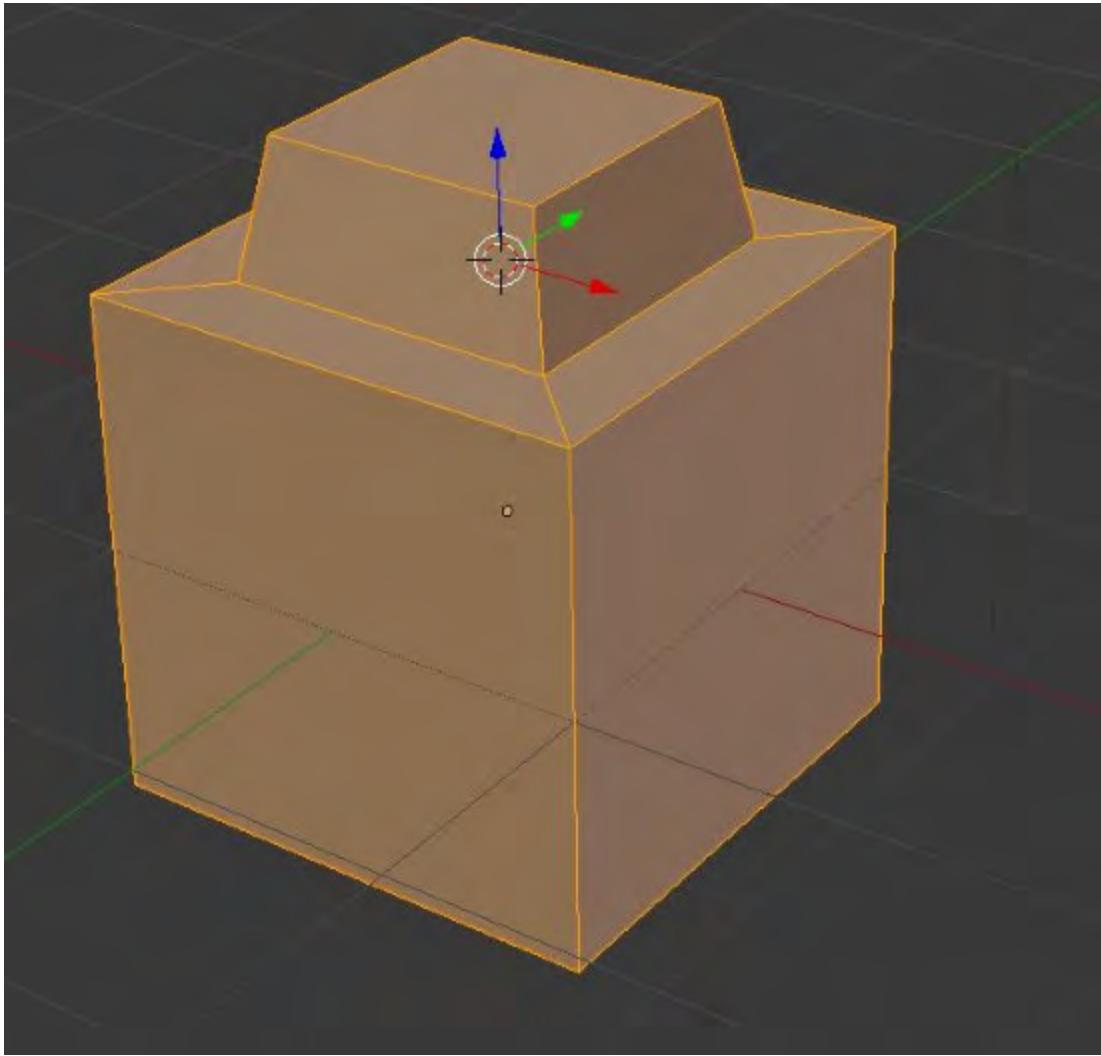
You can render any model with **Freestyle** (just by checking the box), but for the best results, models should be created specifically with freestyle in mind.

<pagebreak></pagebreak>

So let's talk about how to do that. First, we'll address the problem of meshes that penetrate other meshes:

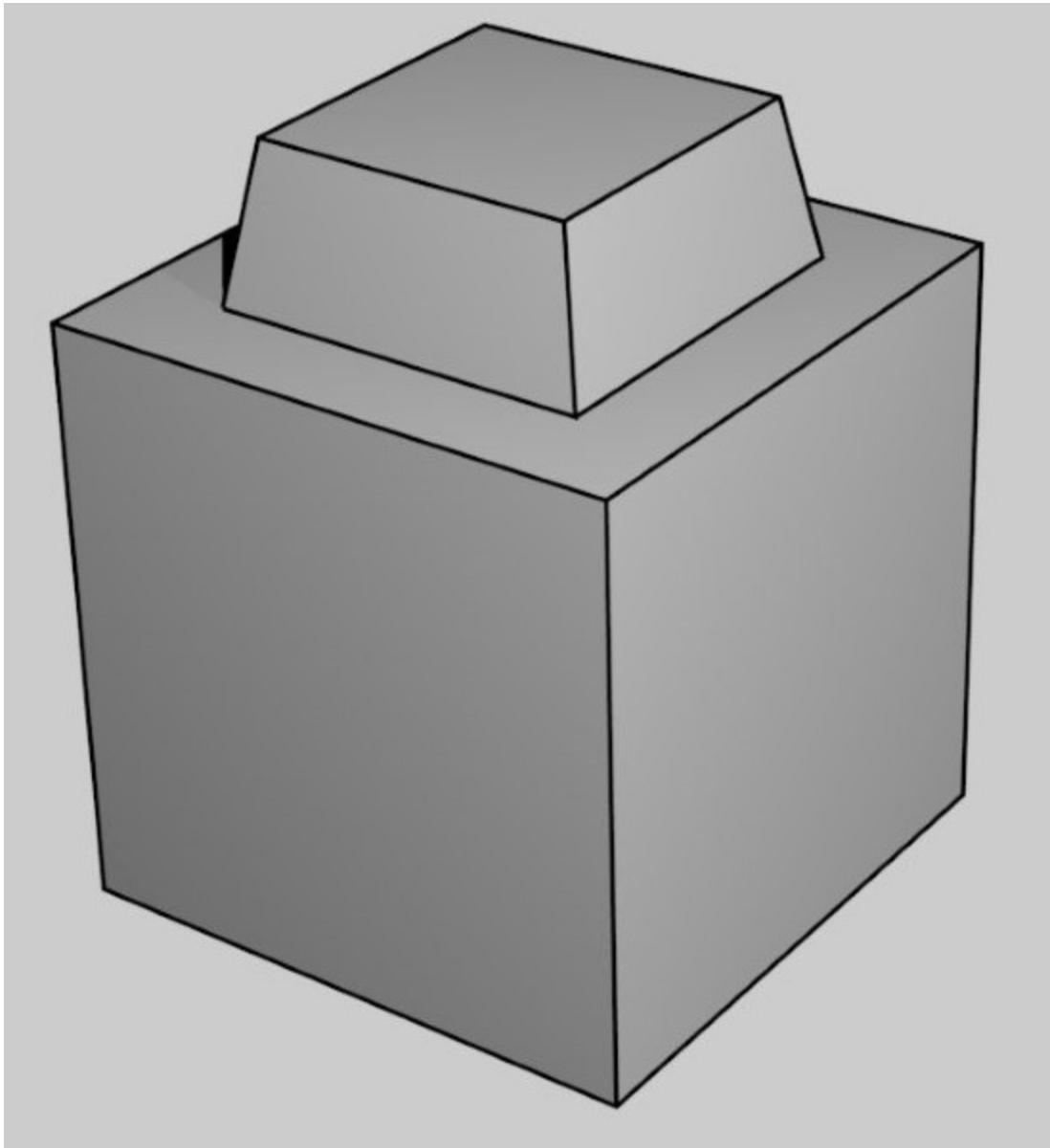


One solution is to connect all the vertices in a mesh, creating watertight or non—manifold geometry:

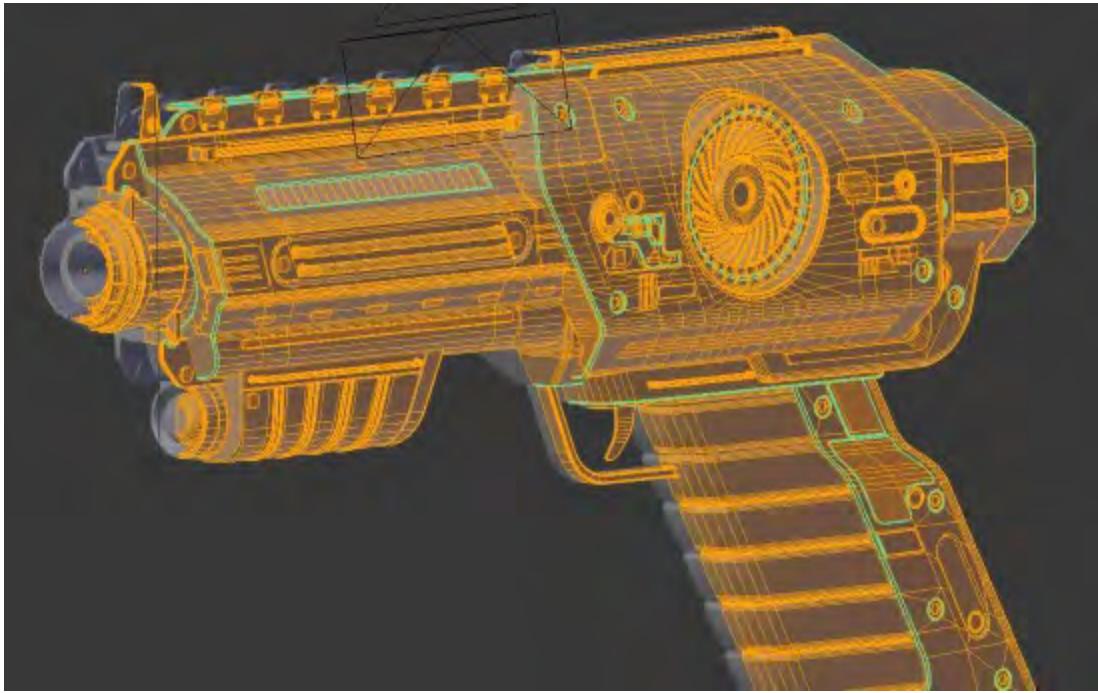


<pagebreak></pagebreak>

This will certainly solve the problem:



However, that doesn't mean it's the best (or even a good) answer. For a simple object like our cube, it works fine. But, imagine if we had to do that for a more complex model like the gun:



<pagebreak></pagebreak>

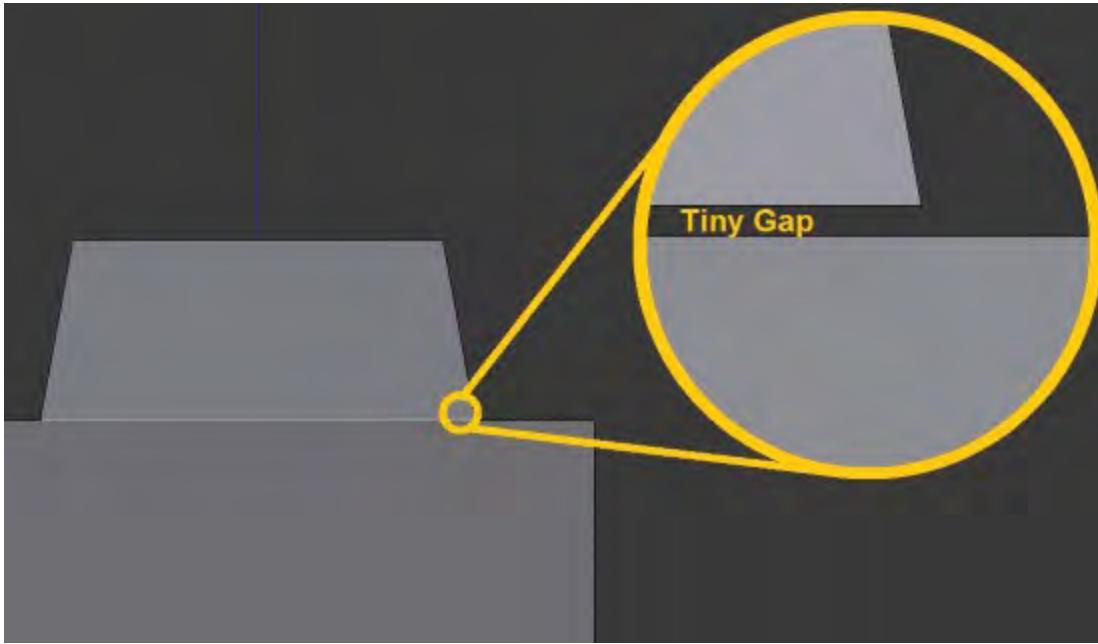
It would be a huge amount of work, and it would drive the polygon count through the roof. You would probably quadruple the geometry in the process. Imagine an even more complex model, like our spaceship:



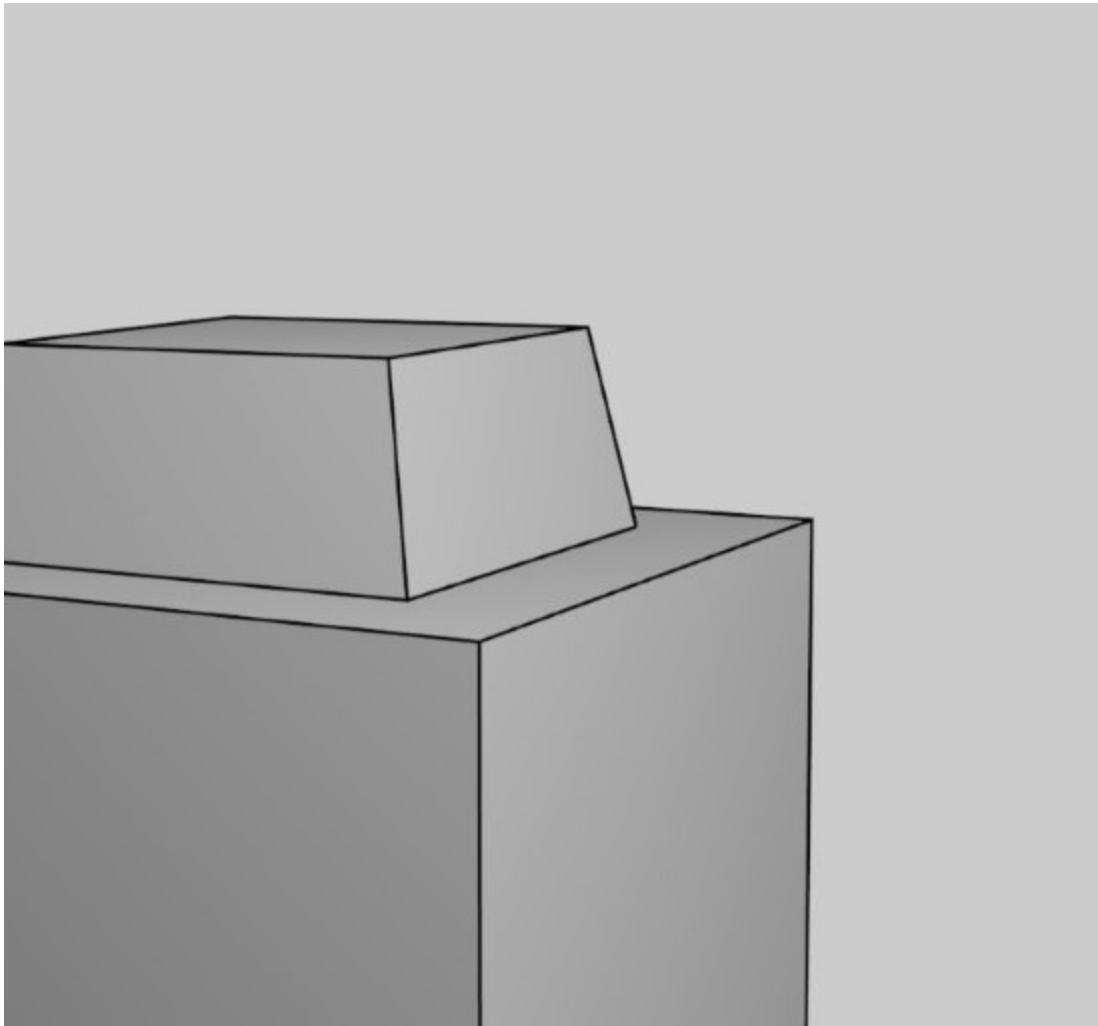
It might require millions of polygons. If we had multiple ships or other objects in a scene, we could quickly overwhelm Blender's ability to handle it. Even if you could handle it, renders would take forever (possibly multiple hours).

<pagebreak></pagebreak>

So, no, that's not always going to work. Another option is to move one piece just a tiny bit above the surface of another mesh:



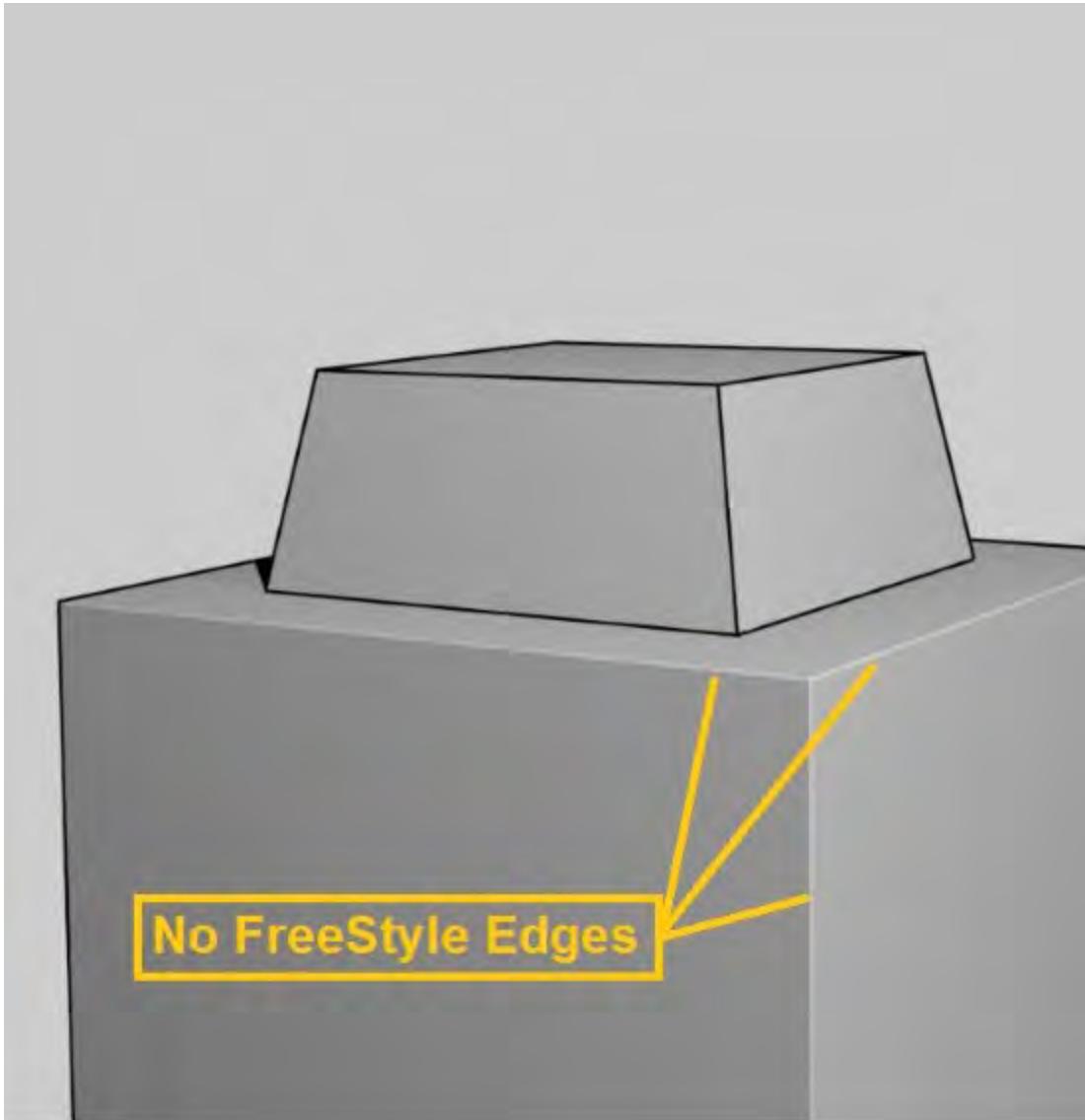
A very small gap will be obscured by the freestyle lines themselves and not even noticeable from most angles:



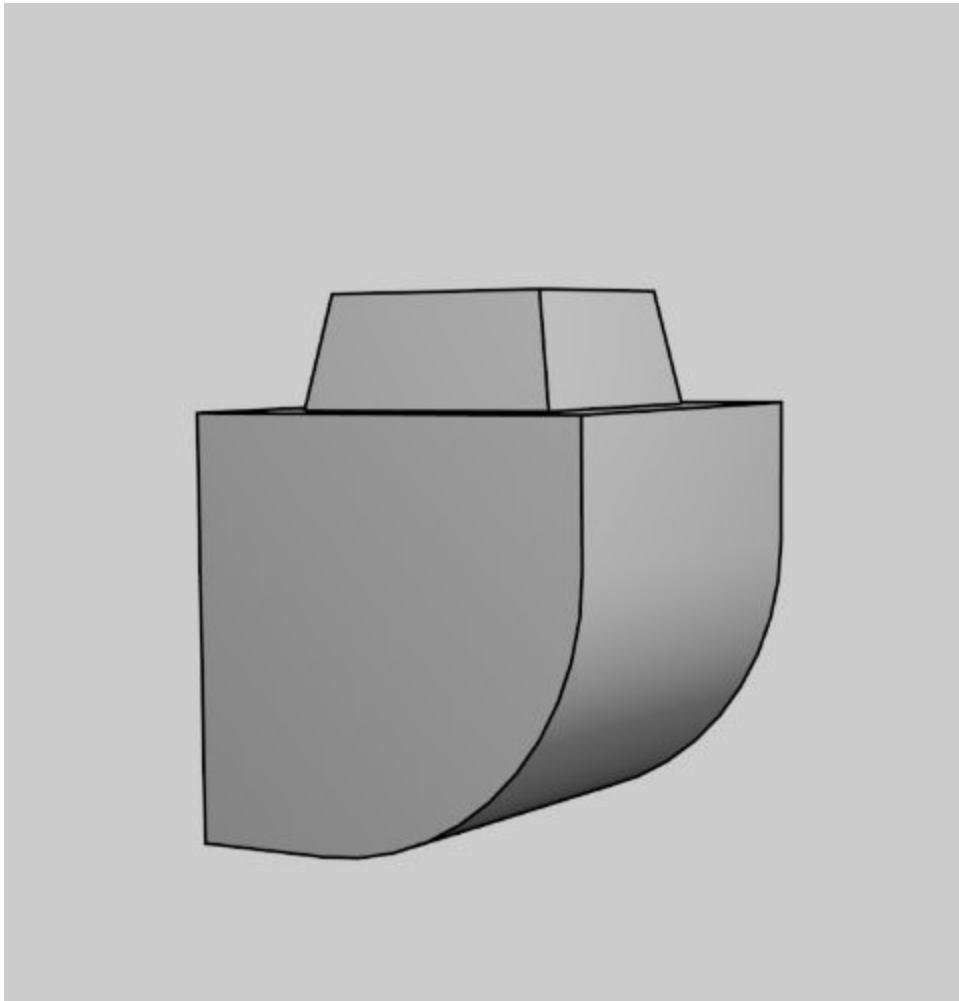
That's a much quicker way that doesn't require so much work (and so much extra geometry).

<pagebreak></pagebreak>

Now, let's briefly discuss the beveling problem that we had:



This is a much easier problem to solve—just don't bevel your meshes when they're intended for freestyle. Note that this only refers to minor edge beveling. Large rounded parts ("Big Beveling") are still fine:



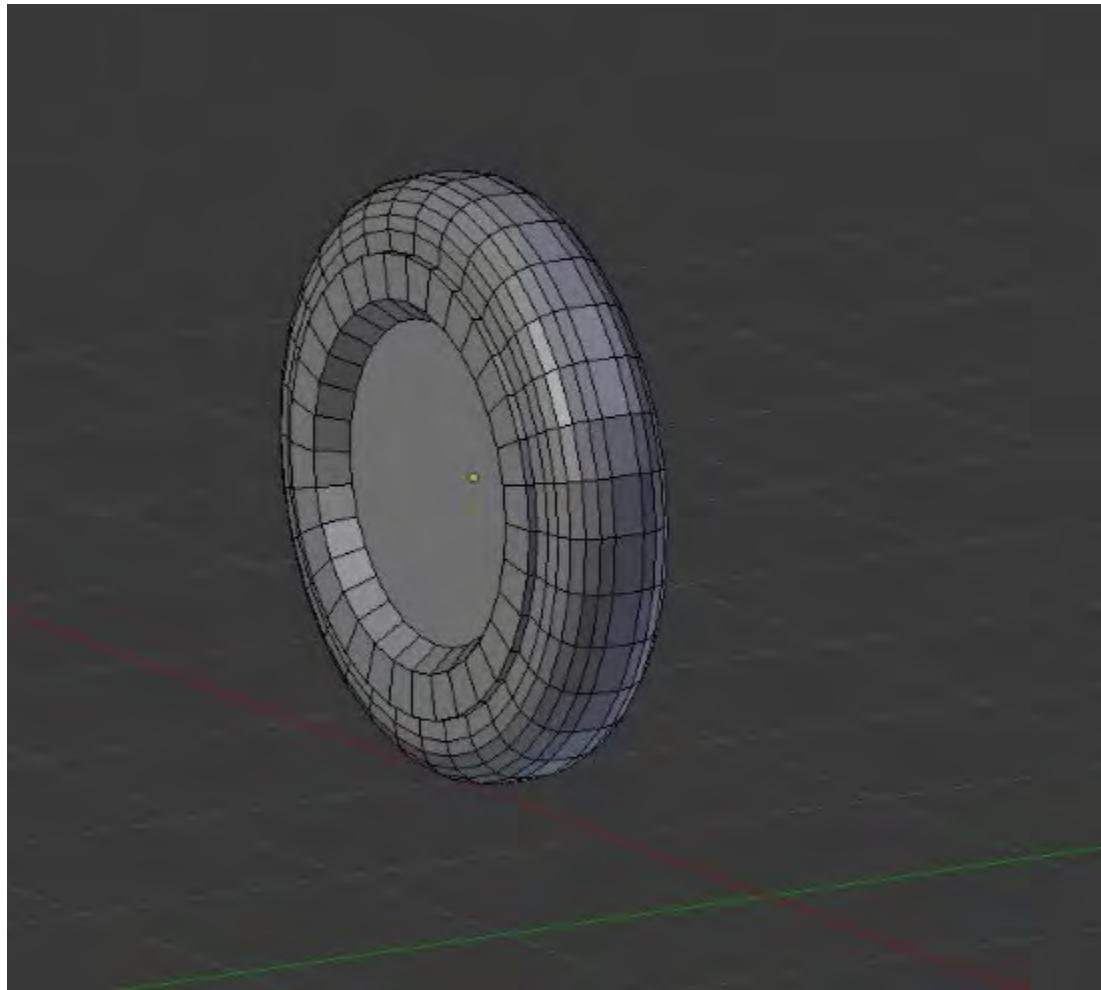
<pagebreak></pagebreak>

Blocking out your robot

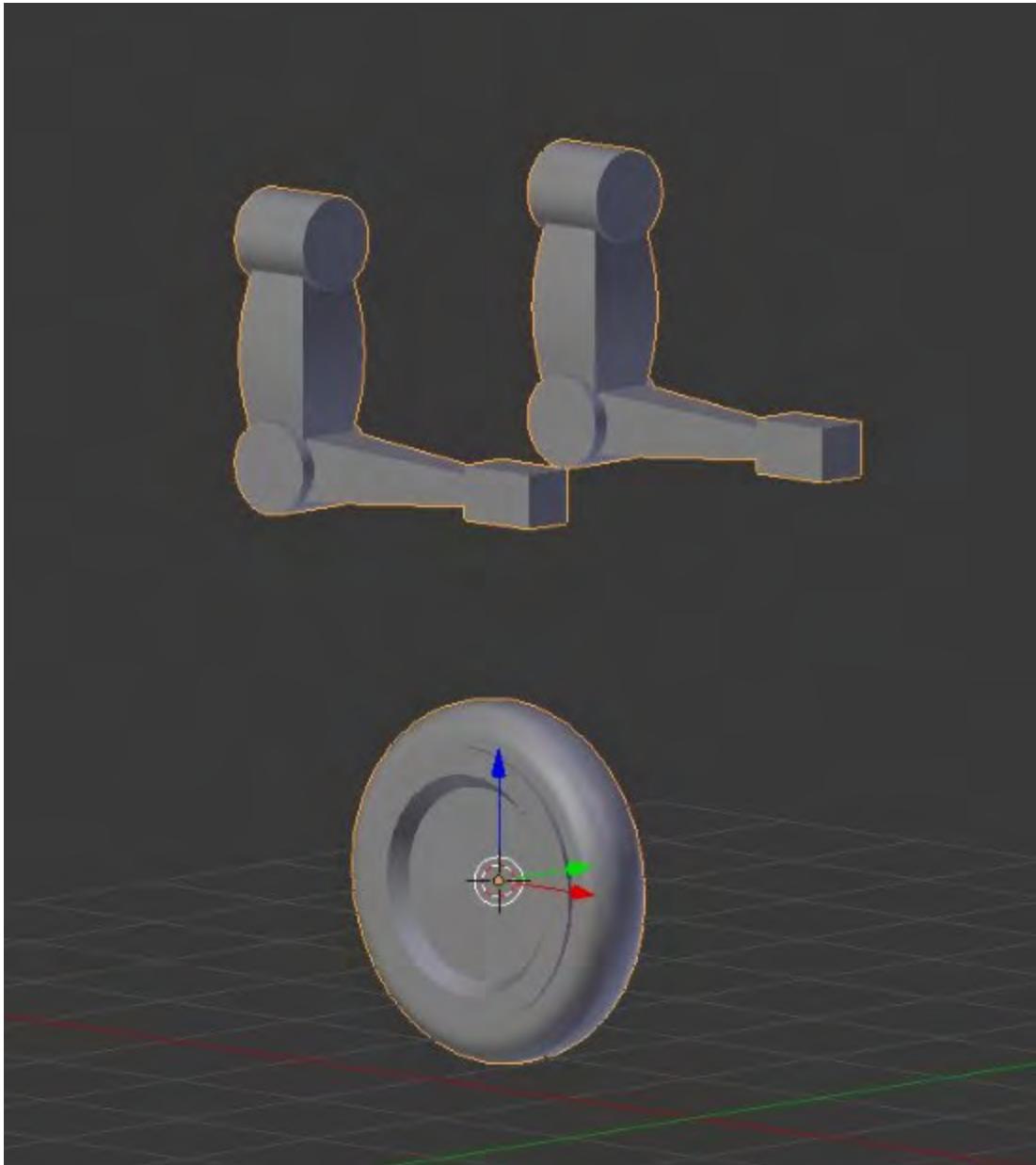
So now, let's move on to building our robot model.

Since this is going to be stylized (not realistic), I'm not going to go crazy with the detailing.

I'll start by blocking out some basic shapes. We'll use a wheel for the robot to roll on:

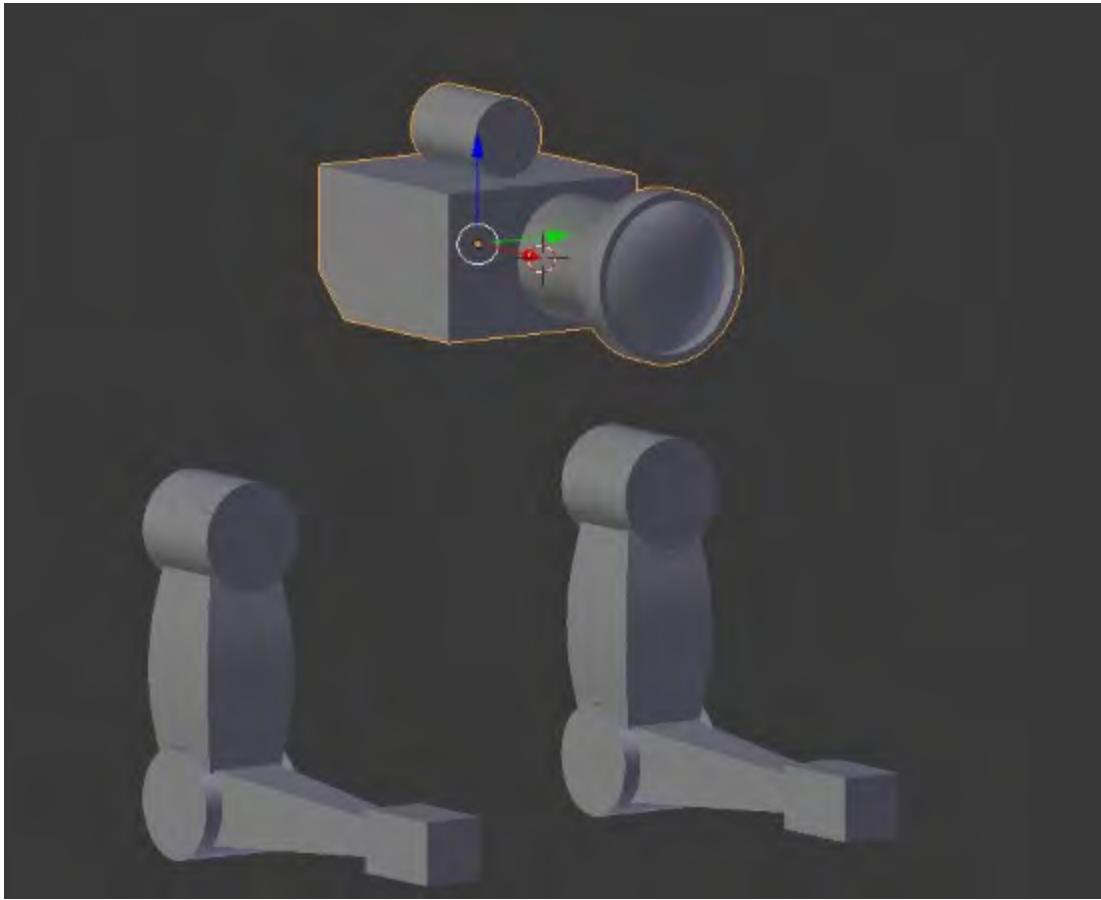


Some basic arm shapes:



<pagebreak></pagebreak>

And a head that looks somewhat like a video camera:



<pagebreak></pagebreak>

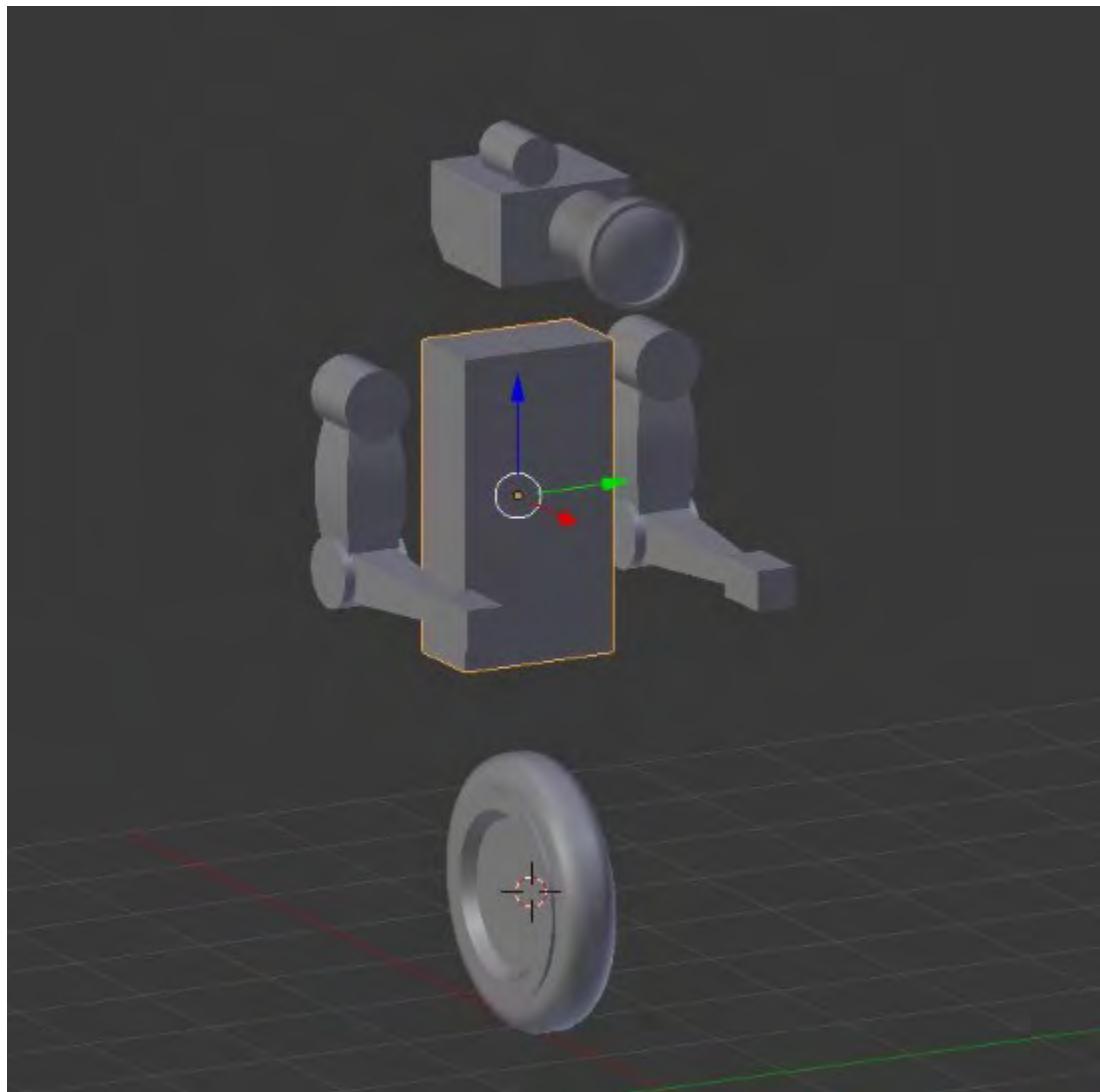
Creating the body with Subdivision Surfacing

For the body, we're going to use a modifier that we haven't talked about yet—**Subdivision Surfacing**.

More experienced users may find it odd that we're waiting until this part of the book before using Subdivision Surfacing (or **Subsurf**, as it's commonly referred to). After all, it's a tool that many people use frequently.

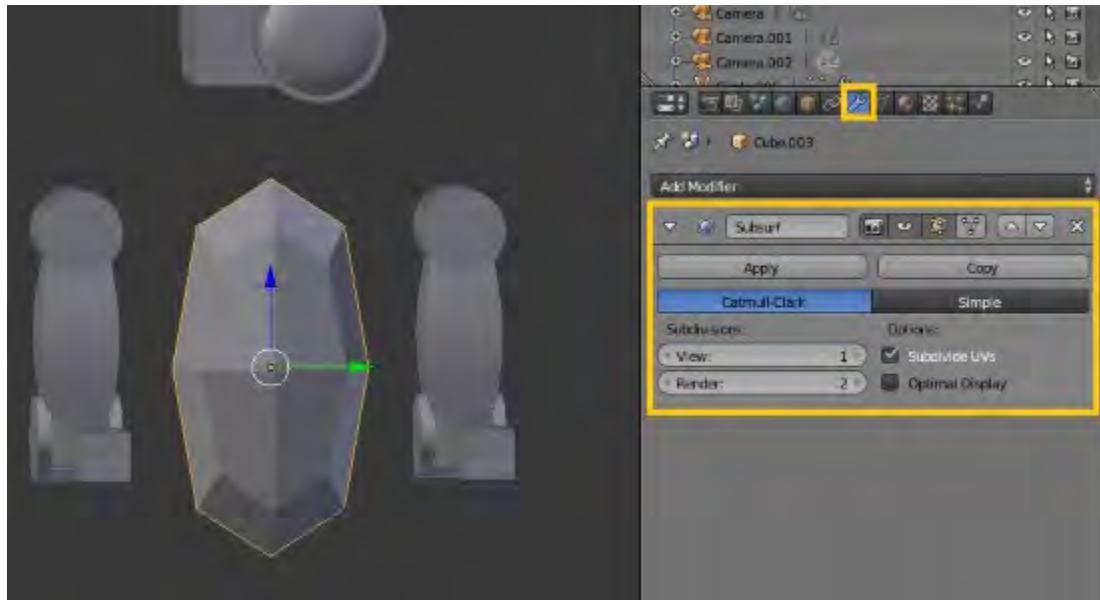
So, let's talk about that.

Subsurf works by dividing and smoothing a mesh. For instance, let's add a basic **Cube** shape for the body of our robot:

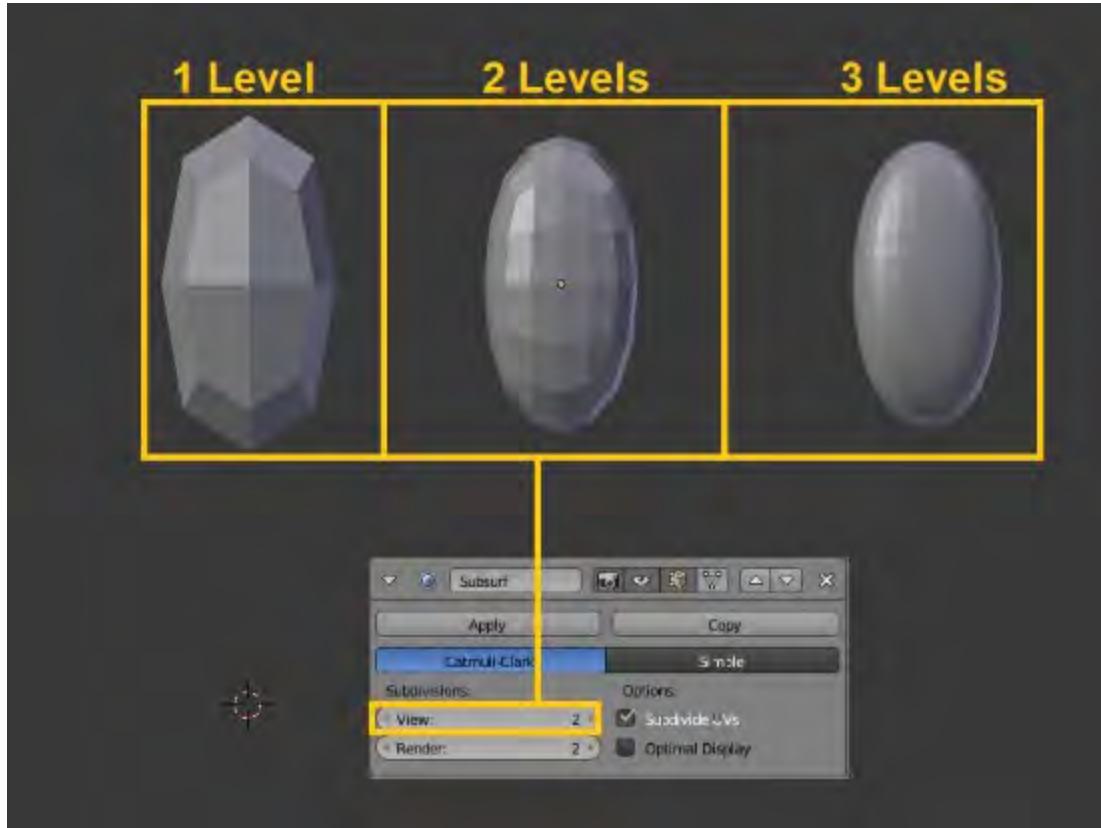


<pagebreak></pagebreak>

Next, let's add a **Subsurf** modifier to it:

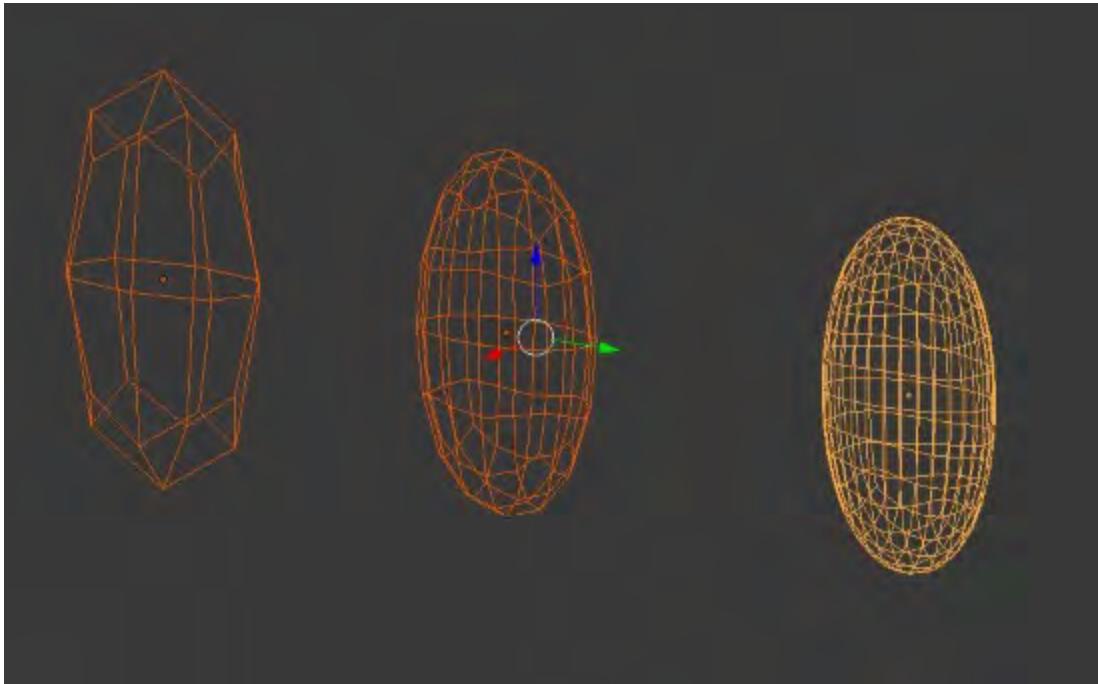


Subsurf has divided and smoothed our cube. The more layers of Subdivision Surfacing you add, the smoother it gets:

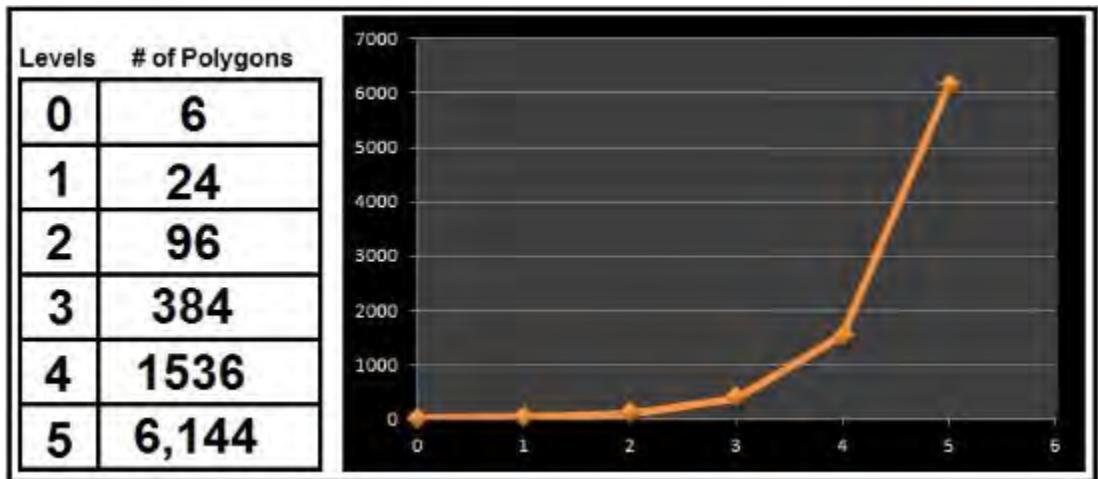


<pagebreak></pagebreak>

A very important note, however, is that with each layer of subsurfing, the polygon count goes up quite a bit:



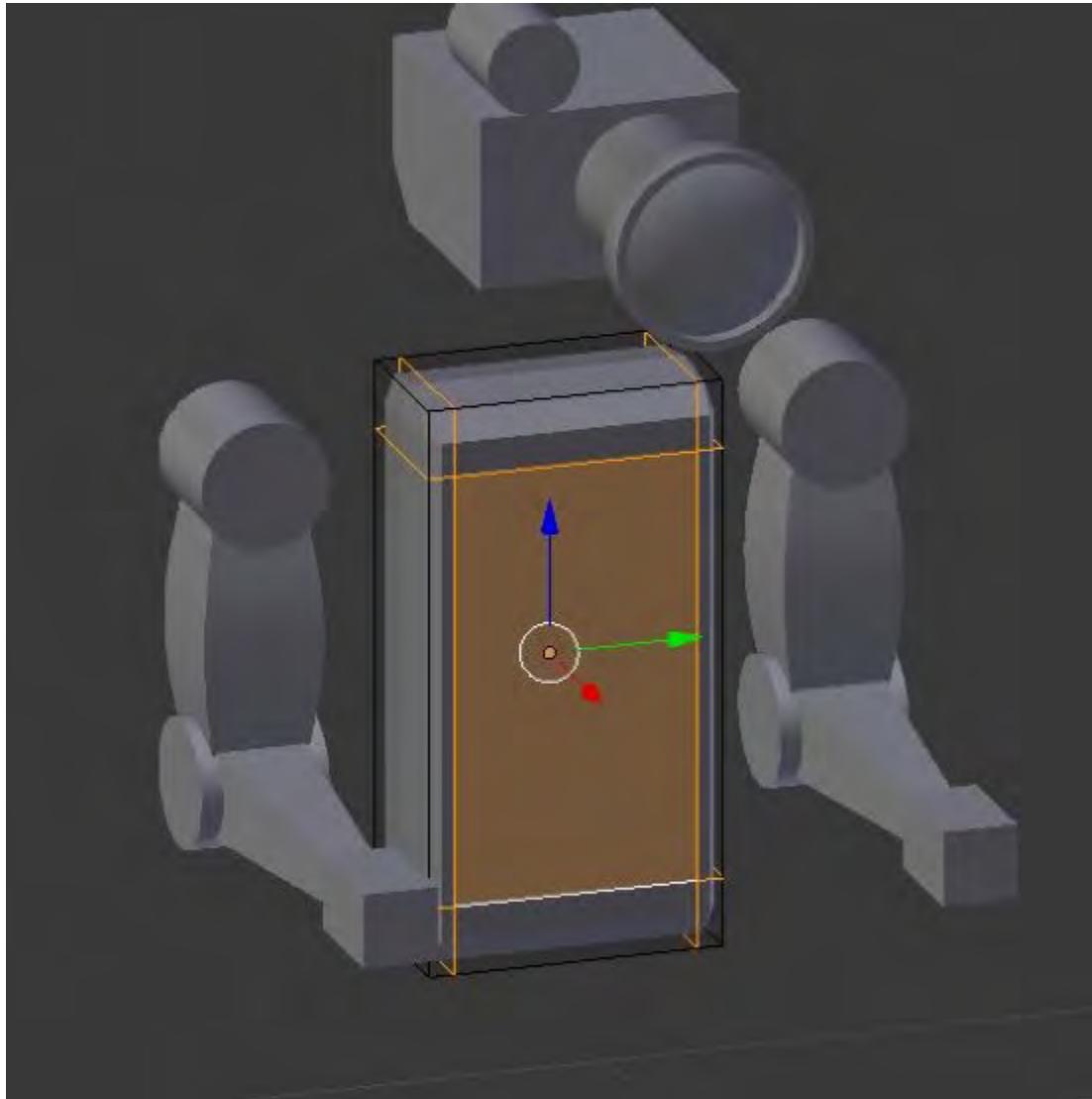
This can be quickly illustrated with a graph:



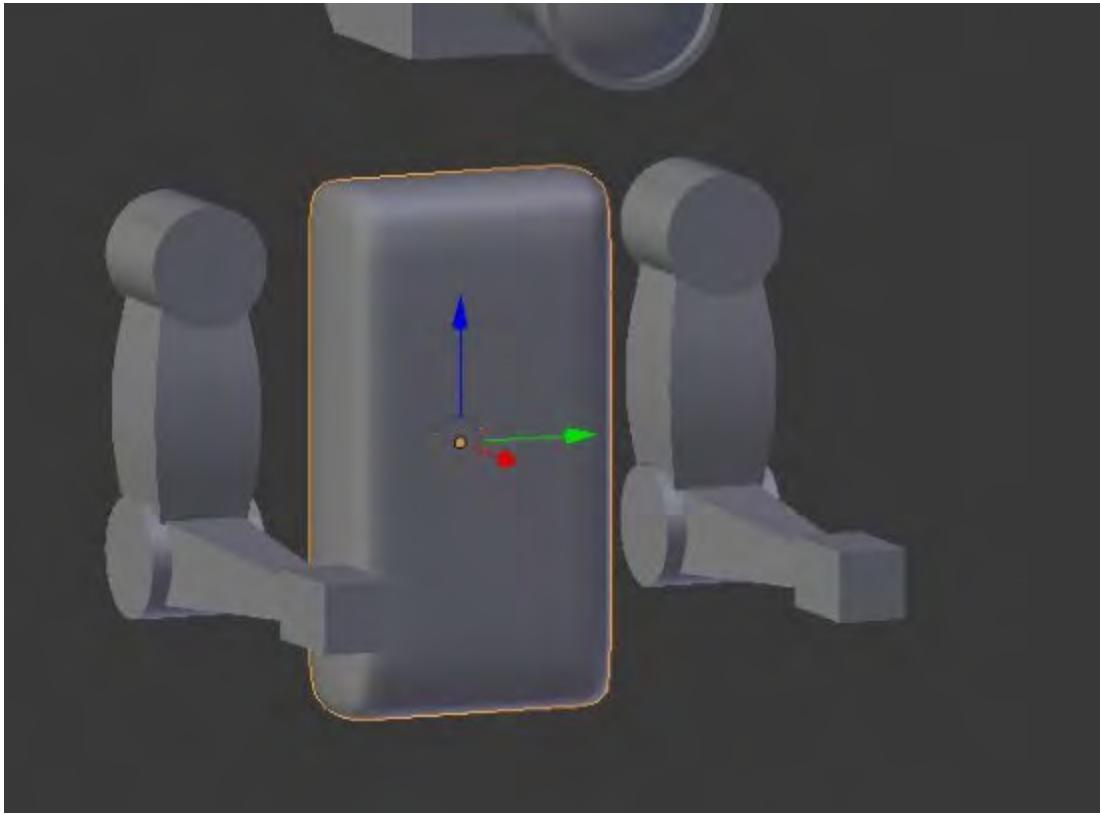
In practice, it would be rare to add more than two or three levels of Subdivision Surfacing. Still, the polygon count is one of the main drawbacks to it. **Subsurf** is a quick way to get smooth, flowing shapes—but, at a cost.

<pagebreak></pagebreak>

In order to add some definition, we can Tab into **Edit mode** and add a few loop cuts:

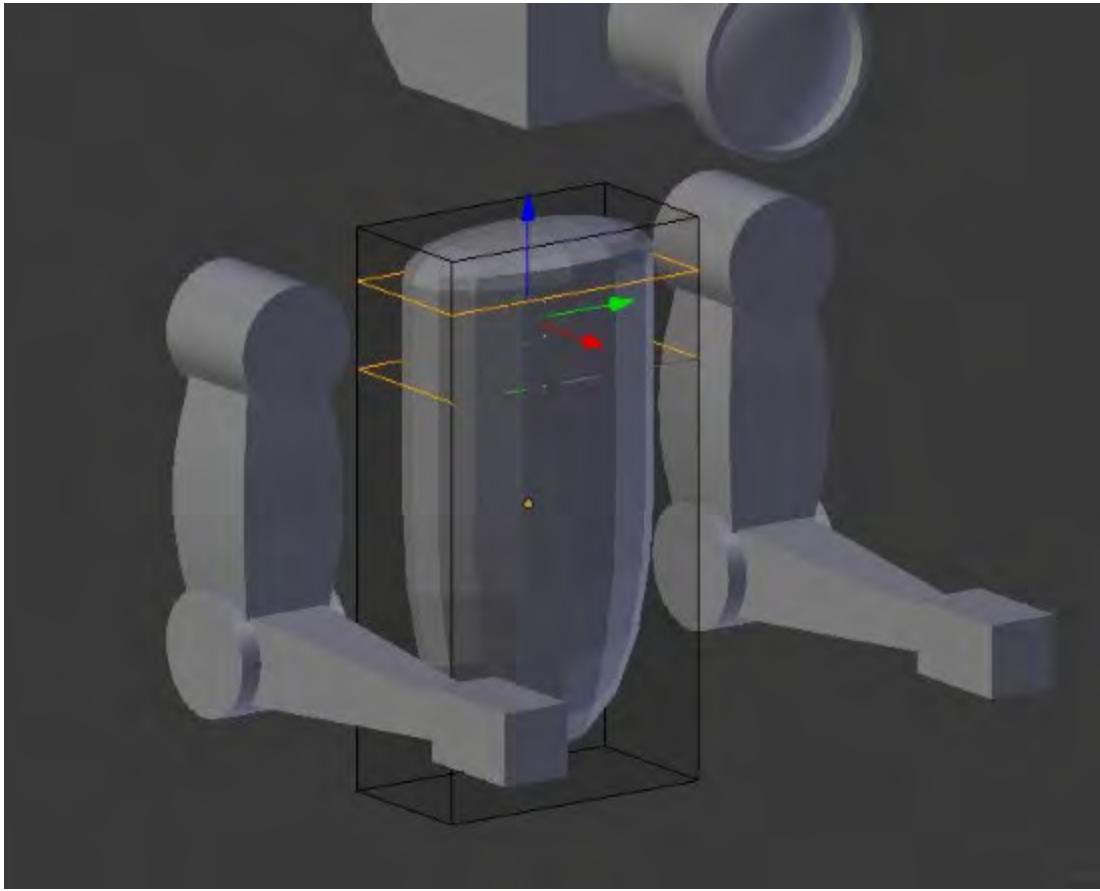


You can see the effect that this has:

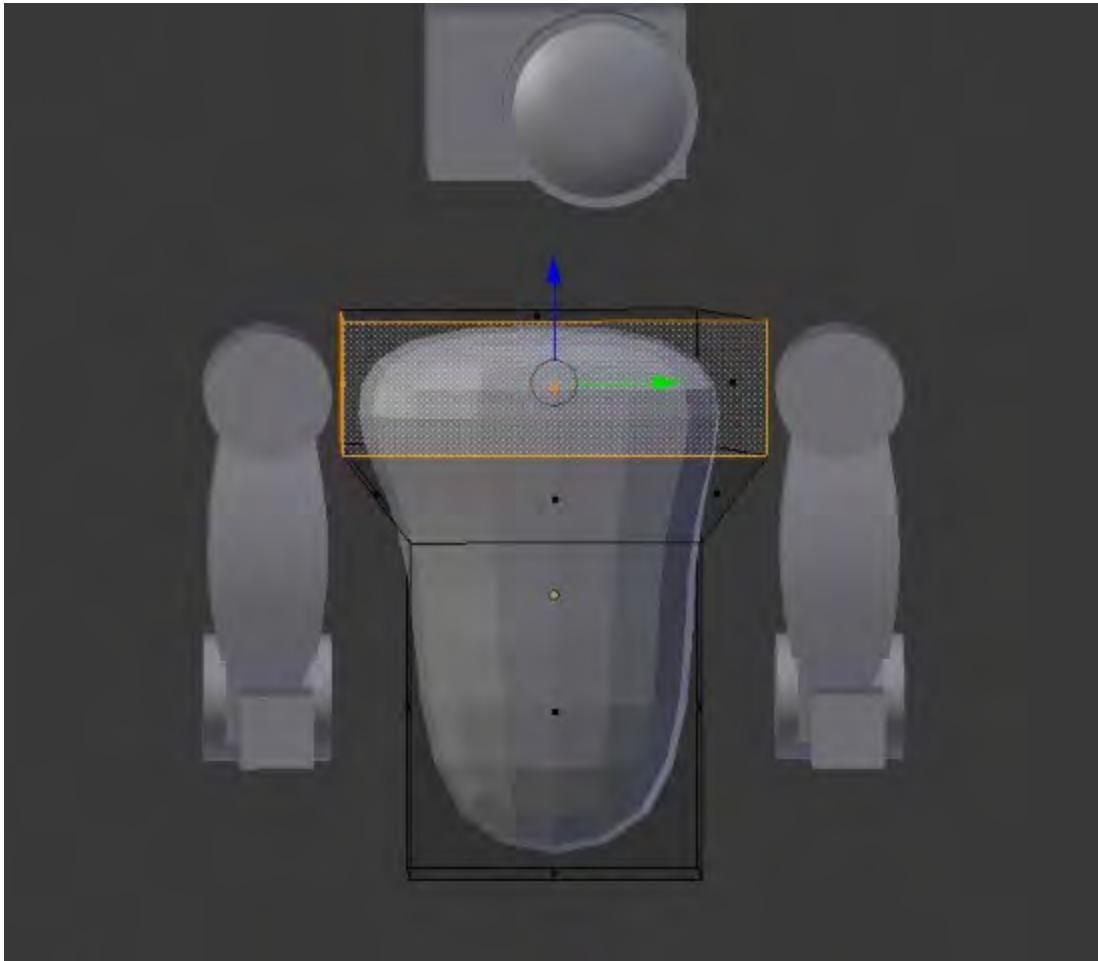


<pagebreak></pagebreak>

Let's remove those edge loops—it was just for demonstration—and continue building our body. We'll run a couple of edge loops around the upper torso area here:

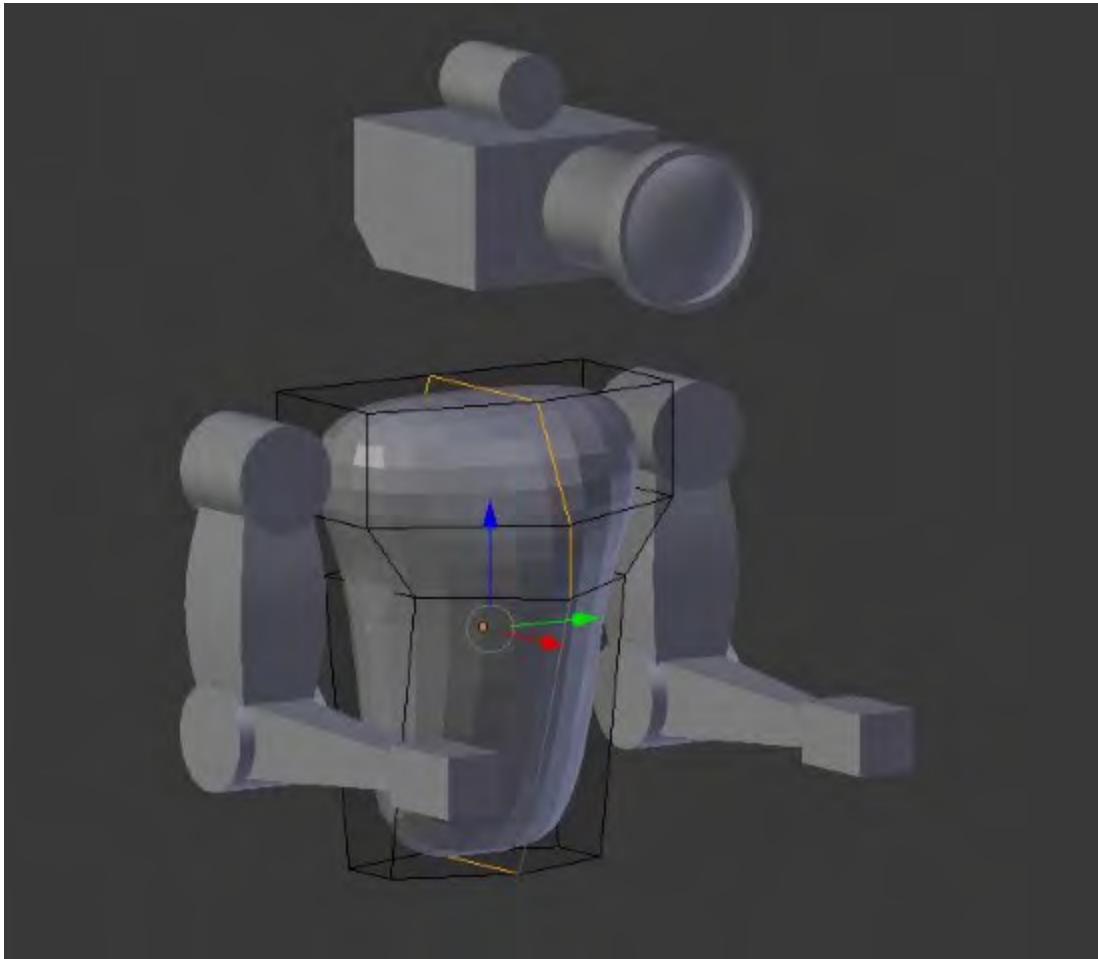


And widen the torso:



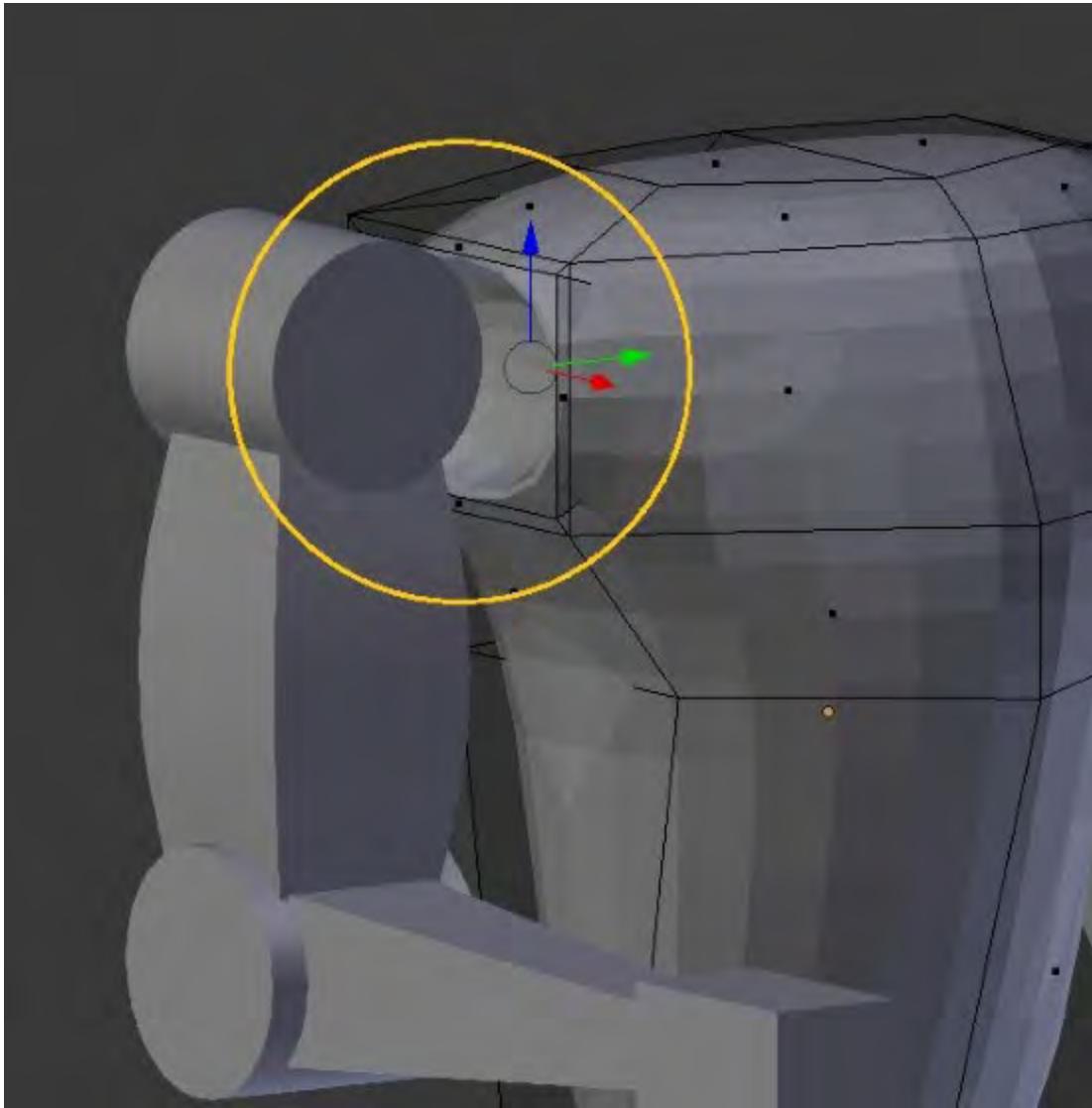
<pagebreak></pagebreak>

Then, we will run another loop around the body and start to refine the shape a little bit more:



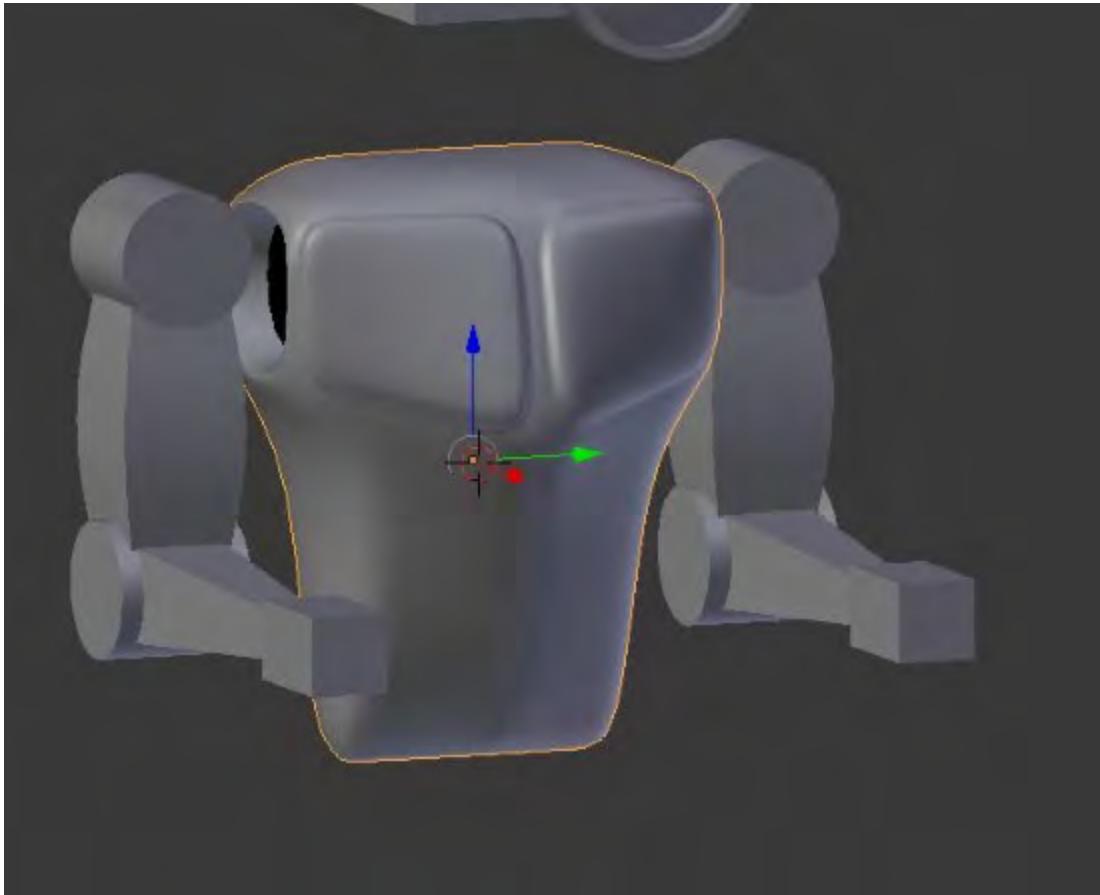
Remember that you don't need to do much (if any) beveling here. We're basically creating a "frame" or "guide" for the **Subsurf** modifier to create our body.

So, I'll next add some arm holes:



<pagebreak></pagebreak>

And maybe a bit of detail here, like chest plates:



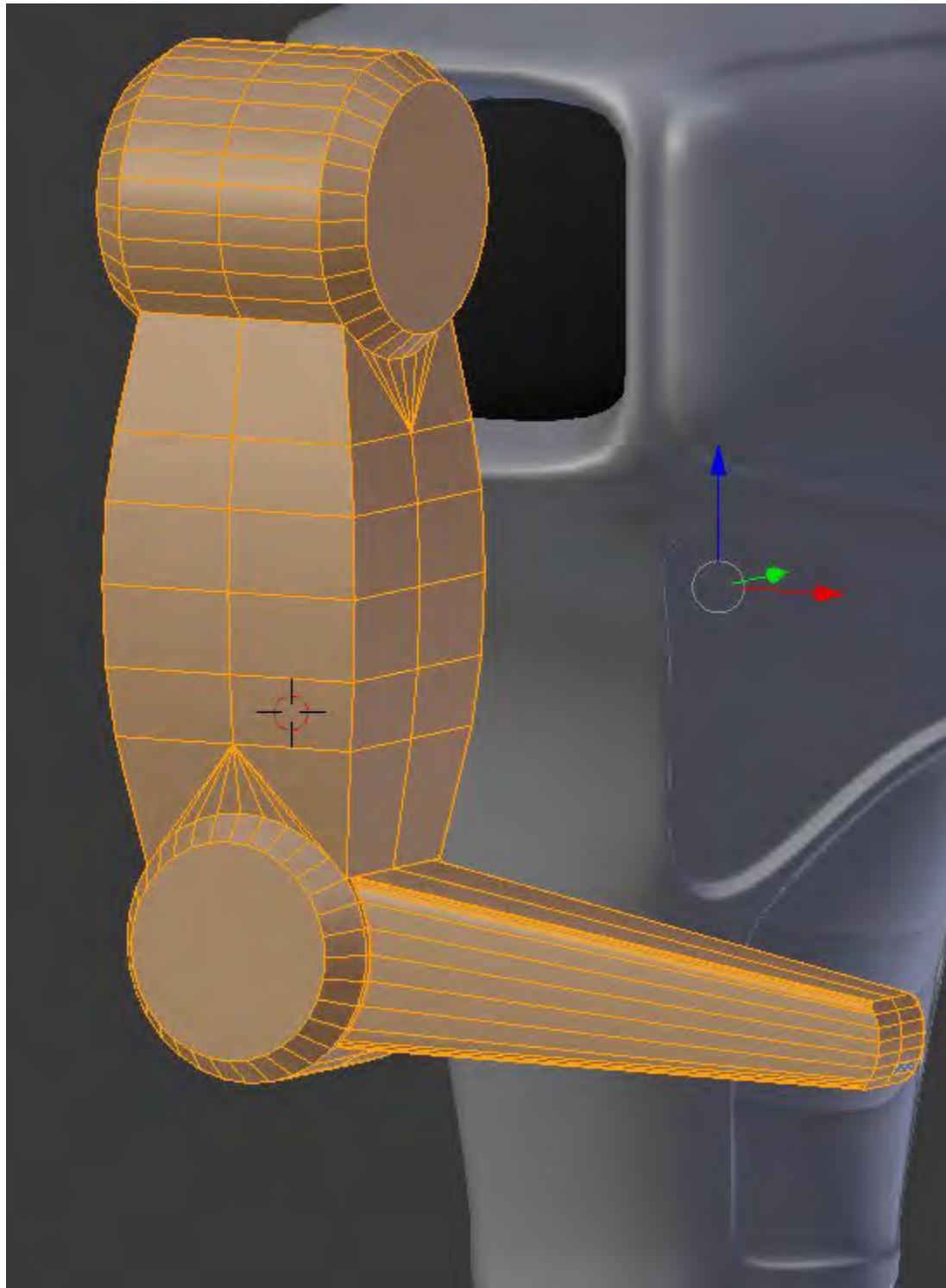
Just continue to add loop cuts and basic extrusions, until you get a body shape that you're happy with:



<pagebreak></pagebreak>

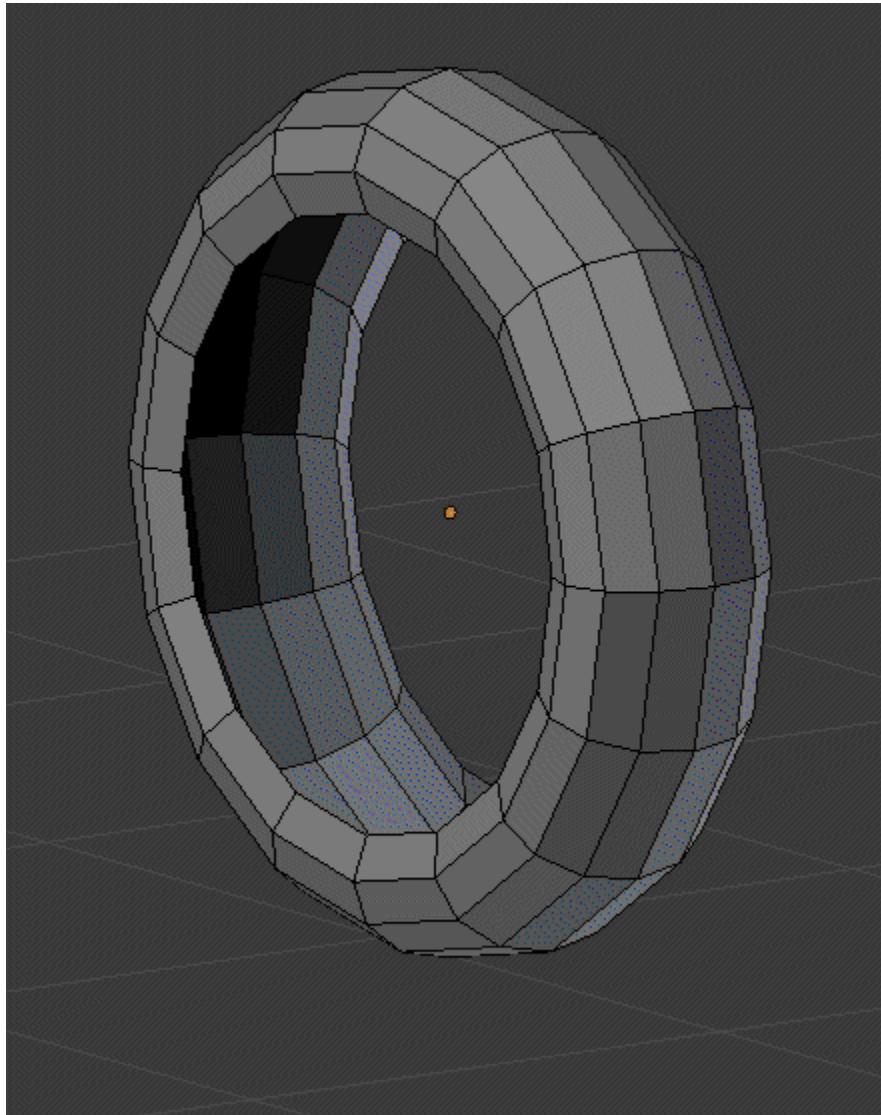
Finishing the robot

We'll add a little detail to the arm pieces now. First, I'll make sure that all my vertices are either connected or sitting slightly above the mesh they connect to:



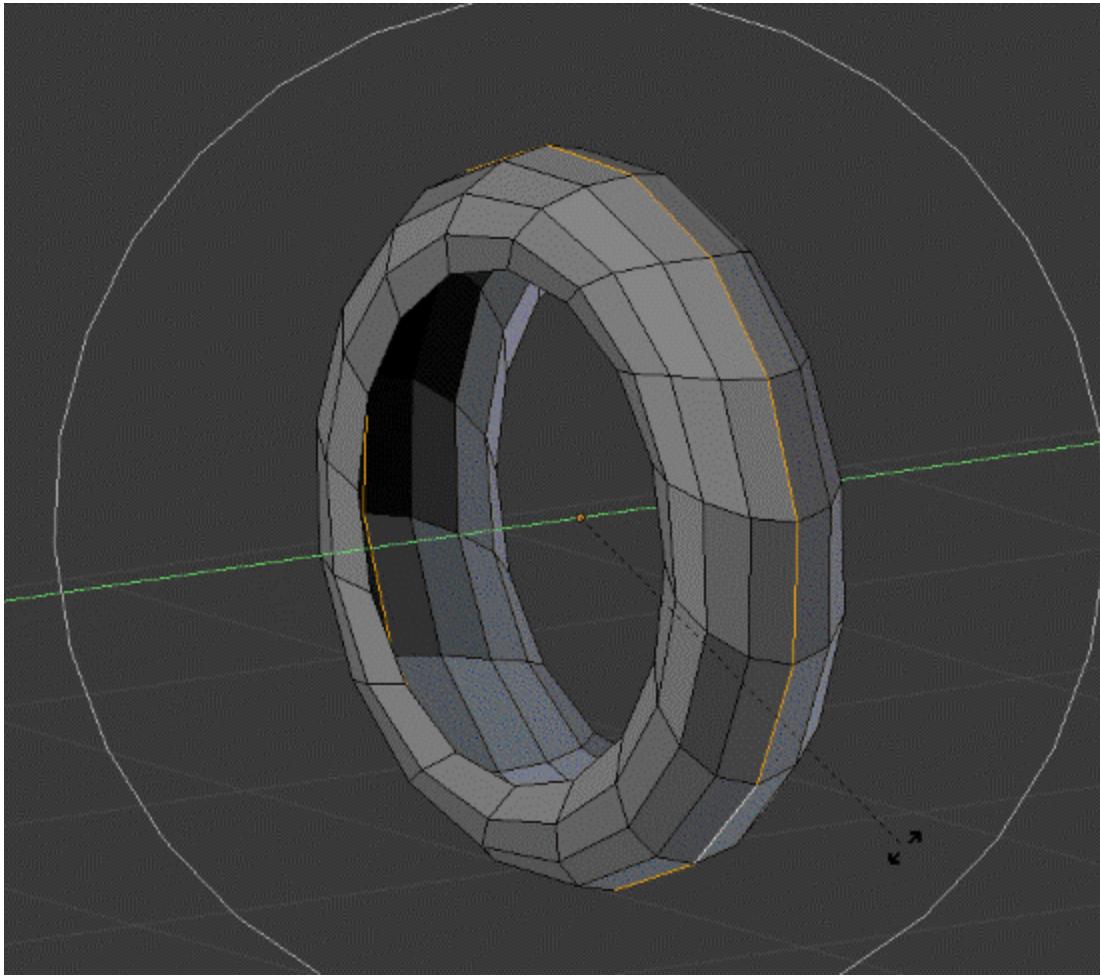
<pagebreak></pagebreak>

For the wheel at the bottom, we'll use **Subsurf** to create a sort of futuristic tire. We'll start simple:



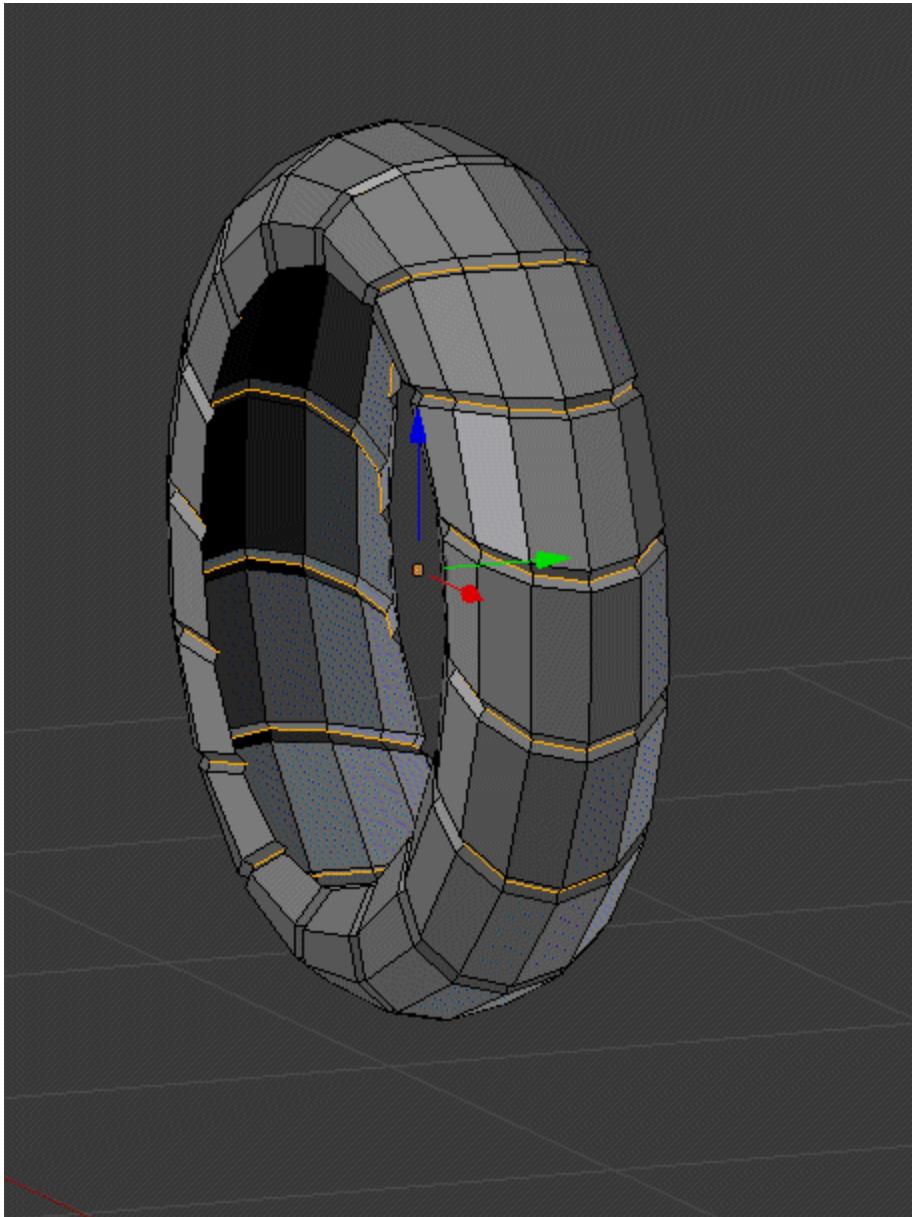
Remember, **Subsurf** will divide and smooth your mesh, so don't use too many sides to your circle originally.

Using proportional editing, I'll spin the central edge loop around the circle to create sort of an "arrow" effect to the tire:



<pagebreak></pagebreak>

Then, using the **Bevel** tool, I'll create some basic tread:

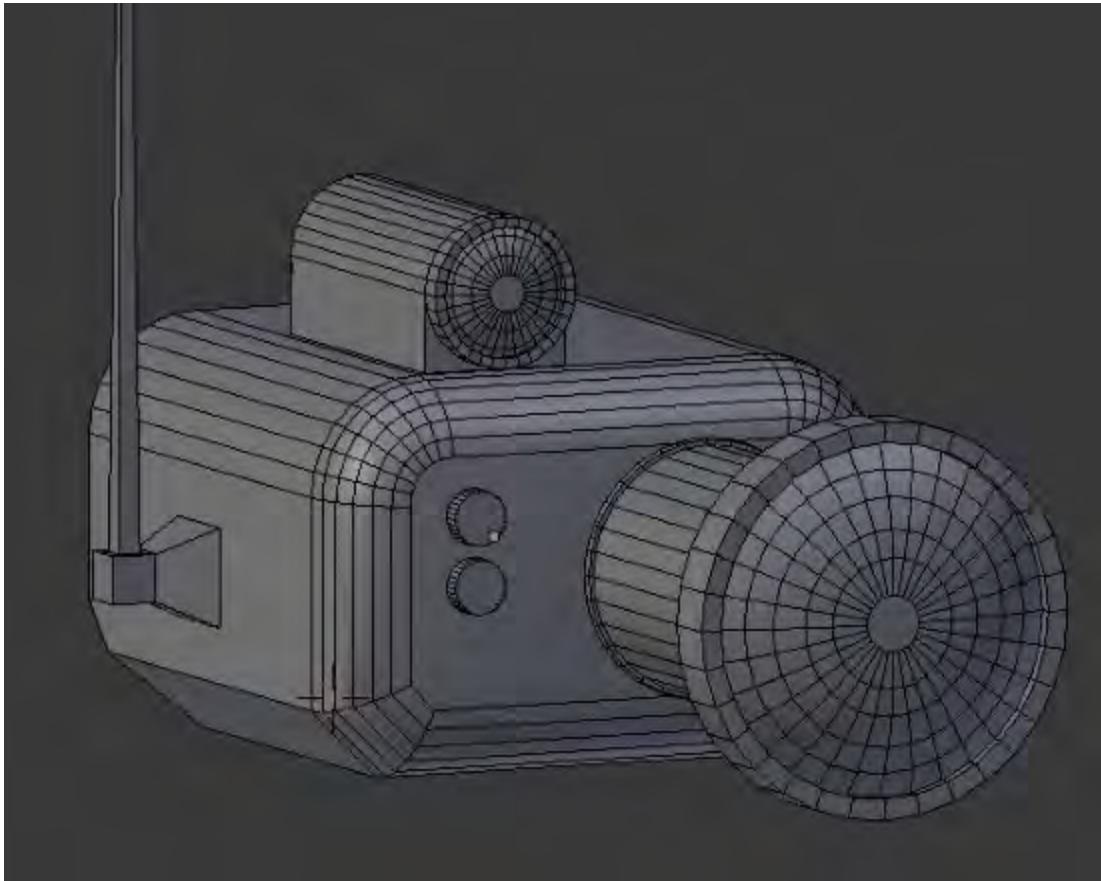


Then, I'll add **Subsurf** and create a basic wheel:

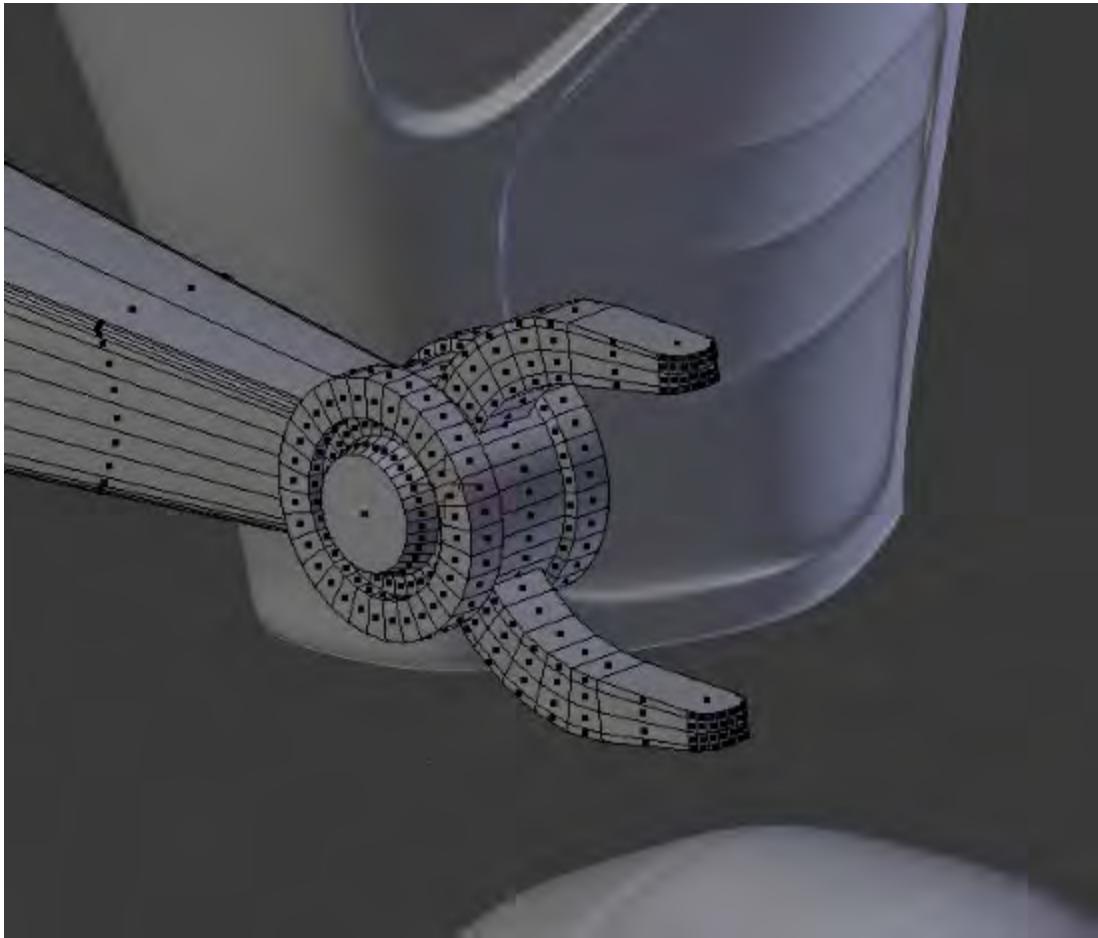


<pagebreak></pagebreak>

Now, you can go through and add some details if you'd like. I'll just do a little bit with the head here:



And add some hands:



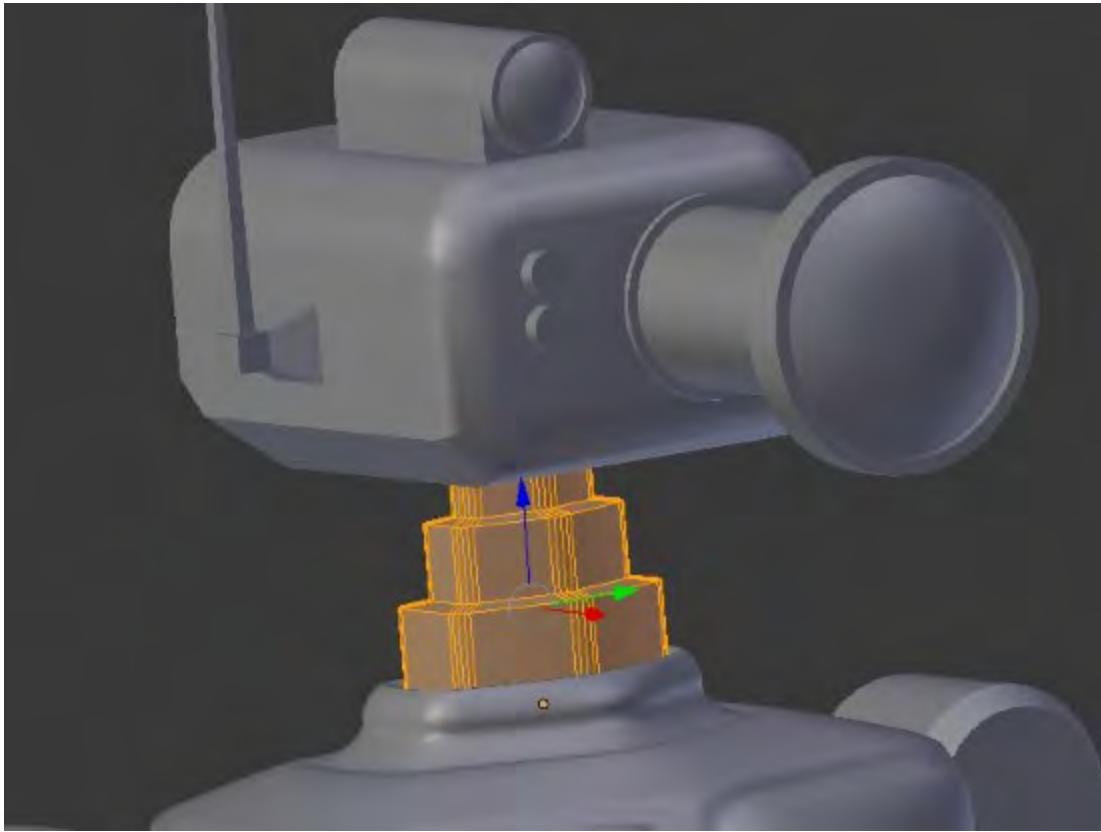
I'm not adding a lot of detail as I do this. When you're going for more of a cartoon or NPR look, that can sometimes be counterproductive.

<pagebreak></pagebreak>

Note

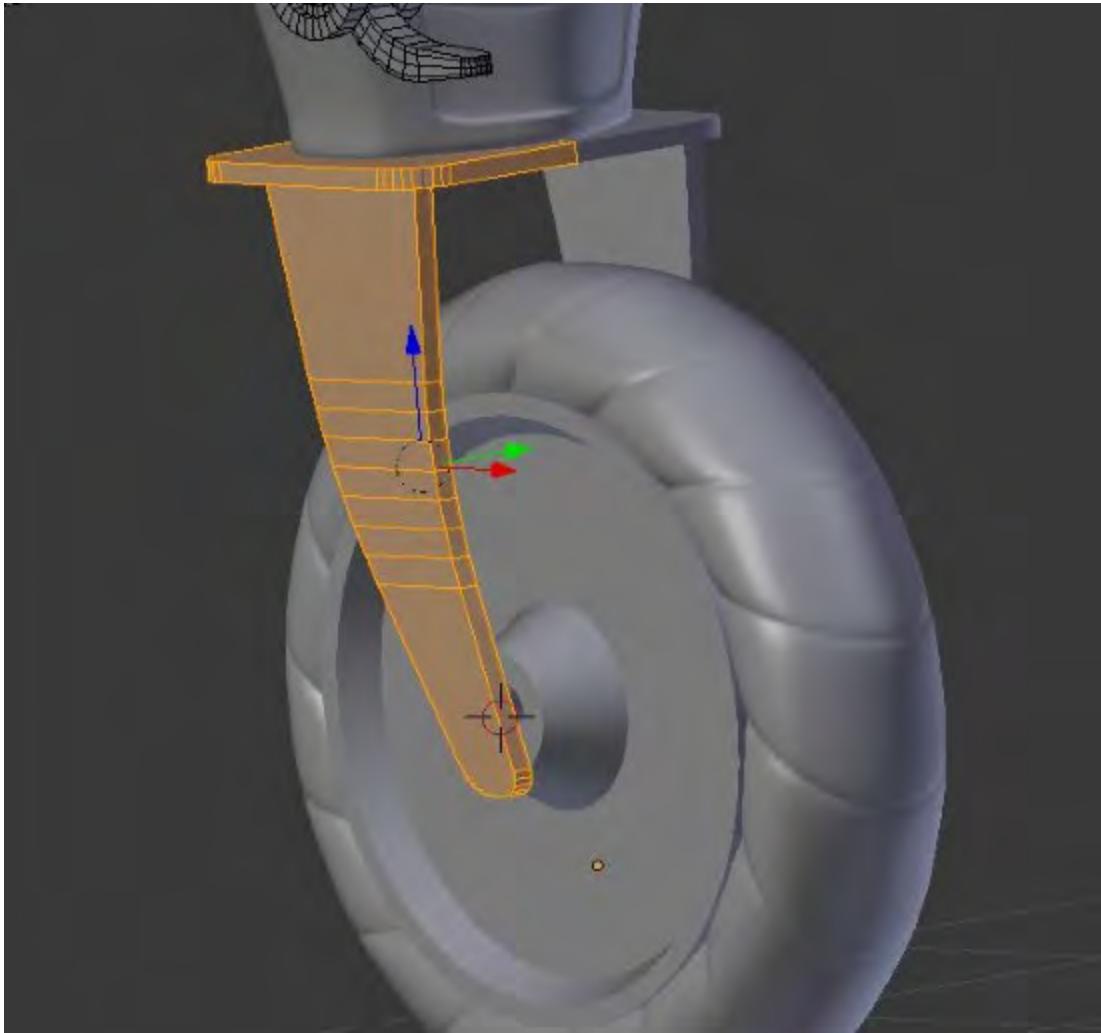
In previous chapters, we've been pretty careful about our use of polygons. We've tried to use as few as possible because there's a multiplying effect. If you have 100 extra polygons that you don't need on some object, and then you have 100 copies of that object, you'll be stuck with 10,000 extra polygons. In this case, however, we're keeping the robot pretty simple. We know the final model will be well within Blender's ability to render it, and a fraction of what our last project was. Therefore, polygon efficiency isn't quite so important this time.

Using previous techniques, we'll fill in the neck:



<pagebreak></pagebreak>

And then we'll connect the body to the wheel:



I'll just add a little bit of detail as I finish connecting the parts together. Again, we'll try to keep it pretty simple.

Here's what I ended up with:



<pagebreak></pagebreak>

Summary

In this chapter, we looked at some unique aspects of modeling for Freestyle. There are a number of techniques that will work for other Blender projects that we can't use in Freestyle. There are also methods that only make sense in Freestyle. We looked at these different methods, then went on to build a basic robot for non-photorealistic rendering. In the next chapter, we'll add materials and lights, then explore the various Freestyle options to produce our final result.

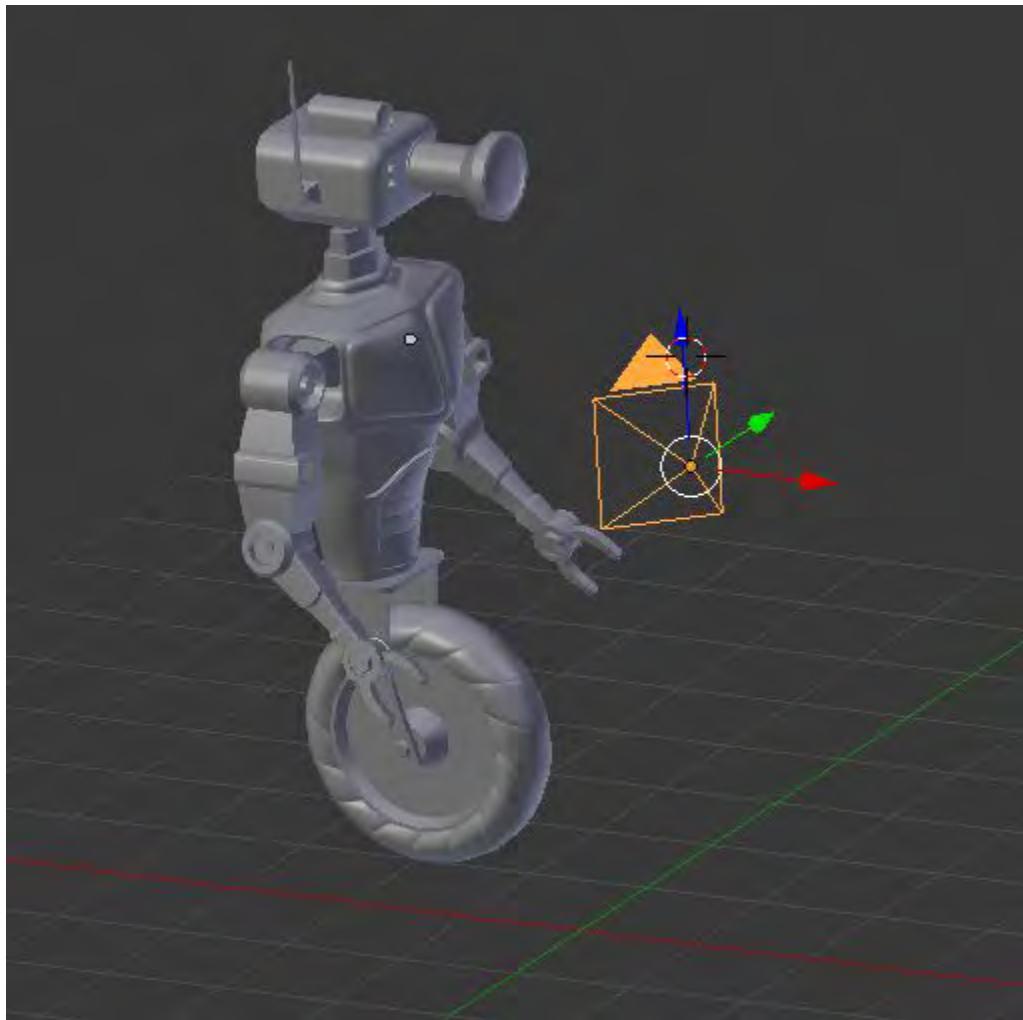
Chapter 8. Robot - Freestyle Rendering

In this chapter, we'll use Freestyle and the Blender Internal rendering engine to give our robot a stylized, cartoon look. We'll explore a few different rendering and material options and alternative possibilities:

- Preparing the scene
- Marking Freestyle edges
- Creating materials
- Rendering and material options
- Conclusion

Preparing the scene

Before we render our robot, we'll need to set up the scene. First, we'll add a camera:



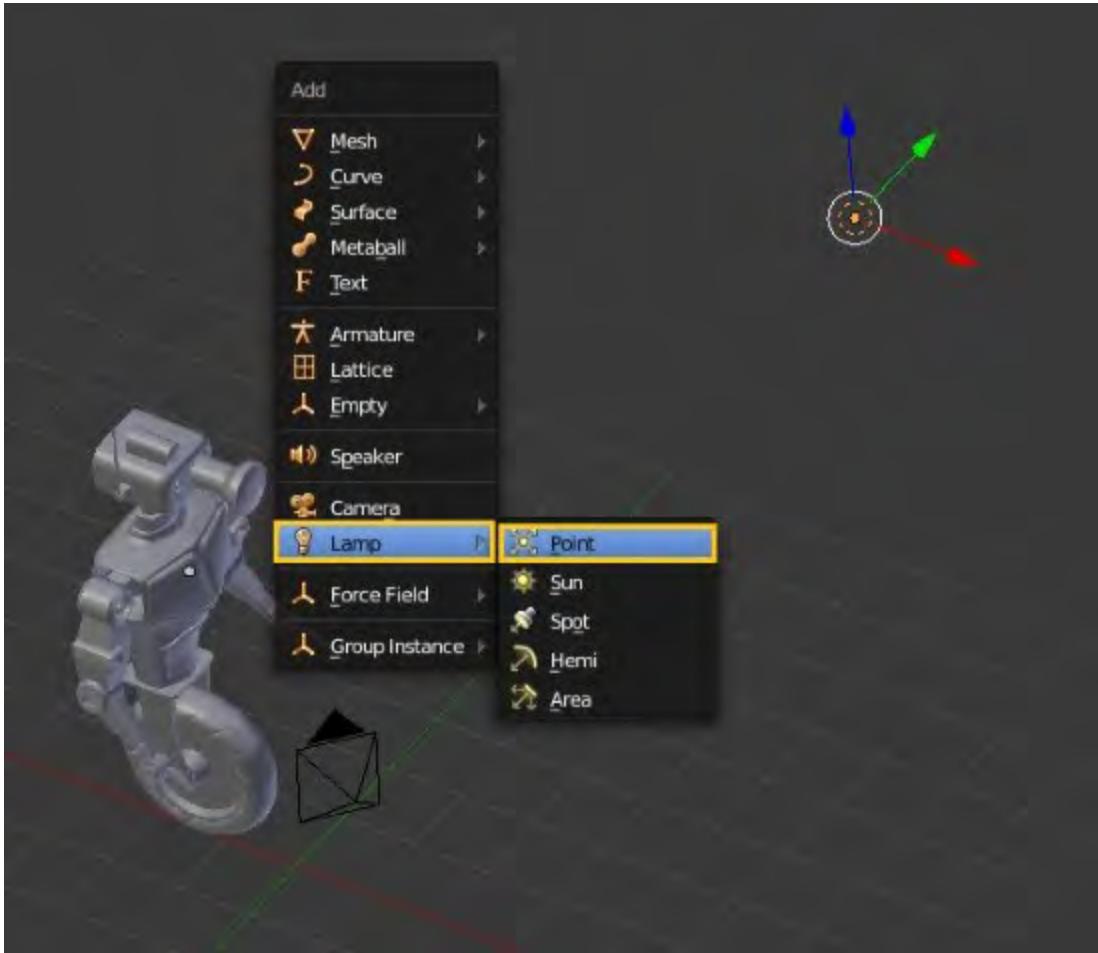
We'll position the camera for the render we'd like to create. If you're doing NPR renders, sometimes it helps to adjust your focal length down quite a bit:



This can help simulate the exaggerated proportions you might find in a comic book or other NPR products.

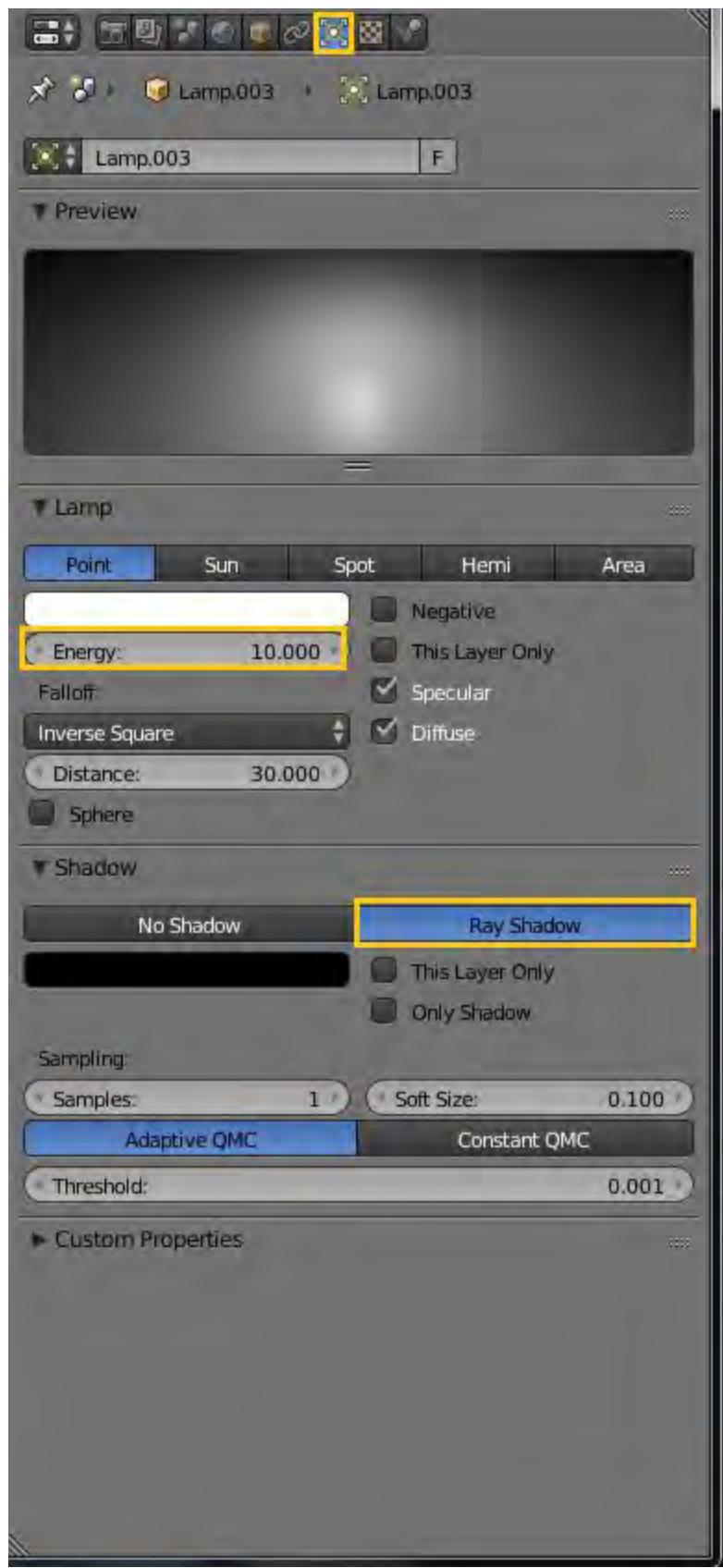
<pagebreak></pagebreak>

Next, we'll add a basic light to the scene. Unlike Cycles, Blender Internal only supports mesh lighting in a limited fashion. It's much better to use the lamps that are provided.



<pagebreak></pagebreak>

You can adjust the **Energy** of your light in the **Lamp** panel:



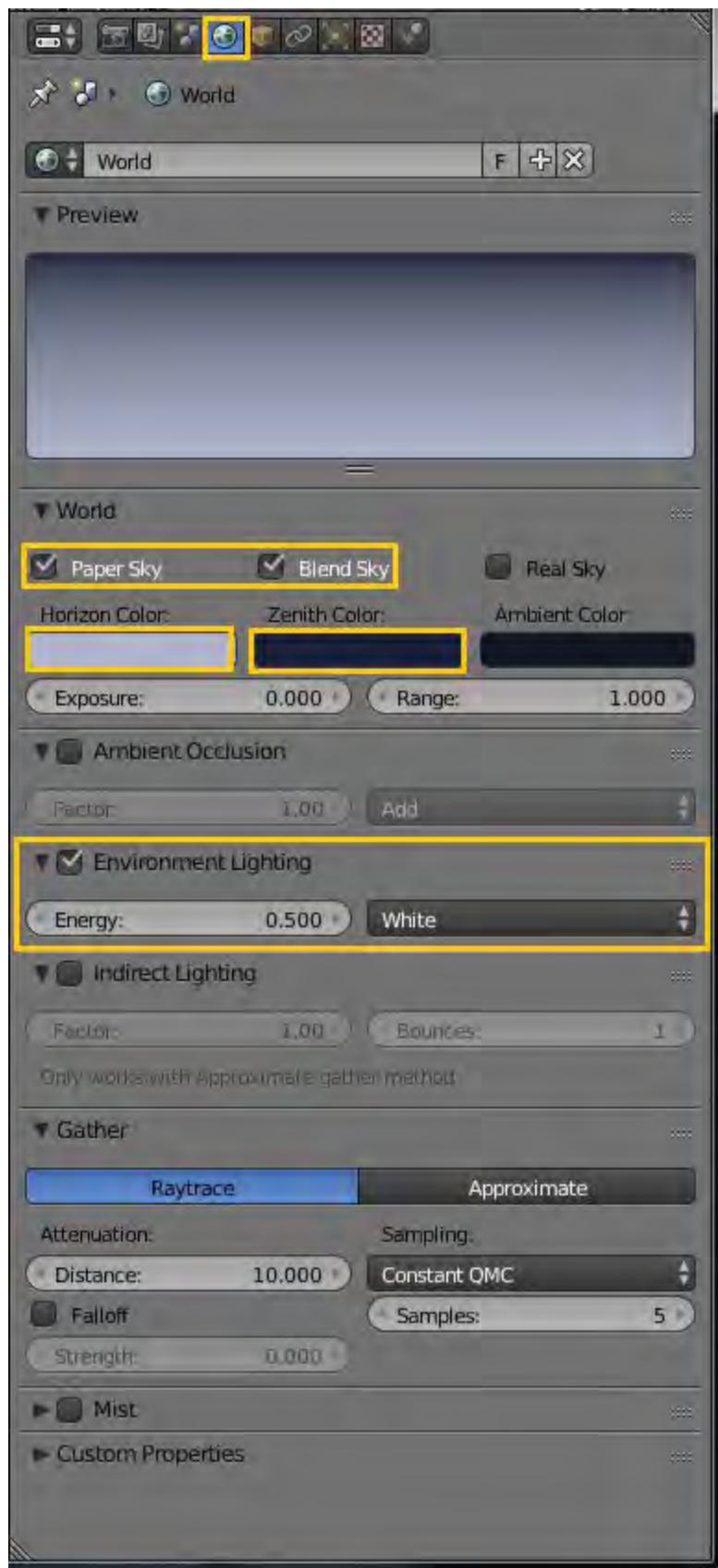
You can also choose either **No Shadow** or **Ray Shadow**. This is different than Cycles, but pretty self-explanatory. You are just choosing whether or not you want your lamp to cast shadows.

Note

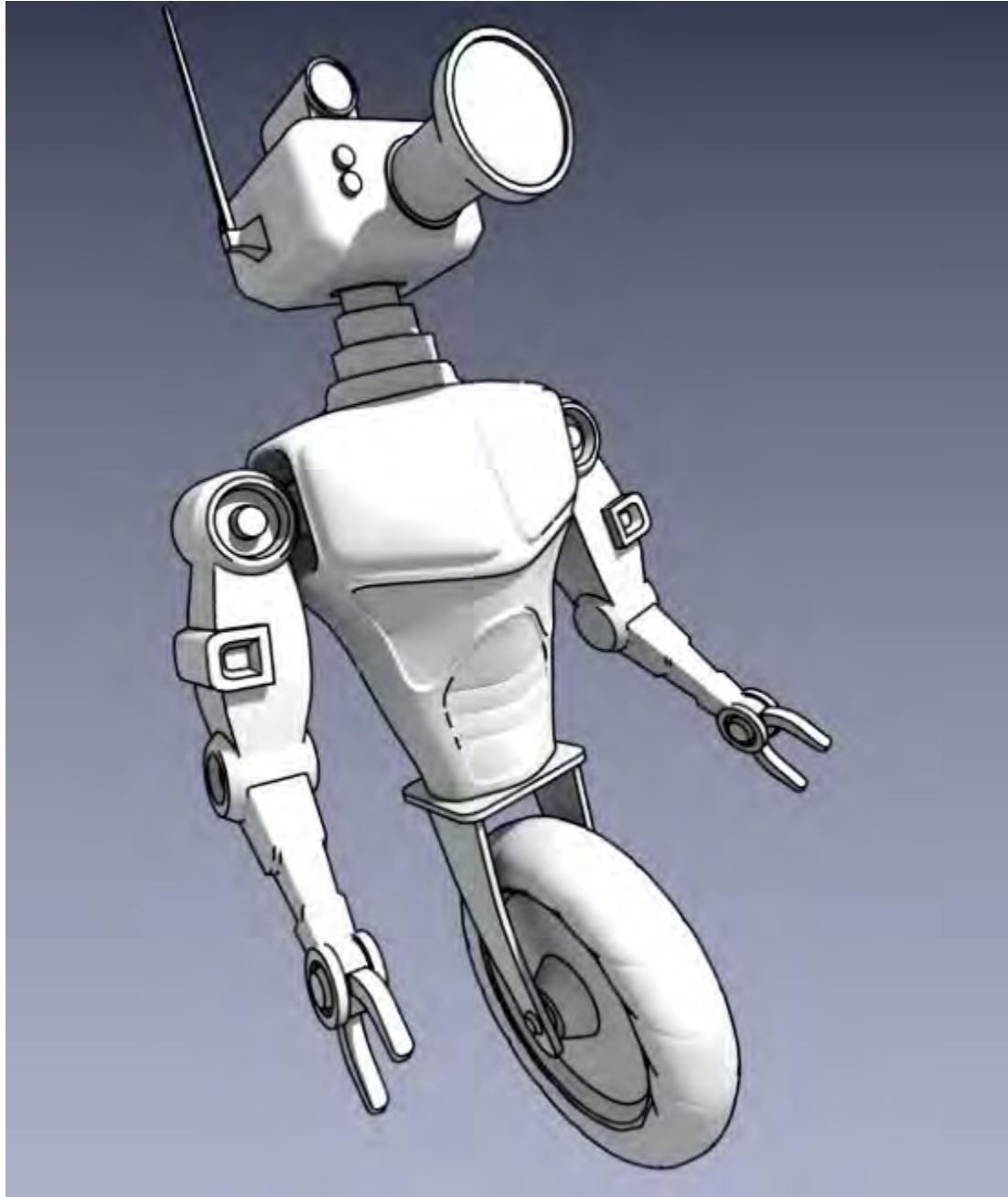
The term **Ray Shadow** is generally only used with Blender Internal. That's because you have the option (with other types of lamps) to choose something called **Buffer Shadow**. A Buffer Shadow is a non-raytraced shadow, which means it's an approximation not based on the actual calculation/reflection of light waves. It's very rare these days to use Buffer Shadow for anything.

Next, we'll move to our **World** tab and set things up here. For the sky, we'll pick two colors and select both **Paper Sky** and **Blend Sky**. This gives us a nice gradient for a background in our scene.

We'll also select **Environmental Lighting** and adjust the value to around **0.500**. This is just additional light that comes from the surrounding environment, rather than the single light we've added. It helps avoid very dark areas or harsh shadows. Sometimes you may want those things, but in this case, we don't.



<pagebreak></pagebreak>With these settings selected, let's try a Freestyle render of our robot now:

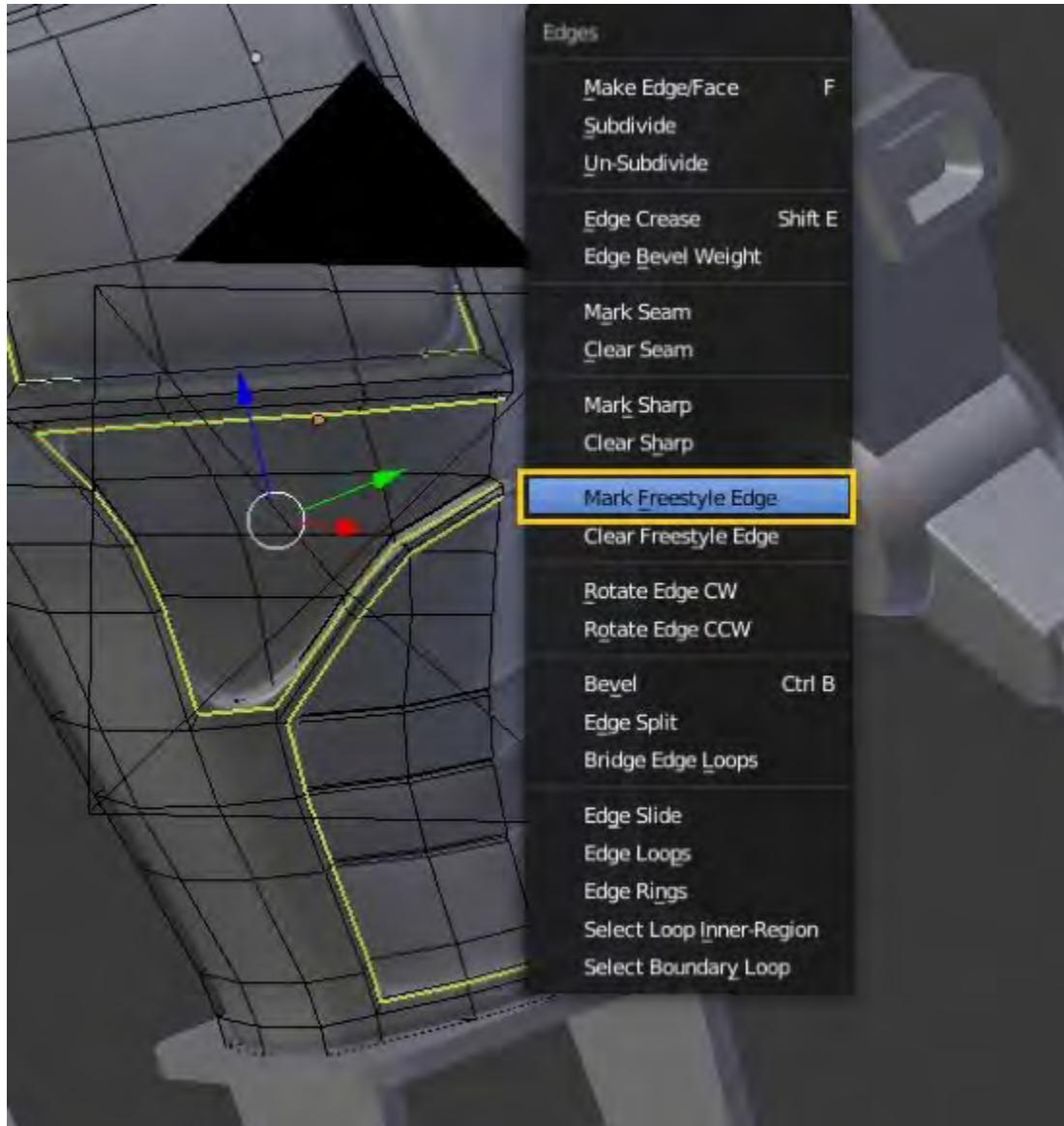


It looks okay, but obviously we've got a few problems. Despite the fact that we modeled it correctly (specifically for Freestyle), Blender still hasn't highlighted all the edges that we want it to. That's okay—there's an easy fix.

<pagebreak></pagebreak>

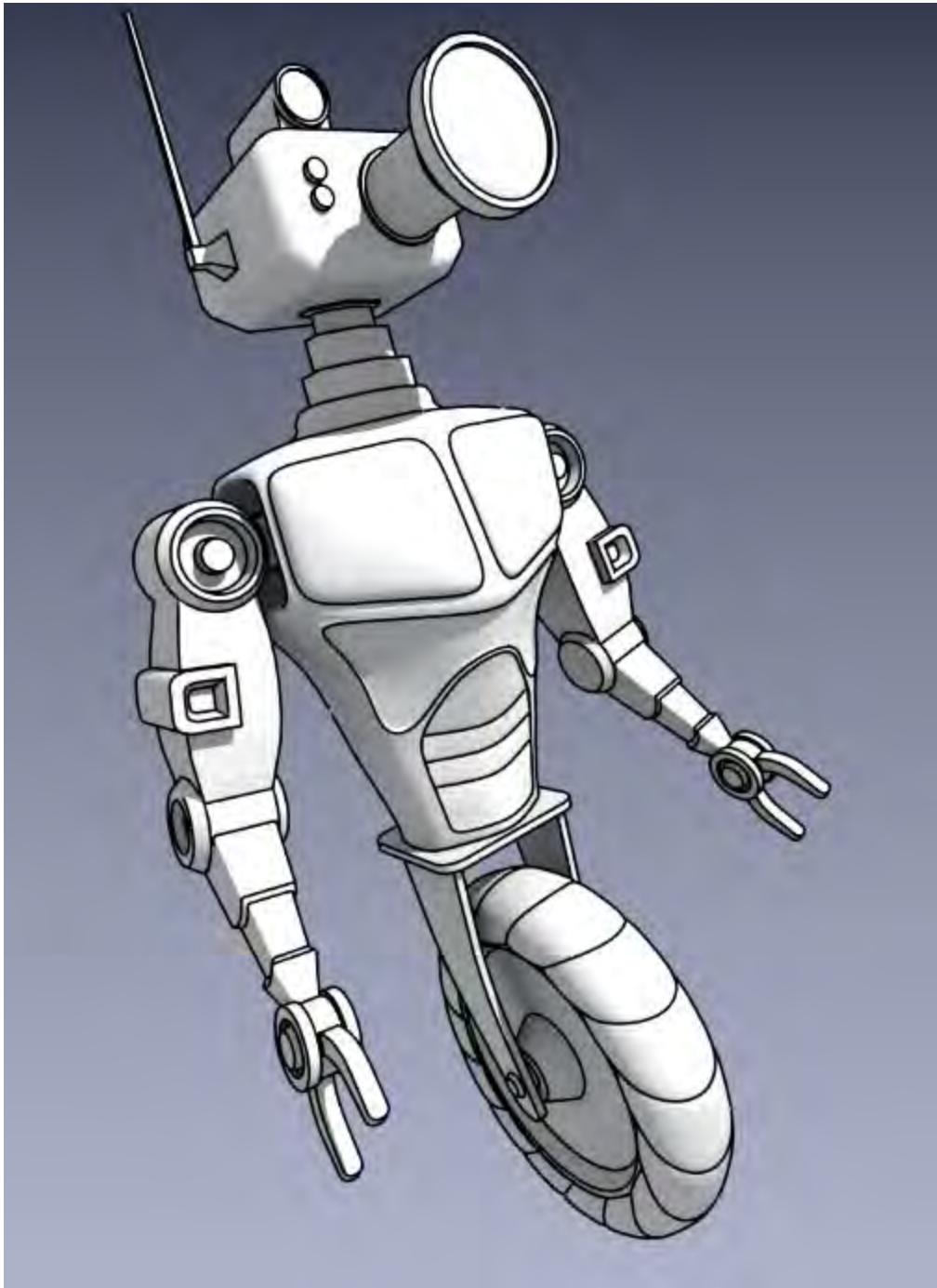
Marking Freestyle edges

When you want to make sure that a particular edge shows up in Freestyle, the easiest way to do it is to just select that edge in **Edit Mode** and pick **Mark Freestyle Edge**:



<pagebreak></pagebreak>

Now, Blender will always render that edge (when it's not obstructed by another object). So, let's go through and mark all the edges that we want in our render. When we're finished, it looks a whole lot better:



Now we have a decent Freestyle setup, although we'll play with a few options later. The next thing we'll need to do is create some materials for our robot.

Creating materials

Materials in Blender Internal work a lot differently than they do in Cycles. In some ways, it's easier and more straightforward.

Cycles gives you a lot of preset shaders (**Glass**, **Glossy**, **Diffuse**, and **Emission**) and more. Blender Internal doesn't work that way. Instead, you manually configure the properties for each shader. You select your diffuse color, specular color, specular intensity (shininess), hardness of specularity (how smoothly the shiny parts blend into the rest of the material), and more.

<pagebreak></pagebreak>

You also have a number of different shading options. Since we're going for a cartoon look to our robot, we'll pick the **Toon** shader for both our **Diffuse** and **Specular** (glossy) shaders. You can adjust the various settings until the preview material looks approximately the way you'd like:



<pagebreak></pagebreak>

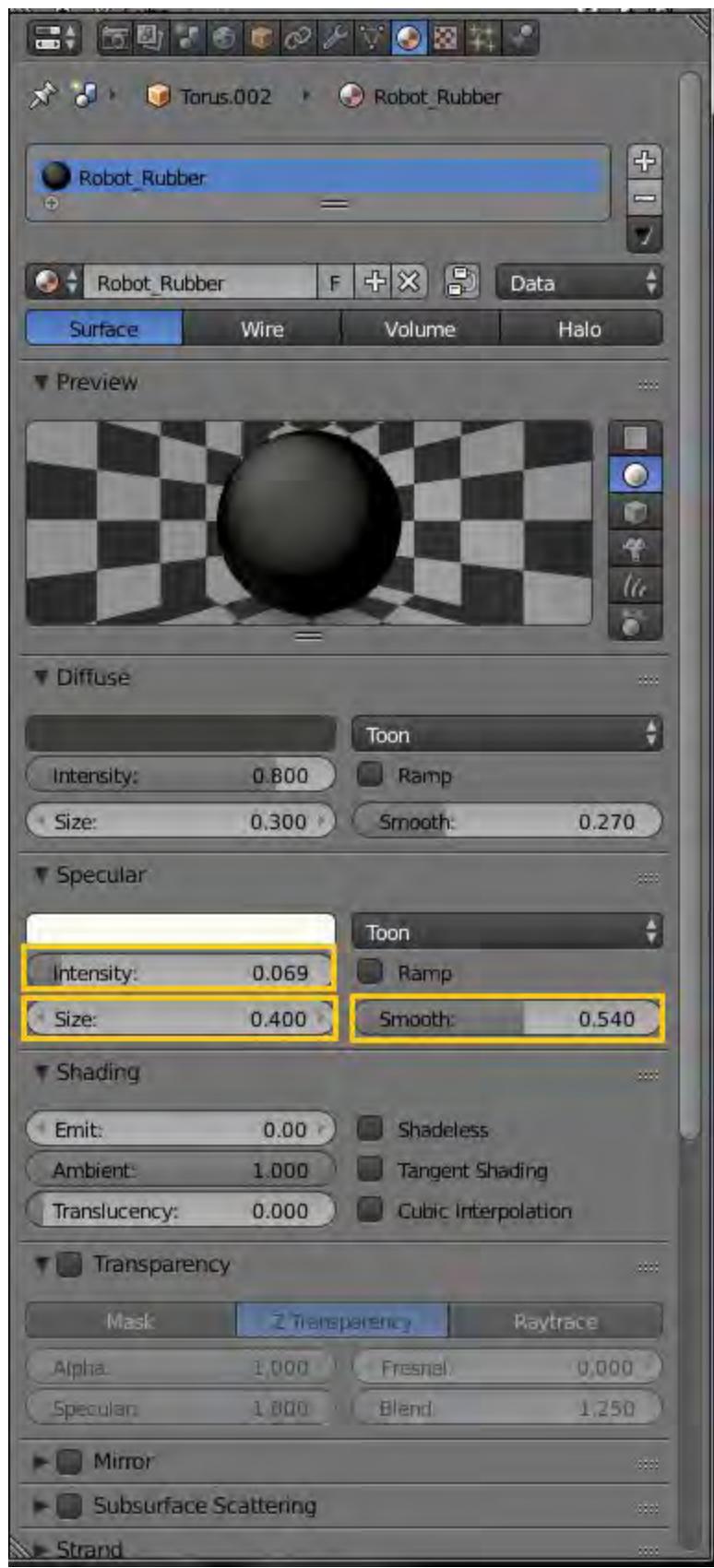
When you've got a decent setup, go ahead and try a quick render:



One thing you may notice here is that the scene renders significantly faster than it did in Cycles. We're not getting the added realism that comes with the Cycles rendering engine (because we don't want it), and as a side effect, it takes Blender much less time to produce an image.

<pagebreak></pagebreak>

At this point, you can adjust your various material settings the way you'd like to. When you're happy with the orange paint (or whatever color you're using), let's create a second material for rubber:



<pagebreak></pagebreak>

I've adjusted the settings here for a less shiny (and darker) material. We'll then go through and apply that material to the appropriate parts of the robot:



<pagebreak></pagebreak>

That's starting to look better. I'll also create a metal material to highlight a few parts:



Finally, we need a glass material for the two camera lenses. Note that this isn't actually glass (in the Cycles sense), since it has no reflection or transparency. That's what we want for an NPR render though, particularly since there are no other objects in the scene (and thus, the glass would have nothing to

reflect). It's also good that we don't have transparency, since we haven't modeled any internal components of the camera lens.

<pagebreak></pagebreak>

These are areas where you can save a lot of time when you're doing NPR work. Here's what I did for a cartoon glass material:



<pagebreak></pagebreak>

After we've assigned our new materials, let's render it again:



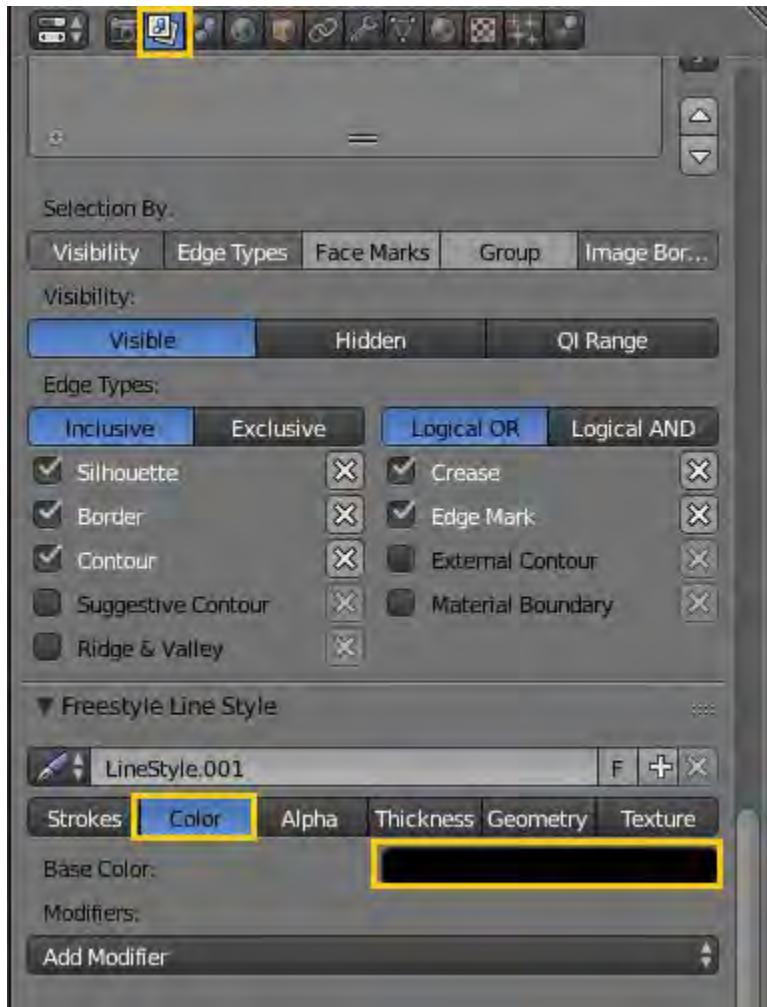
That looks pretty decent for a cartoon robot.

Next, let's take a look at a few options that we have.

<pagebreak></pagebreak>

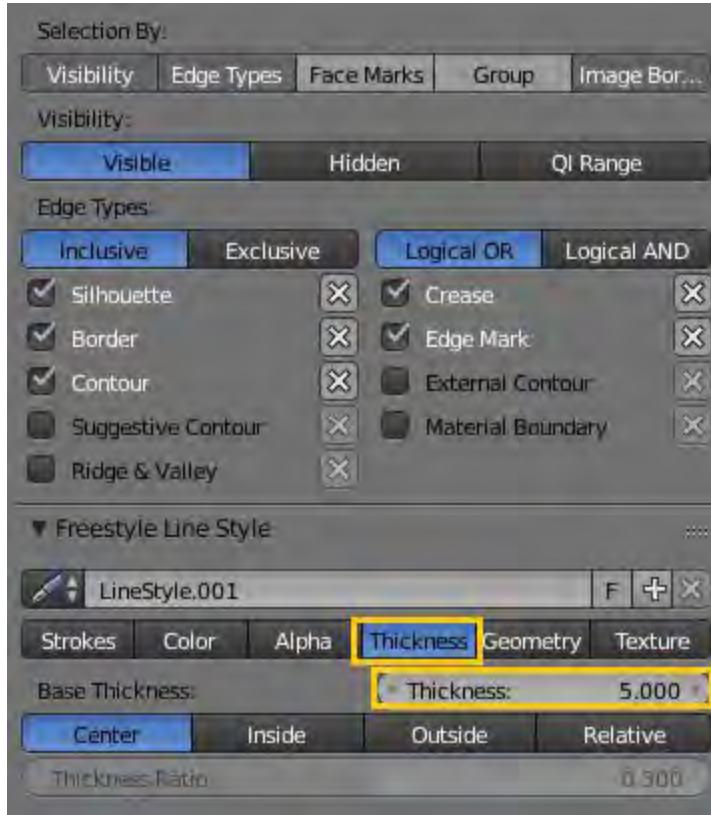
Rendering and material options

In the **Layers** tab, let's take a look at our **Freestyle** options. One thing we can easily adjust is the color of our lines here:



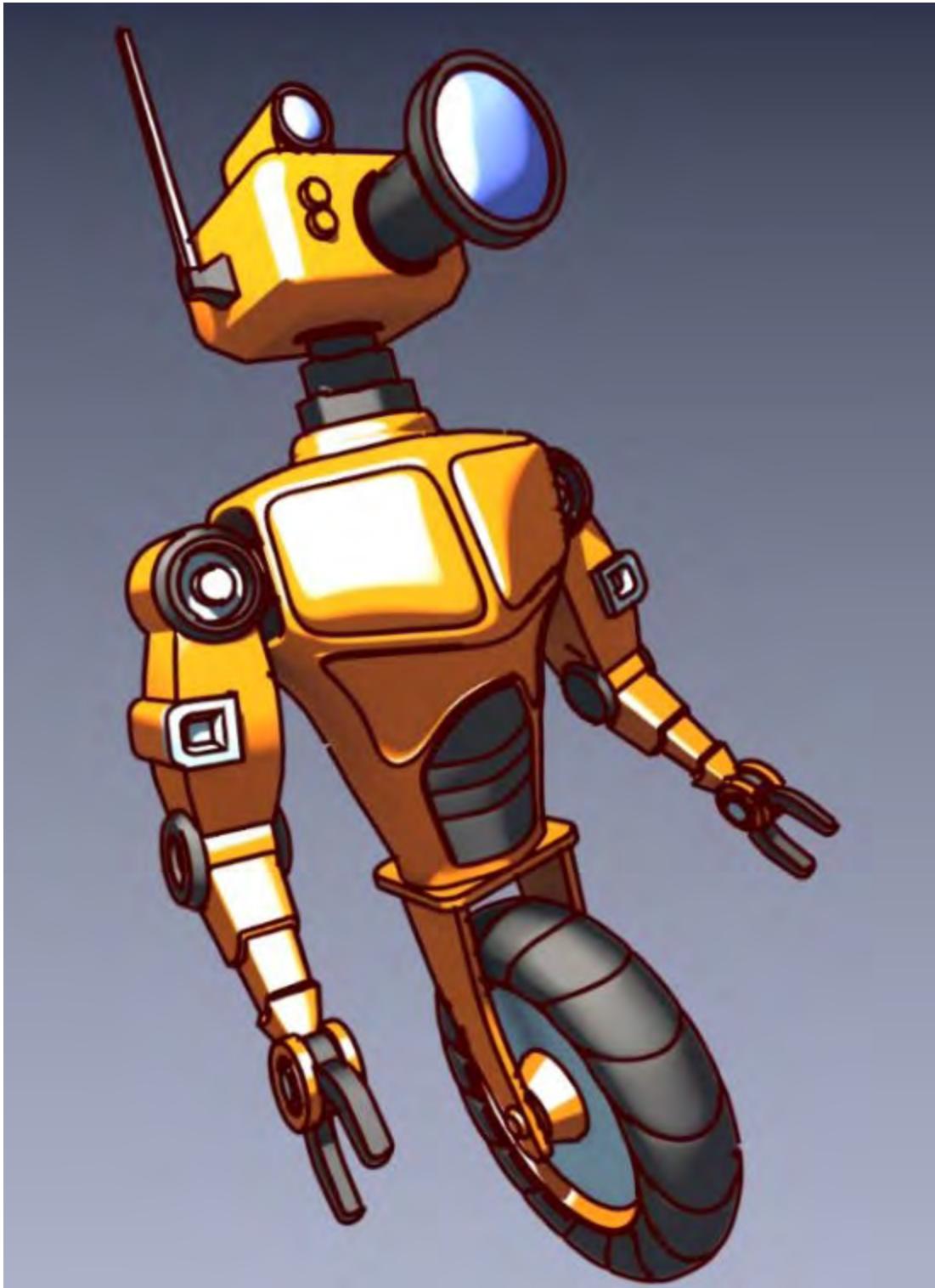
<pagebreak></pagebreak>

Black is more traditional for cartoons, but you can certainly experiment. Another thing we can easily adjust is the thickness of our lines:



<pagebreak></pagebreak>

When we render with thicker red lines, here's what it looks like:



<pagebreak></pagebreak>

You can also change the types of Freestyle edges that Blender will highlight for you. For example, we'll highlight the **Suggestive Contour** and **Ridge & Valley** options:



<pagebreak></pagebreak>

As you can see, this has quite an interesting effect on our render:



For our robot, it just looks messy—I don't think we'll use it. But if you were doing some type of landscape or other smooth, flowing shape, it might help draw out a bit of detail.

<pagebreak></pagebreak>

Another option you have is to use the **Ramp** feature of the Blender Internal materials. This is an easy way for a material to fade from one color to another:



<pagebreak></pagebreak>

It can be used to simulate color changing paint, glass, or anything else where the color varies with the angle. The setup shown is pretty basic, but I think it makes the robot look a little bit better:

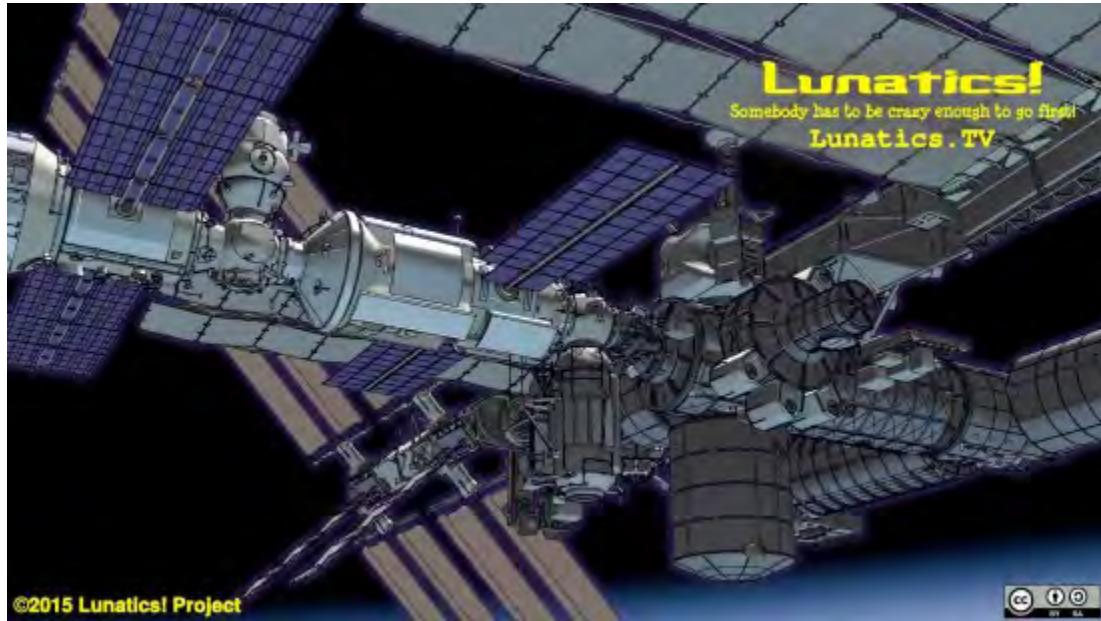


Feel free to experiment with the various settings until you've got a material you're happy with.

That's about as far as we're going to take this particular project. To be honest, we've barely scratched the surface of what Freestyle can do. It's a very broad topic, and could easily fill an entire book on its own.

<pagebreak></pagebreak>

Just as an example of what you can do, here's a Freestyle render of the International Space Station from the web series Lunatics! (, reprinted with permission):



As you can see, there are multiple different sets of Freestyle lines here, creating a sort of "ink effect" to the render. You can also create sketch drawings, blueprints, and multiple other rendering effects. Hopefully, what we've covered will serve as a good introduction if you want to explore the topic further.

<pagebreak></pagebreak>

Summary

In this chapter, we created a basic material and rendering setup for Freestyle. We looked at a few of the different options and how they affect your render. We also briefly explored the material and rendering differences between Cycles and Blender Internal (which can be an advantage when doing NPR renders).

In the next chapter, we'll look at another use of Blender—creating game models.

Chapter 9. Low-Poly Racer – Building the Mesh

In this chapter, we'll build a low-poly model for use in a racing game. We'll look at several ways that game models differ from the projects we've created so far. We'll also look at some pros and cons of various methods, and important considerations when building these types of models.

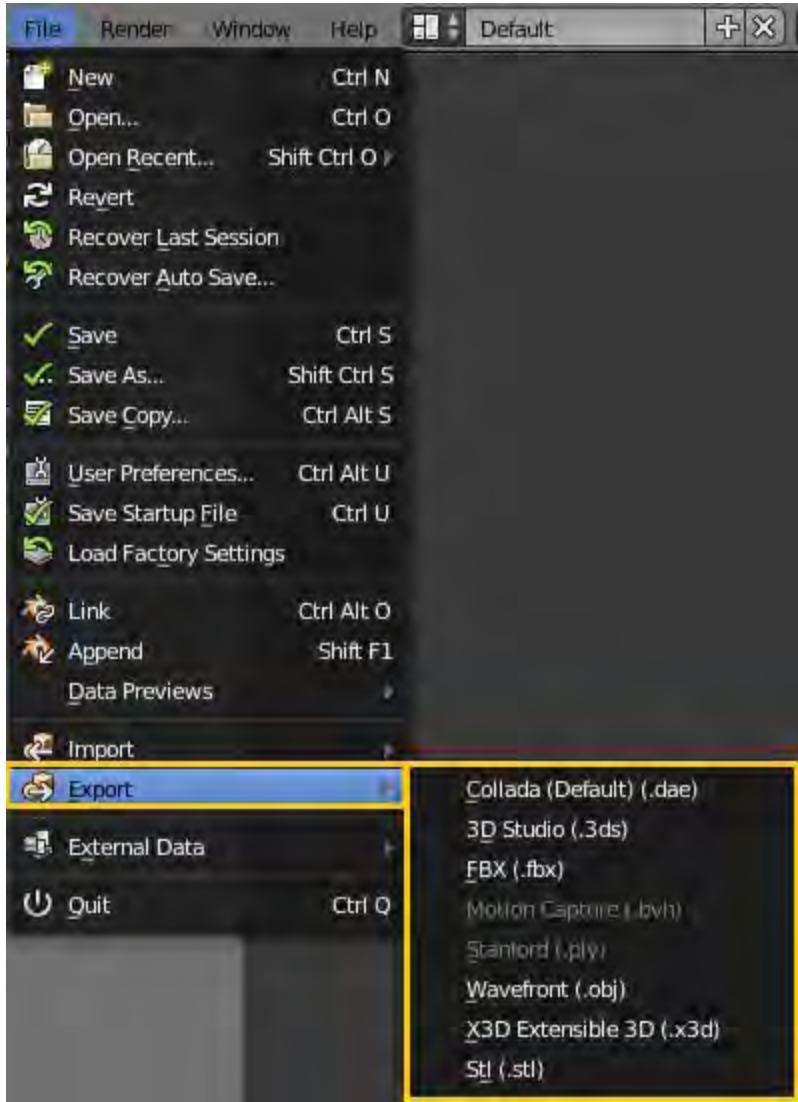
- Building for external applications
- Starting the truck model
- Manifold versus non-manifold meshes
- Adding details

Building for external applications

Up to this point, all the models we've done have been strictly for Blender use. In other words, we knew that all of the texturing, rendering, and animation would be done completely within the Blender software. However, sometimes you'll want to create 3D models for use in external programs.

<pagebreak></pagebreak>

For instance, Blender is an excellent tool for creating 3D game models. You can build and texture the model in Blender and then export it to other formats. Let's take a quick look at our export menu to see what options we have:

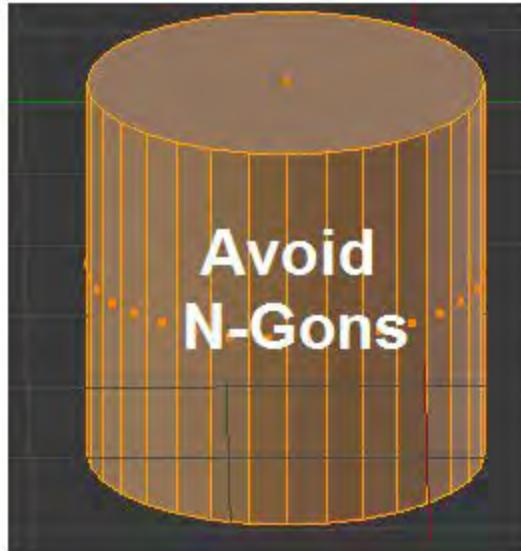
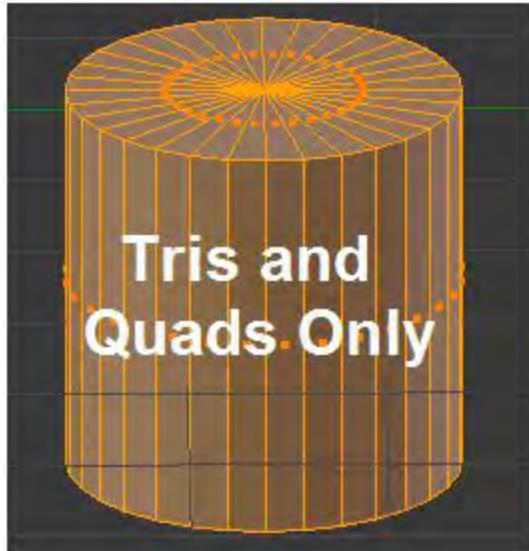


These are the formats that Blender can directly export to (and there are more that can be activated with add-ons). Additionally, there are other methods (some online) that can be used to convert one model format to another. Many game and simulation platforms also have import functions, allowing you to bring in a model from another format.

When you're talking about export and conversion, it's important to note that not everything carries over from Blender. You'll want to create your models in certain ways to prevent data loss and maximize compatibility.

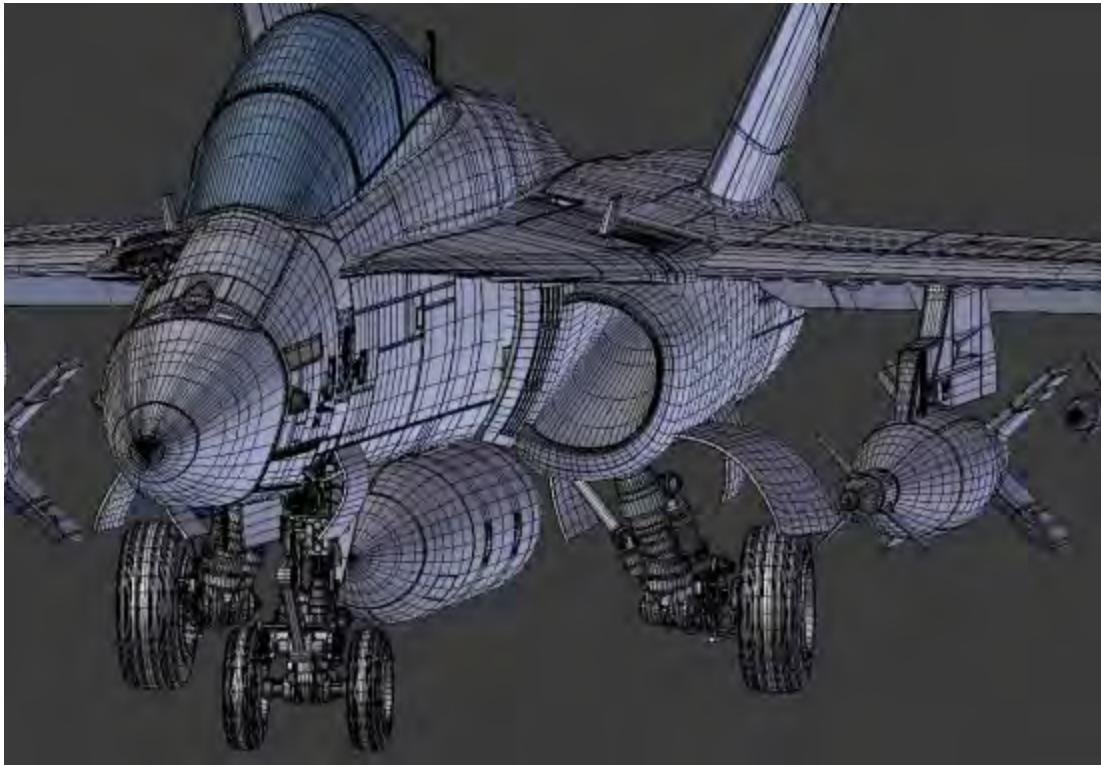
<pagebreak></pagebreak>

Mesh data is (almost) universal, which means that the physical structure of a model should be usable in most 3D and game programs. One thing to bear in mind, however, is that N-Gons are not universal. They are increasingly common, but there are still several popular platforms that don't accept them. So when you're building for export, I highly encourage you to use only Tris and Quads:



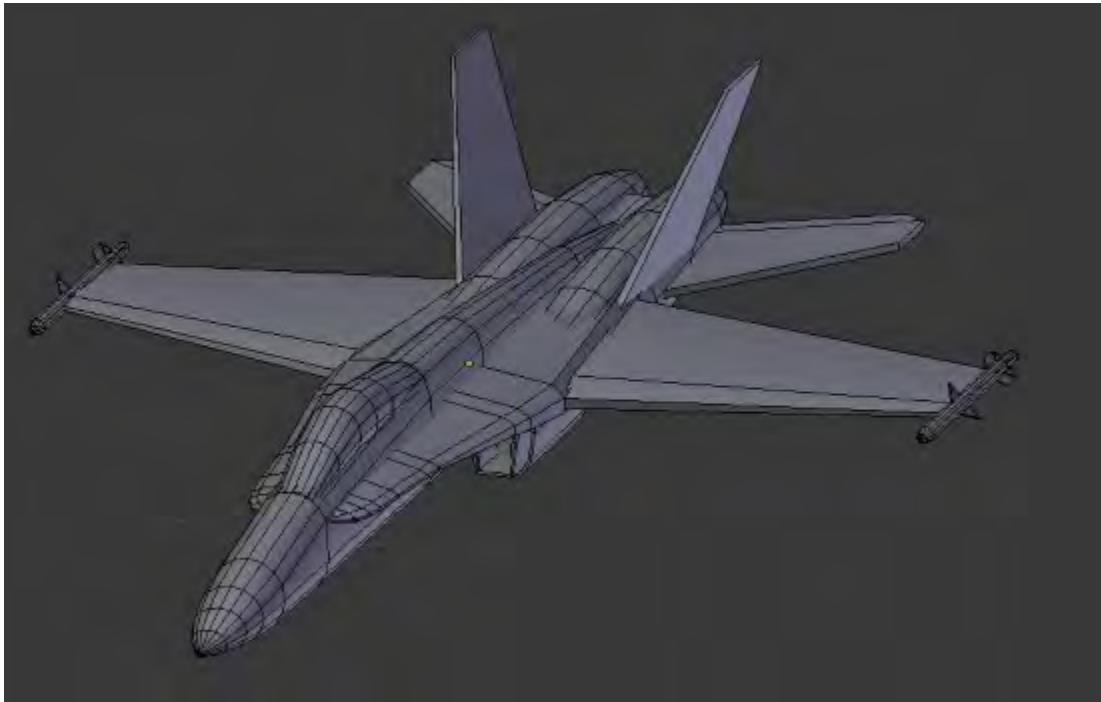
The polygon count is very important for these types of models as well. If you're familiar with flight simulators, you'll know that a typical aircraft looks far less realistic in a game than it does in a Hollywood movie. Both are 3D models, but they have very different requirements and uses.

Let's take a look at two models I did of an F-18 Hornet. The first is for 2D rendering only (or non-real-time animation). Since I could afford to spend minutes (or hours) on rendering, I was able to model a lot of complex detail into the mesh itself:



<pagebreak></pagebreak>

By contrast, here's another F-18 model I did for a simulation:

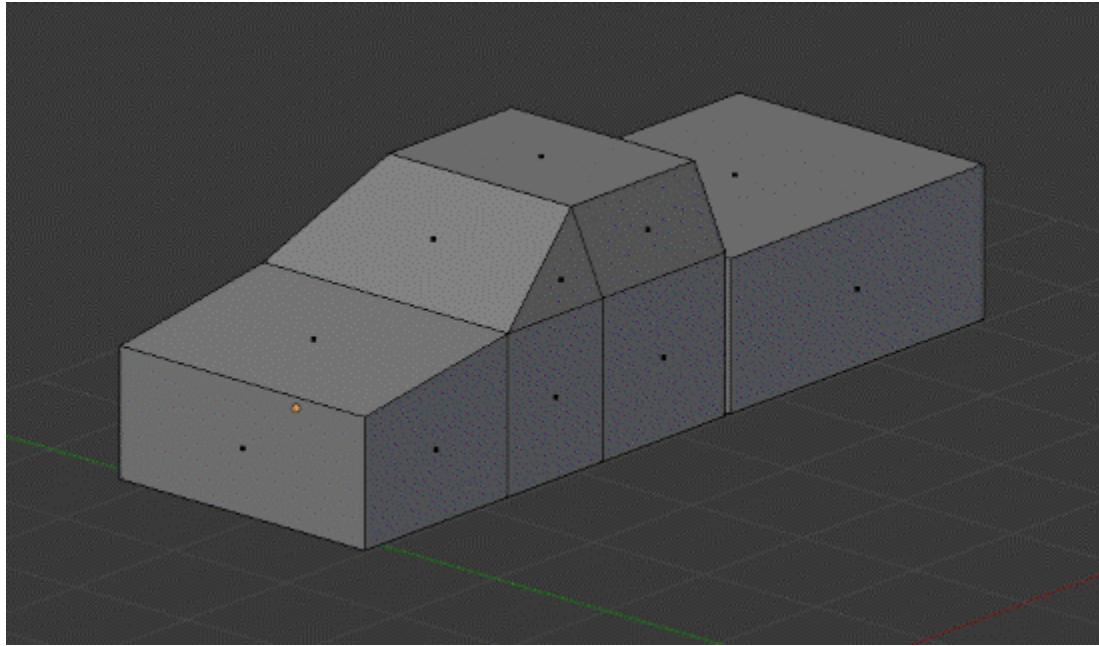


You can see that it's far less detailed because it needs to be rendered in real time (in other words, it's constantly rendered as the user "flies" the airplane). So as we build our model, we'll need to keep the polygon count as low as possible.

How low does it need to be? Well, that really depends on the application. Some truck models (like the kind we'll be making) may only be few dozen polygons, while others can reach into the tens of thousands. It all depends on your application. For this project, we'll shoot for around 5,000 polygons (or about 10,000 triangles—more on that in a second). That should be usable for the majority of real-time render engines.

Starting the truck model

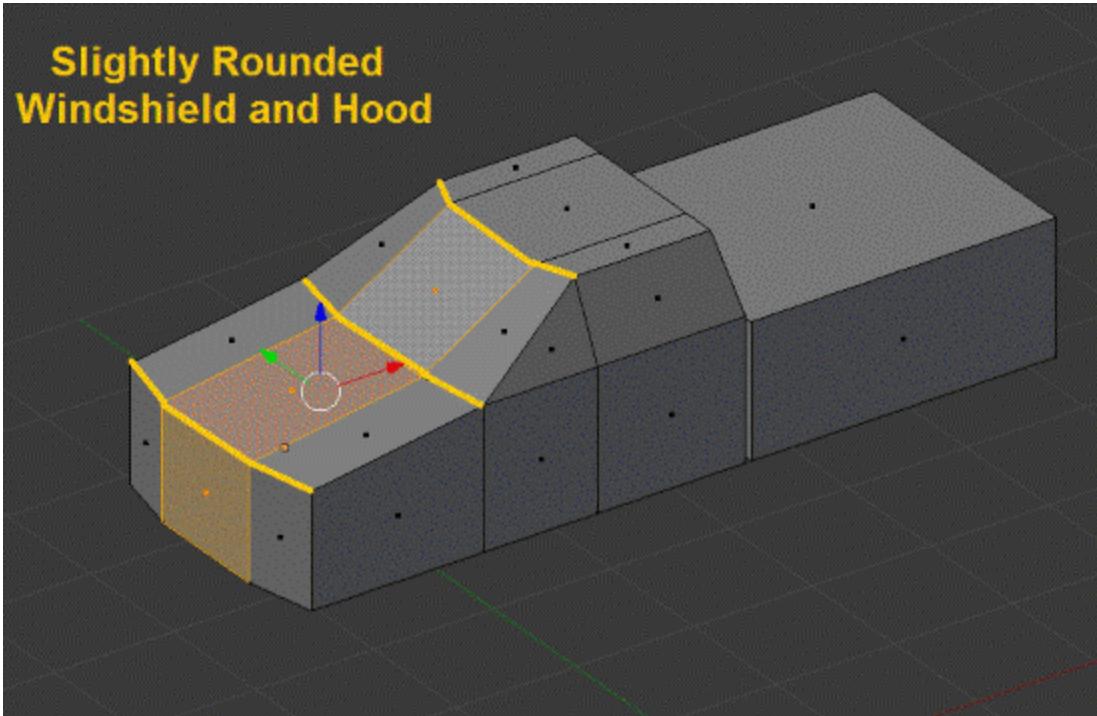
Let's start by creating the basic shape of our vehicle. We'll make this an off-road racing truck, so let's just block out the basic shapes. We'll start with the rough dimensions of the body:



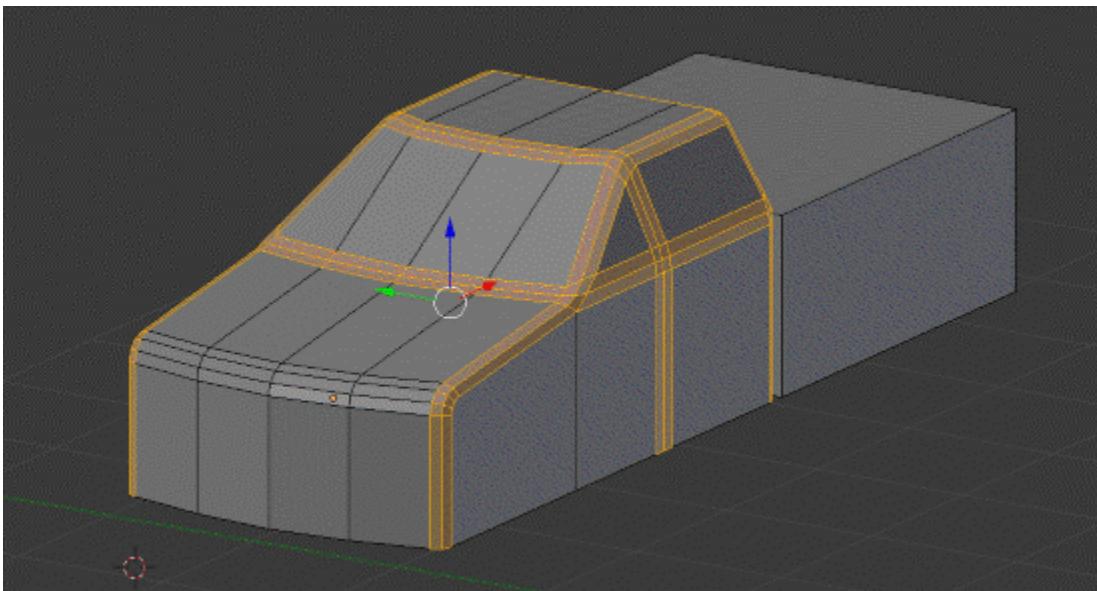
<pagebreak></pagebreak>

Then, I'll just run a couple of loop cuts around the cab and add a slight curve to the front of the truck:

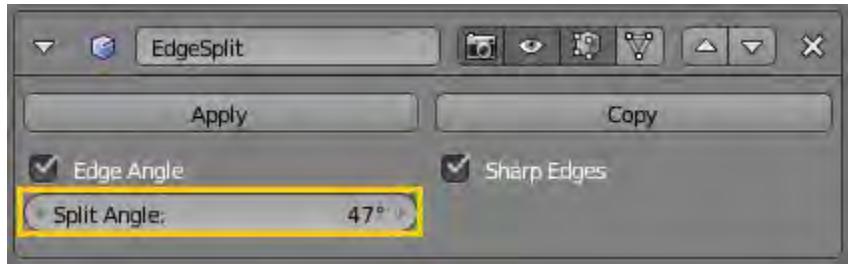
Slightly Rounded Windshield and Hood



Next, I'll bevel the edges very lightly. If we were going for maximum polygon savings, we probably wouldn't bevel them at all. However, beveling doesn't add too much geometry here, and it makes the vehicle look a lot nicer.



Next, I'll add an edge split modifier to the mesh. We'll use a split angle of 47 degrees or so. I wanted to make sure all the 45 degree angles were smooth, so I just added a couple of extra degrees in case there was a weird corner in the mesh somewhere.



<pagebreak></pagebreak>

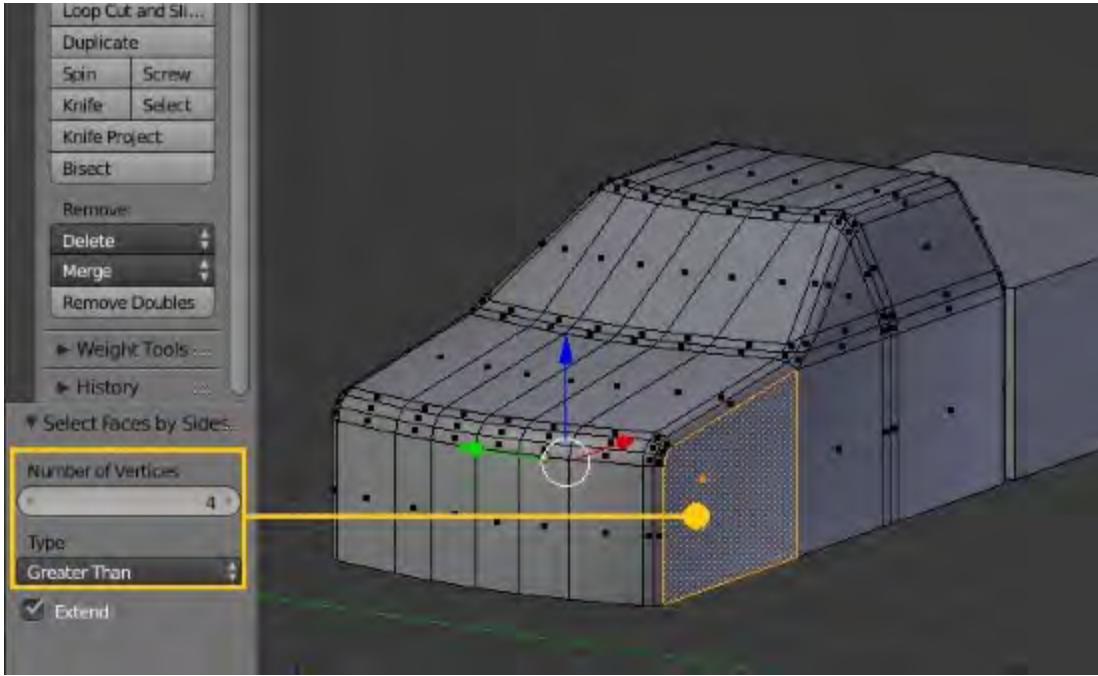
It's important to note that the **Edge Split** modifier is only temporary. Since the model is going to be exported, the smooth and flat shading will be determined by whatever game engine you end up using. It's still useful, however, as it can show us when corners are too sharp (or not sharp enough), and we can make corrections as we model.

Earlier, I mentioned that we don't want to use N-Gons. However, sometimes you end up with them by accident or force of habit. At any time, you can go to your **Select** menu and pick **Select Faces By Sides:**

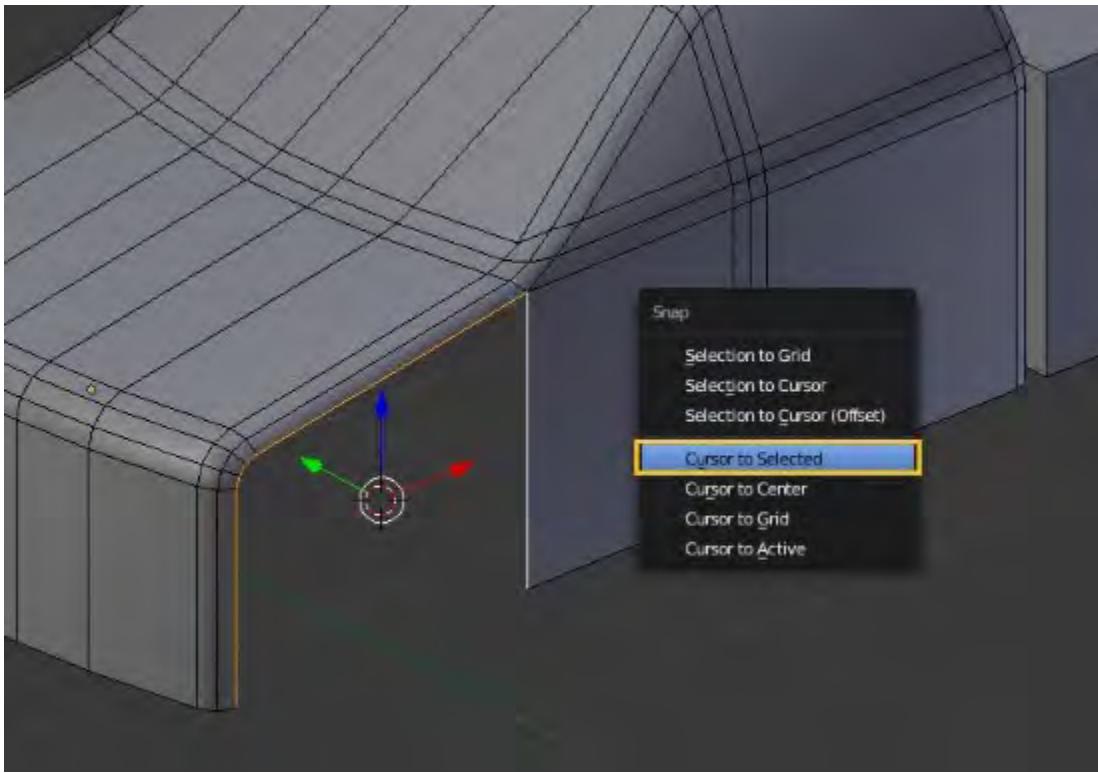


<pagebreak></pagebreak>

Then, you can select all the faces with a number of vertices/sides greater than four. This will highlight any N-Gons that may have slipped in:



The front quarter panel would need to be redone. Of course, we need to cut out the wheel area anyway, so we'll just go ahead and delete that face. Then, we'll set our cursor to the middle of those edges:

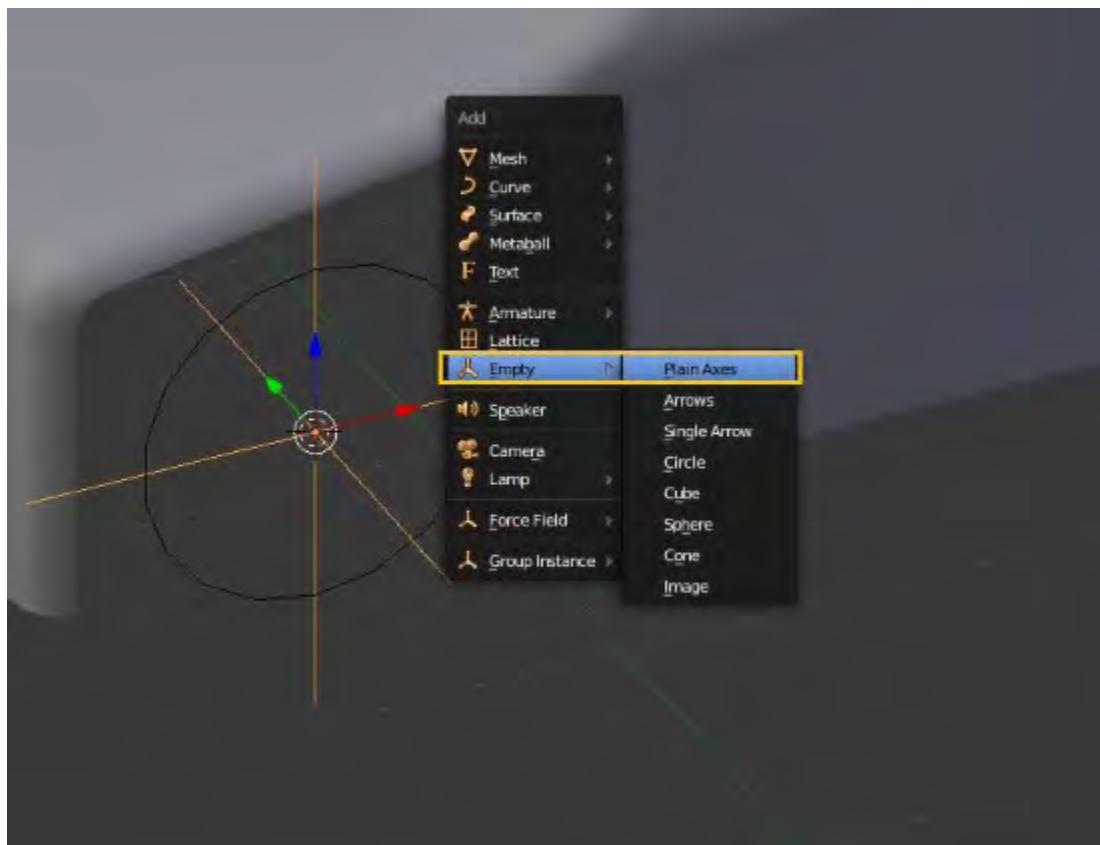


<pagebreak></pagebreak>

Next, we'll add two things.

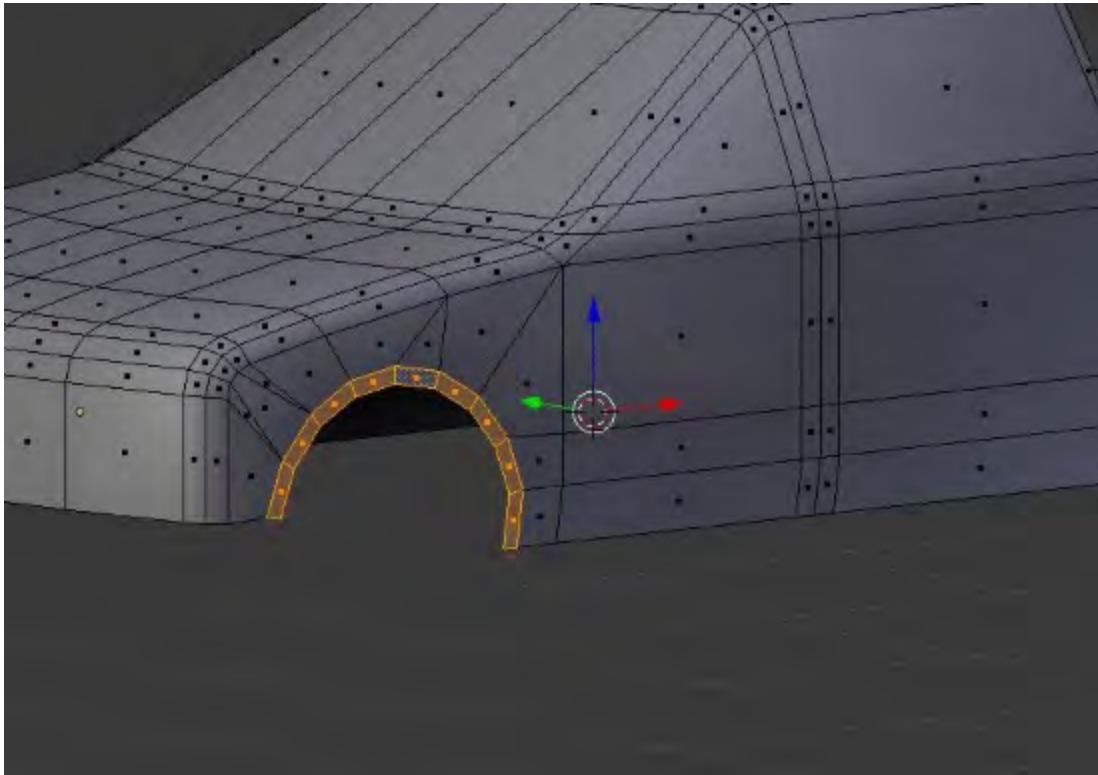
Within the mesh, we'll add a circle to form the shape of our wheel well. Be careful how many sides you use here. Round objects (circles, cylinders, spheres, and so on) are notorious for consuming polygons. We'll go with 24 sides to this circle and add it in.

I'm also going to add an here. This will allow us to keep track of the original center point of that circle (in case we want to add a tire at that exact point later). I'm also going to add an **Empty** here. This will allow us to keep track of the original center point of that circle (in case we want to add a tire at that exact point later).



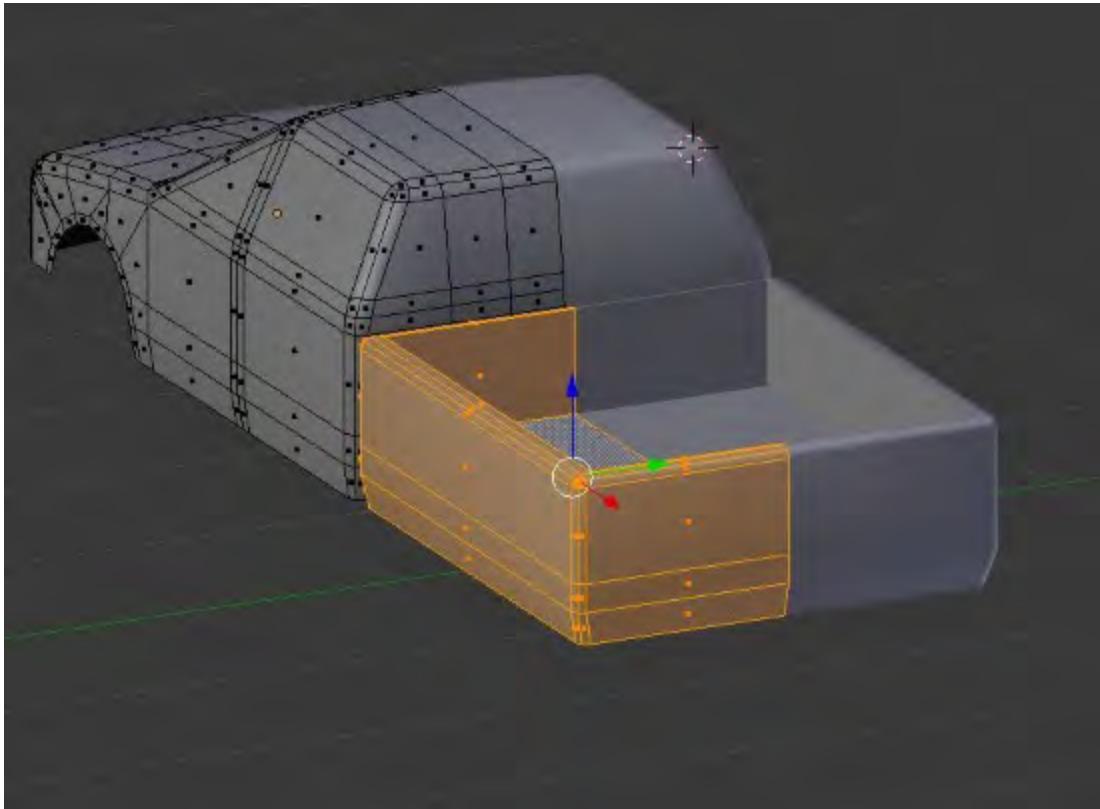
<pagebreak></pagebreak>

Next, we'll eliminate the parts of the circle that we don't need and build it into the mesh. I'm going to create a ring of faces around the outside of the circle, which will allow us to add a slight flare to the fenders.



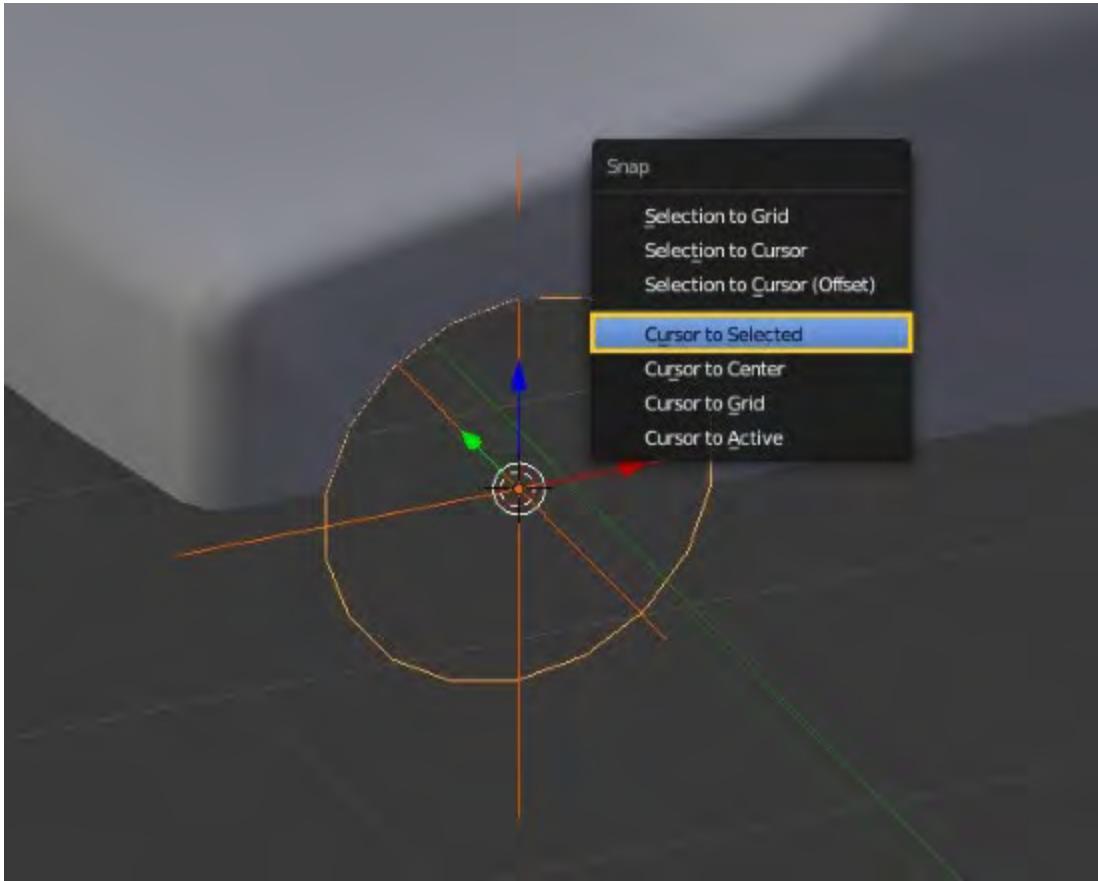
Before going any further, I'll add a mirror modifier to the model and delete half of it.

Next, we'll add a little detail to the bed of the truck:

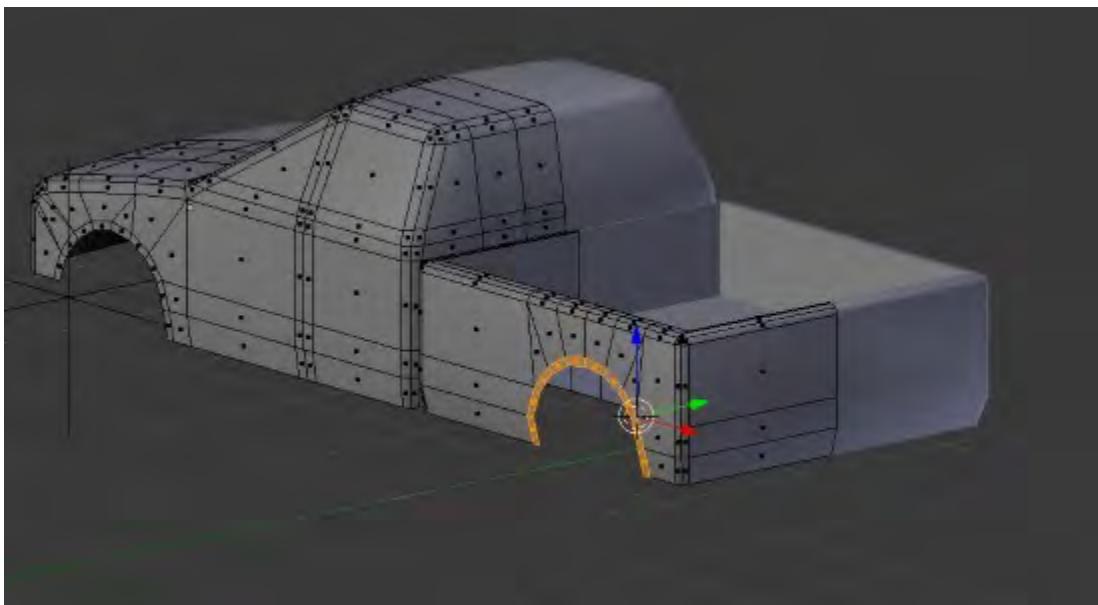


<pagebreak></pagebreak>

Now, I'll set the **3D Cursor** back to the **Empty** and add another circle. I can match the exact size and position of the original circle we cut in, and then move it back to create the rear wheel well.

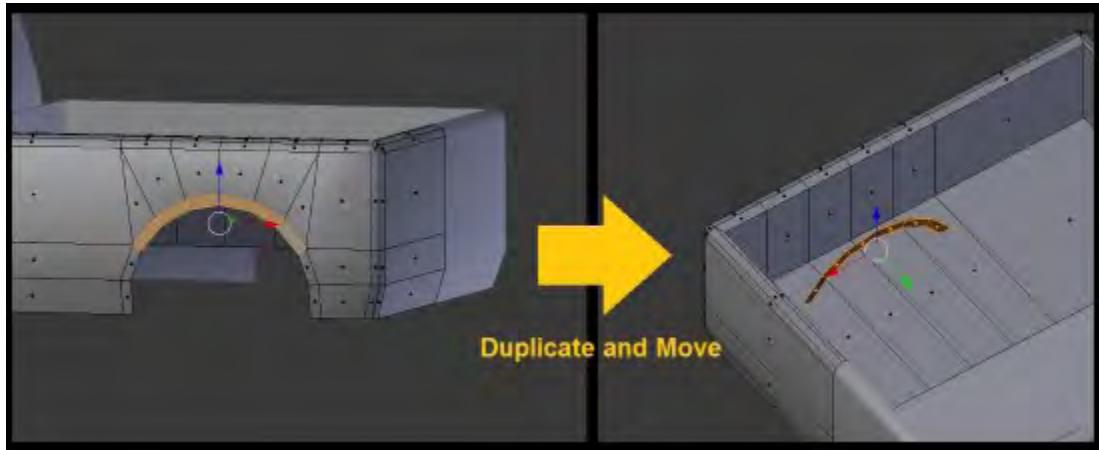


Using the same technique as before, we can create the cutout for the rear wheels:

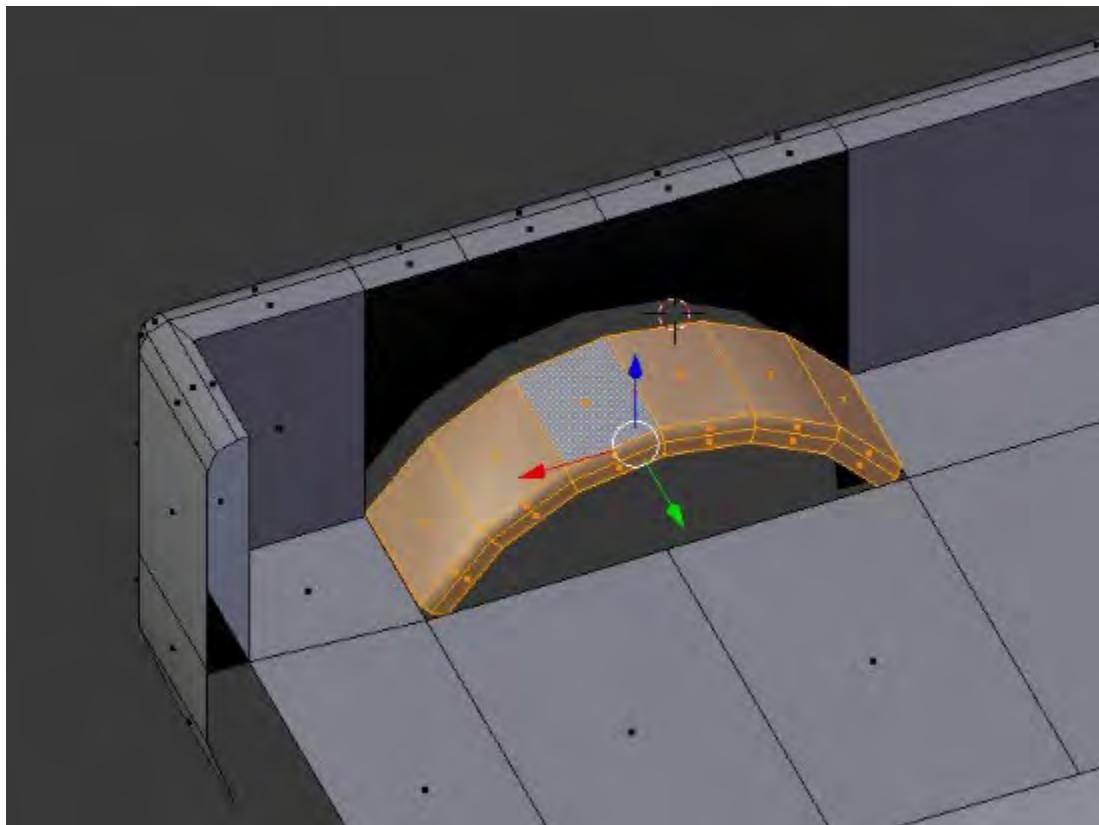


<pagebreak></pagebreak>

Afterwards, we'll duplicate a portion of that circle and move it in towards the center of the truck bed. This way, we can create the shape of the wheel well inside of the bed.



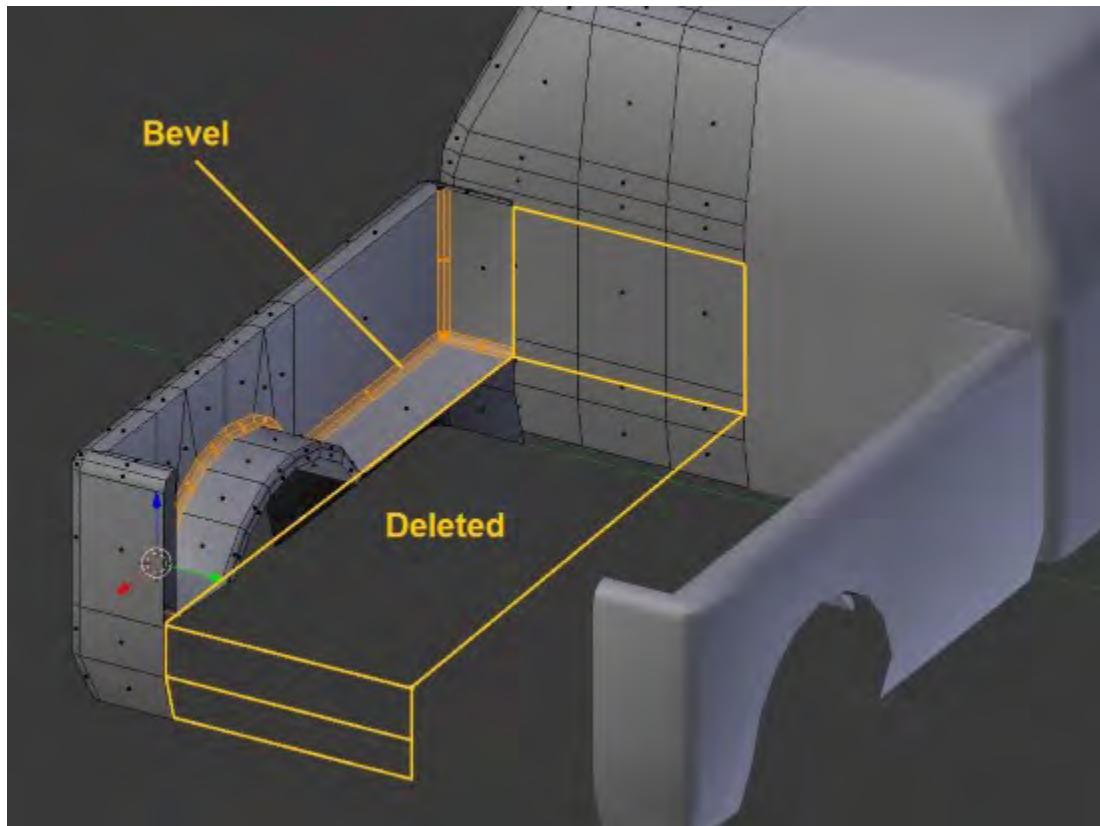
Then, we can delete the faces we don't need and build that wheel well into the mesh of the bed.



I'm just going to bevel the inside of the bed slightly.

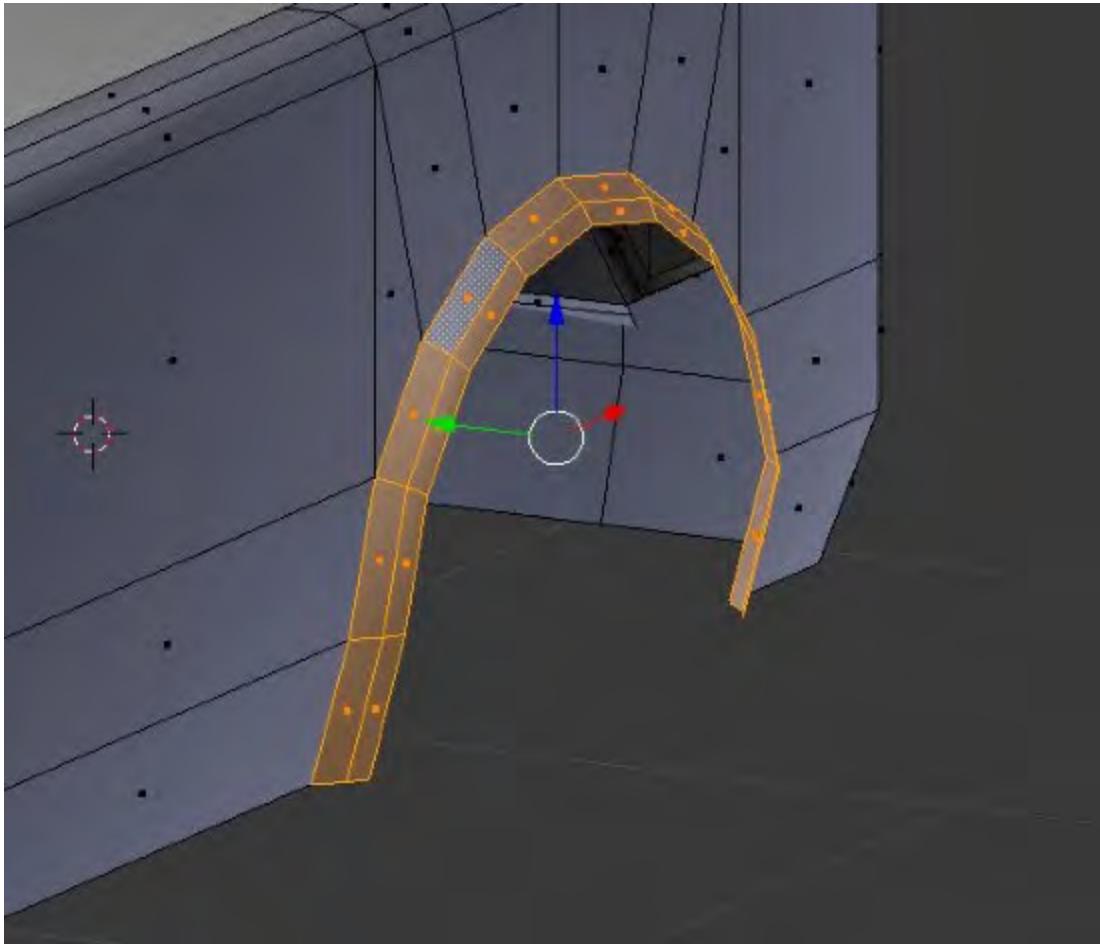
Note

Sometimes it's easier to delete the middle portion of an object first and then just extrude the edges back to the center once you're done beveling.



<pagebreak></pagebreak>

Now, we can run a loop cut around the ring of faces that makes up the wheel well. We can use Alt + S to give ourselves a slight flare to the fenders. Again, you probably only need one loop cut here—don't add too much geometry in an attempt to make it look smoother.



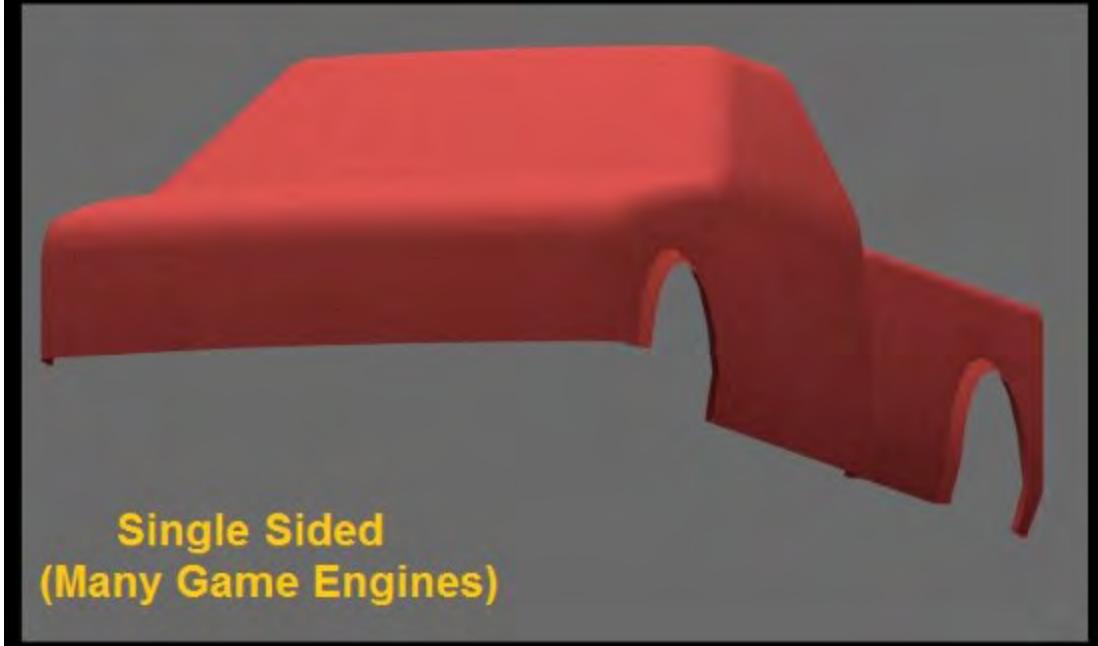
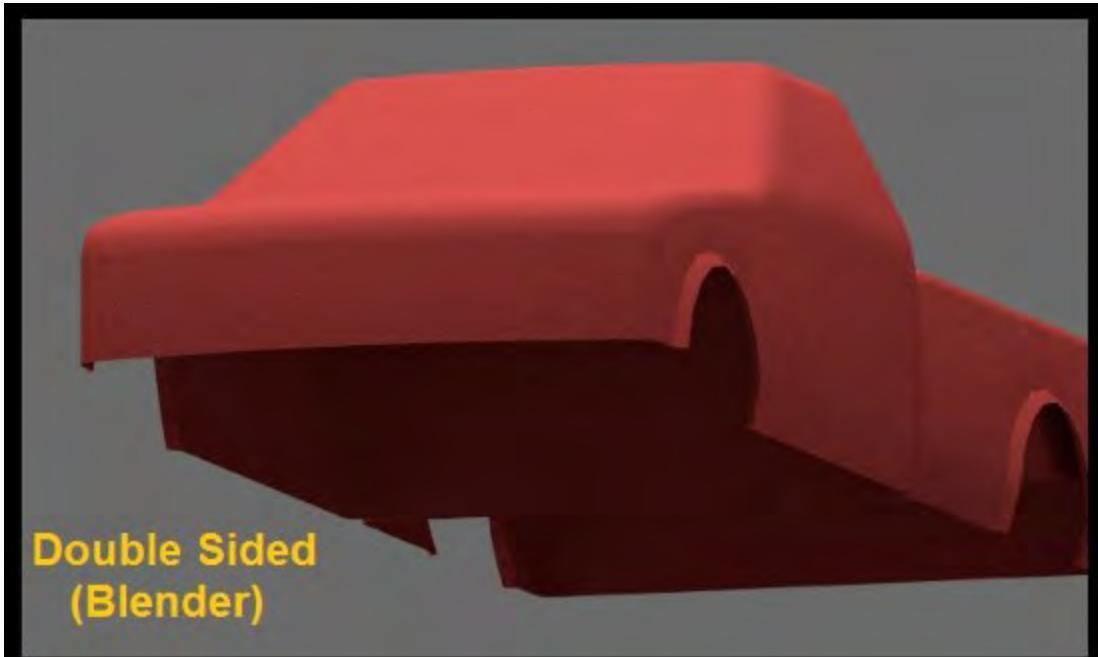
Manifold versus non-manifold meshes

Before going further, there's an important topic we need to touch on. Right now, we can see the back side of our body panels from certain angles. In other words, when you look into the wheel well, you can see the back side of the faces that make up the truck's tailgate.

This generally isn't a good thing, because it reveals the single-sided nature of the faces (and can sometimes have odd effects on your shading and materials). However, you can often get away with it in Blender. If you happened to see the back side of a body panel during an animation, it would be hidden in shadows and probably wouldn't look too bad. This is especially true with a low-poly model like this, where you're not expecting perfect realism.

<pagebreak></pagebreak>

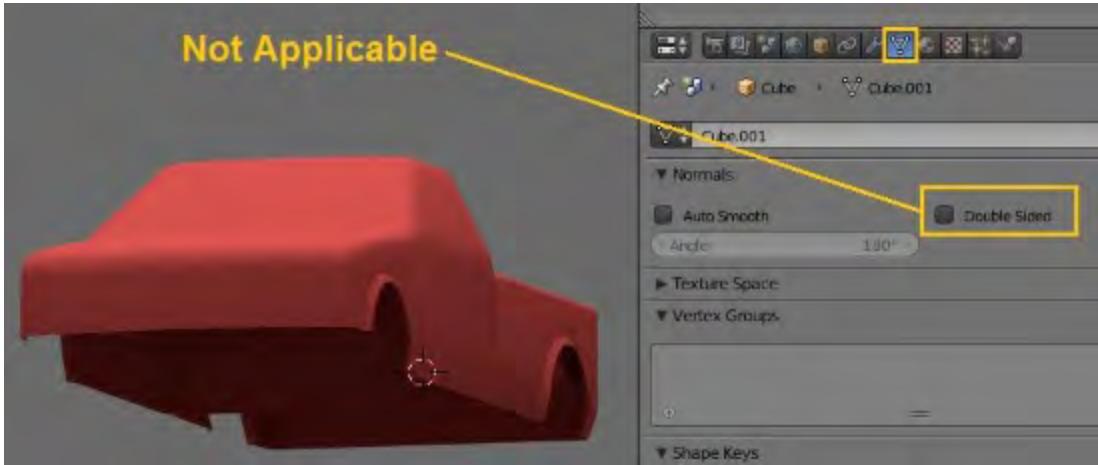
The problem is that some game engines aren't as forgiving as Blender. To increase rendering efficiency and reduce CPU/GPU workload, they simply don't render the back side of polygons. Instead, it will appear that those faces are missing from the model.



So a model that looks okay in Blender may not work in another program.

Note

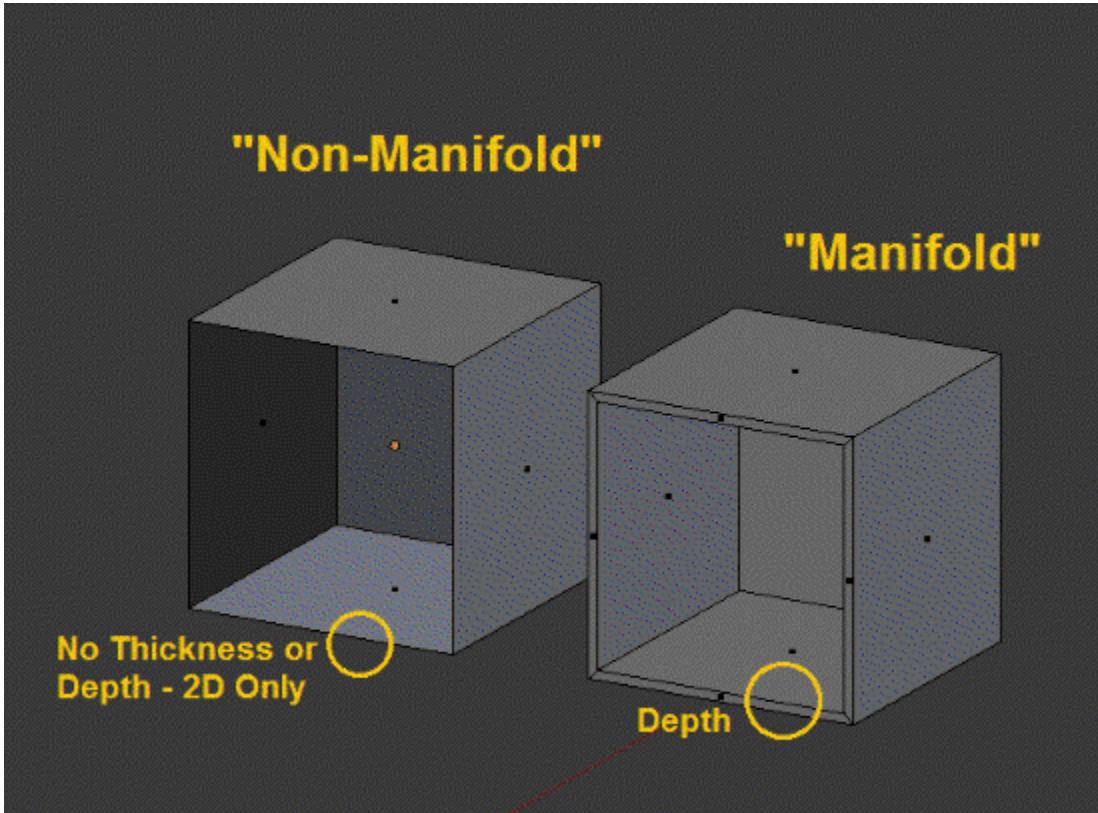
Just to clarify, this topic is only slightly related to the **Double sided** option under the **Mesh** tab. Regardless of whether this option is checked, Blender will still render both the front and back of a polygon. The box just allows you the option to display materials and textures regardless of the direction of your normals.



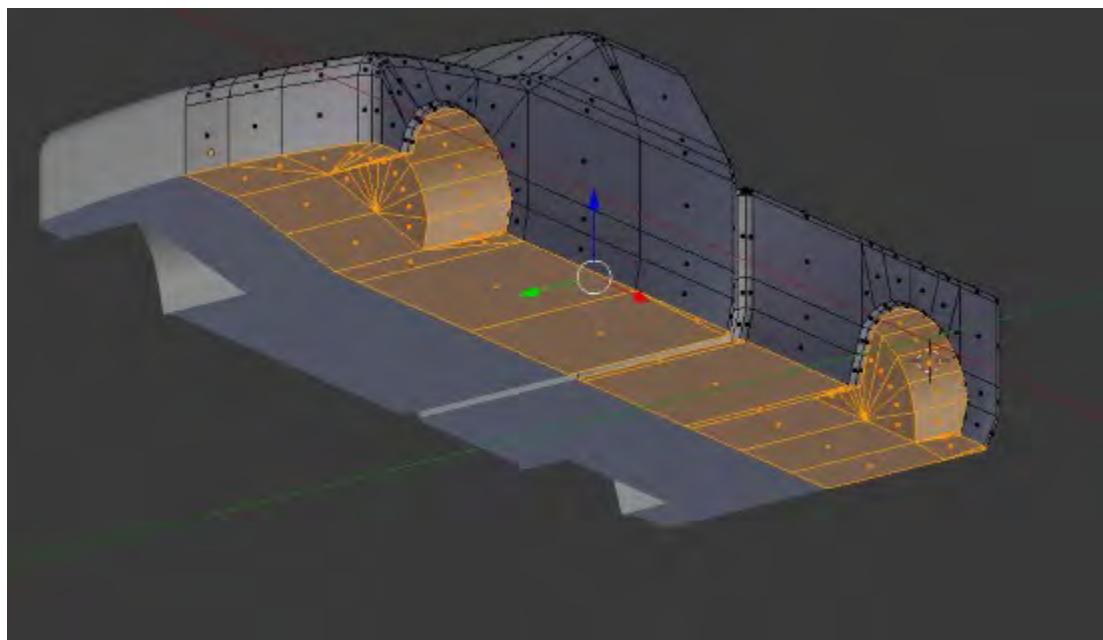
<pagebreak></pagebreak>

To make sure we don't have any problems with the back side of polygons, the best solution is to fill in the geometry so that we have no open or "non-manifold" meshes.

The concept of a non-manifold mesh can be tricky to understand at first. A good way to grasp the idea is to ask whether a given mesh could exist in the real world.



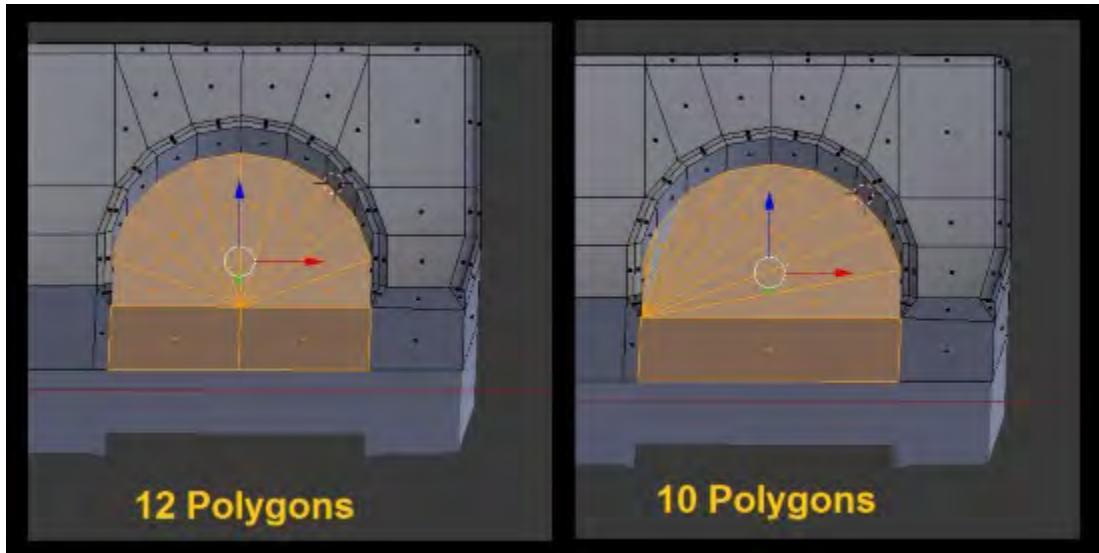
So, let's fill in the bottom of our truck to avoid this issue.



<pagebreak></pagebreak>

Adding details

As we build our model, it's generally good practice to use as few polygons and vertices as possible. However, you may sometimes decide that it isn't worth it. For instance, you can technically save two polygons by filling in the fender area in a nonstandard way:



This tends to look a bit messy, though, and can sometimes make it difficult to see how meshes are put together. You'll have to decide how important polygon efficiency is for your application. In this case, I don't feel that saving two polygons is worth it. The final truck is around 5,000 polygons (as mentioned before), so we're talking about less than a 1% decrease in efficiency here (0.04%, actually).

At this point, we've completed the basic shape of our truck body. You can see where we're at in terms of polygons and vertices.

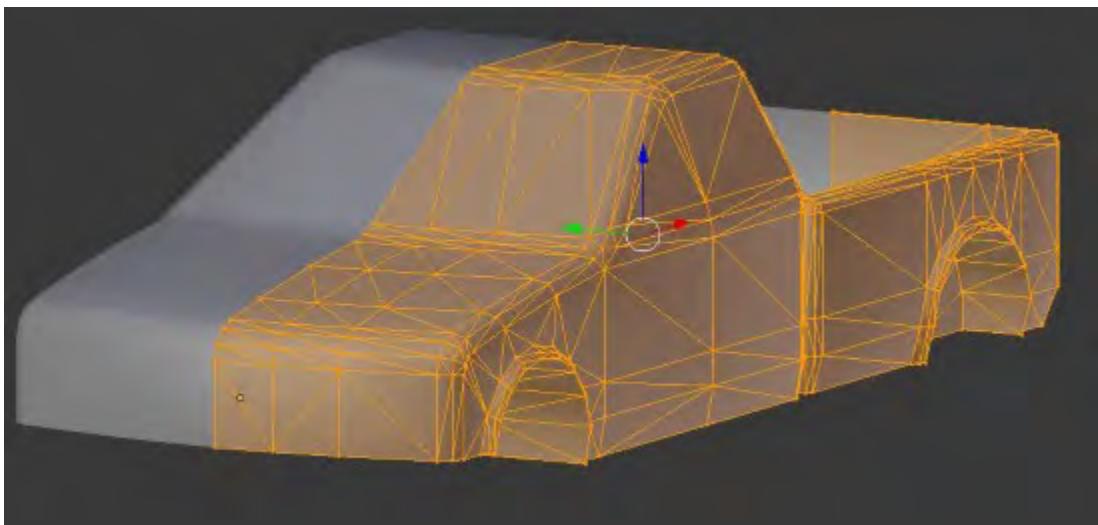


<pagebreak></pagebreak>

One thing we haven't really discussed yet is triangles. We've talked about using triangular faces ("Tris"), but that leads us to a larger point. The reality (at least in most programs) is that every polygon is actually made up of triangles.

Quads and N-Gons don't really exist at render time—they're just a modeling tool to create more efficient collections of triangles.

For instance, we used many Quads to model our truck body. At render time, however, Blender looks at it like this:



Why is this important? In the case of our truck, it really isn't. But with other game models, the number of triangles can be significant.

Let's say that you're building a model for a game engine that does accept N-Gons. You might add a circle with 200 sides and fill it in with one face:

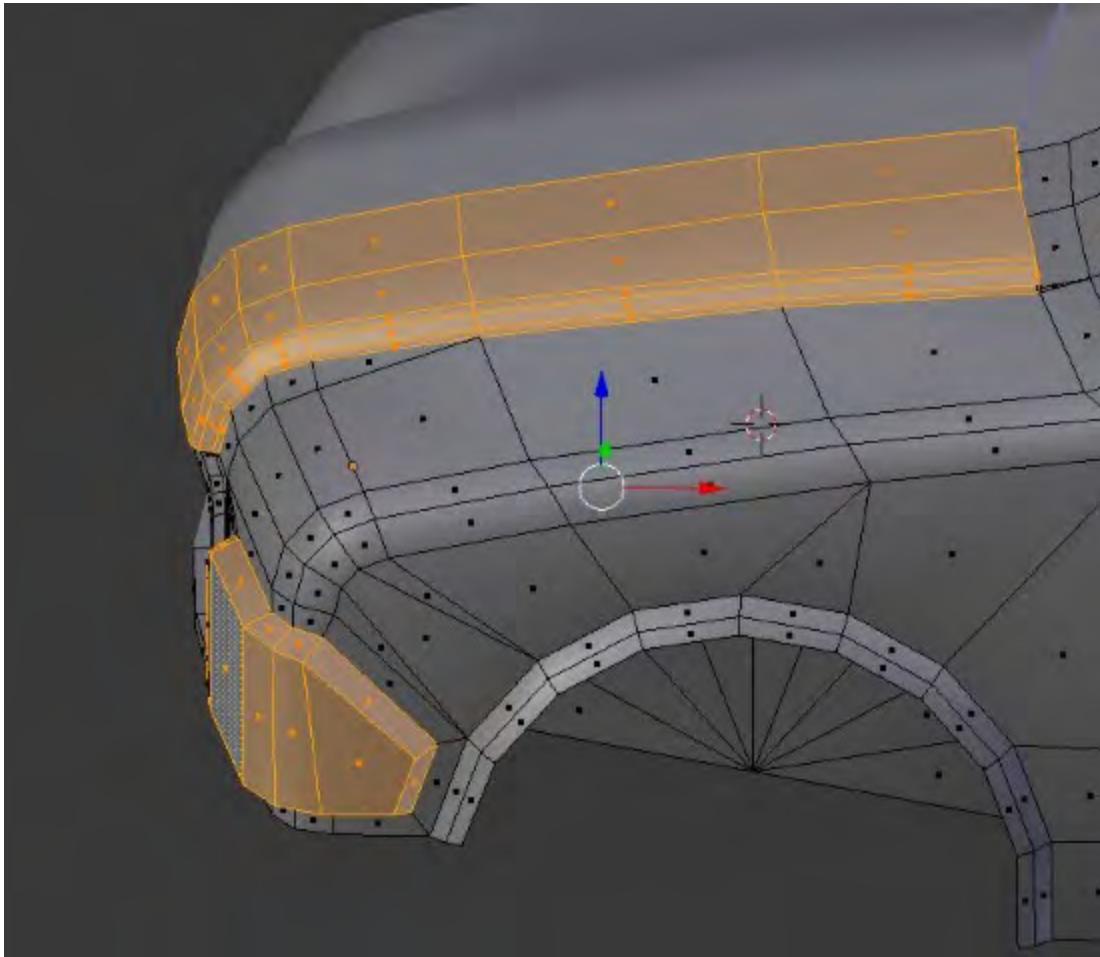


<pagebreak></pagebreak>

Sure, there's only one polygon being displayed, but you're using 198 triangles. What this means is that your mesh isn't nearly as efficient (or "low-poly") as you thought it was.

Again, this doesn't really affect us on this project, but it's something to keep in mind if you're going to be building a lot of game models for export.

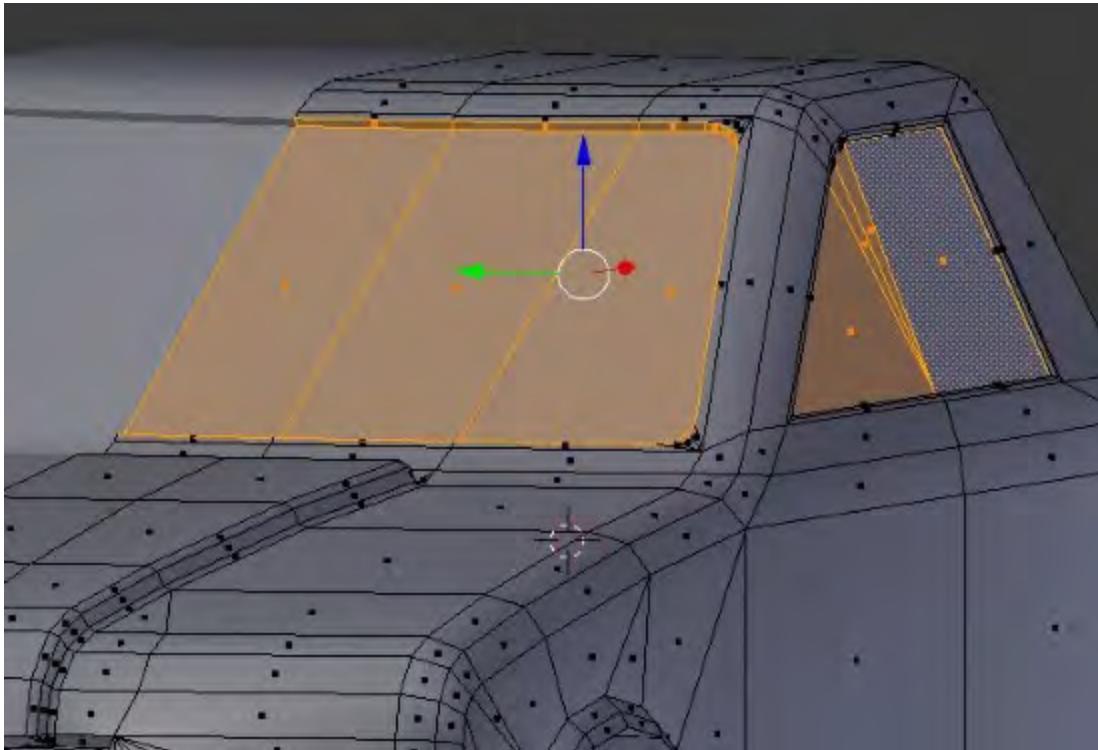
Now that we've got our basic truck body done, we'll go ahead and add some detail. I'll start by extruding portions of the hood to create a cowl and front headlight:



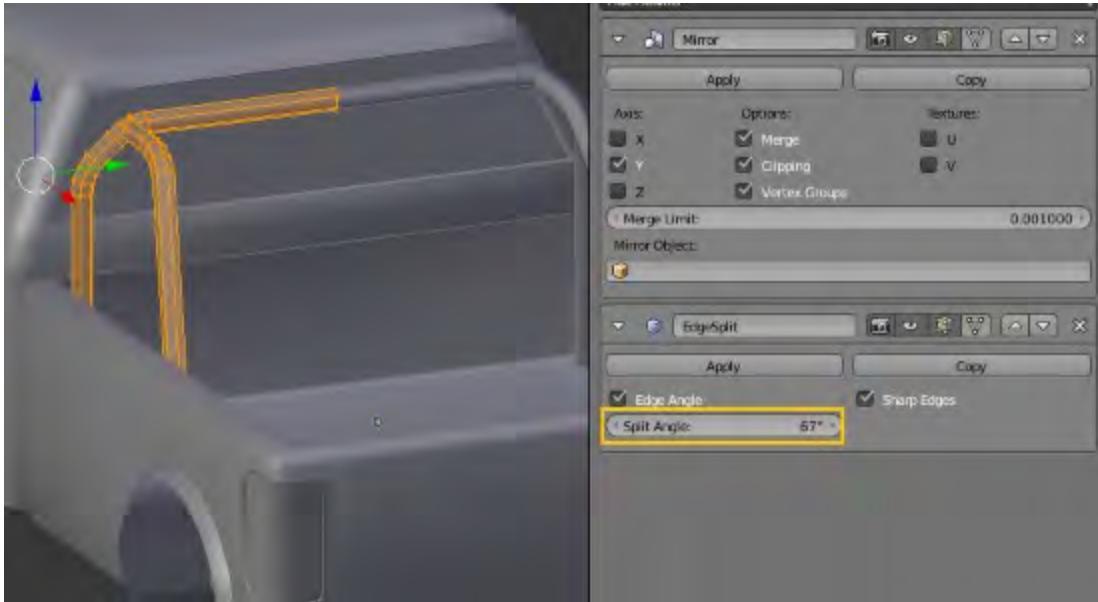
Next, I'm going to extrude the windows in just a bit. This is another area where we could actually save polygons. If we didn't extrude the windows in, we could just texture them later on. However, it doesn't cost us very many polygons to do it this way, and we can later add a "glass" material to just those faces (rather than using a texture map to differentiate between different types of material). So while this isn't the most efficient way to do it in terms of polygon count, it is more efficient in terms of the time you need to invest (and the required number of texture maps).

<pagebreak></pagebreak>

As you will quickly discover if you build game models, there are pros and cons to just about everything. Sometimes, we just need to make a choice.

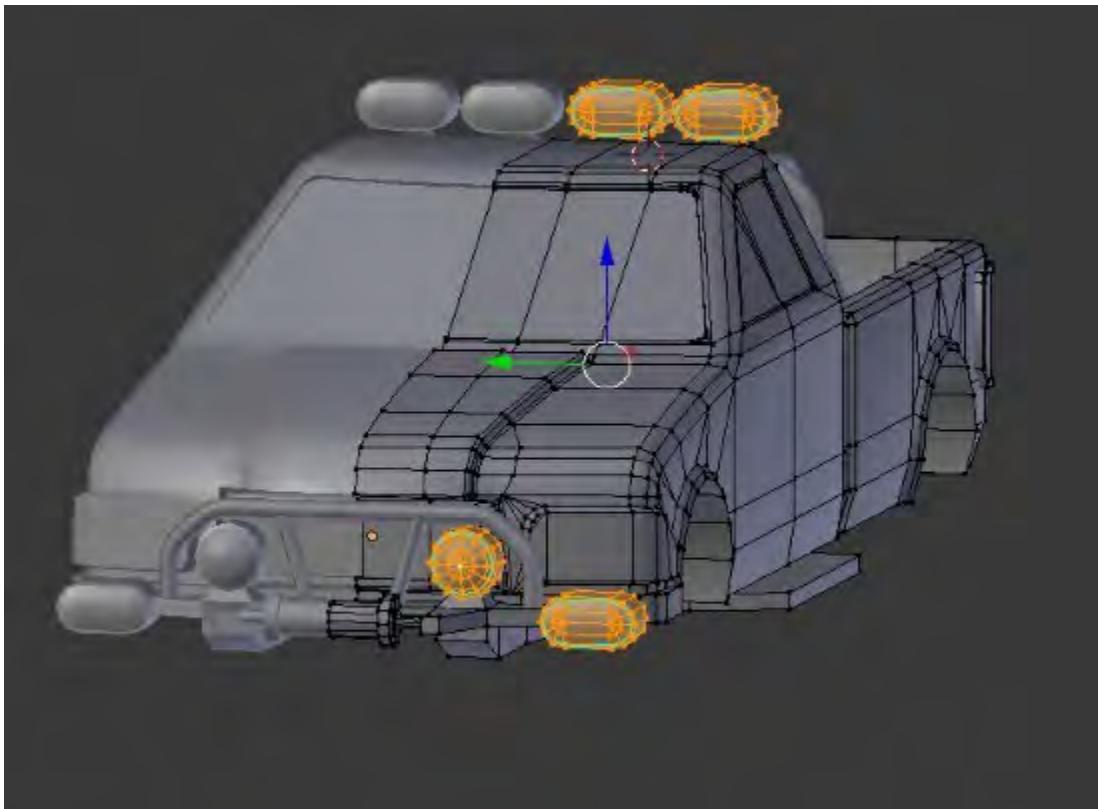


Next, I'll add some tubular roll bars to the bed of the truck. You can do this however you'd like, but I just started with a Torus object and extruded a portion of it (as we've done several times before). Since the end of your torus will be sticking inside of the truck and won't be visible, you can leave that particular part as non-manifold geometry. Alternatively, you can create an N-Gon at the end of it to "seal off" the mesh. In this case, I am only using six sides on each section of tube-they're fairly narrow, and I think that's consistent with the rest of the model. Again, anything circular consumes polygons very quickly, so try to get away with as few sides as you can.



<pagebreak></pagebreak>

I'm also going to add in a few extra lights for our truck:



Next, I'll create some wheels (as a separate object):

Remember, since the truck has four wheels, every polygon you add is being multiplied by a factor of four. You can add as much detail to the wheels as you'd like, but just keep an eye on how much geometry you're adding.

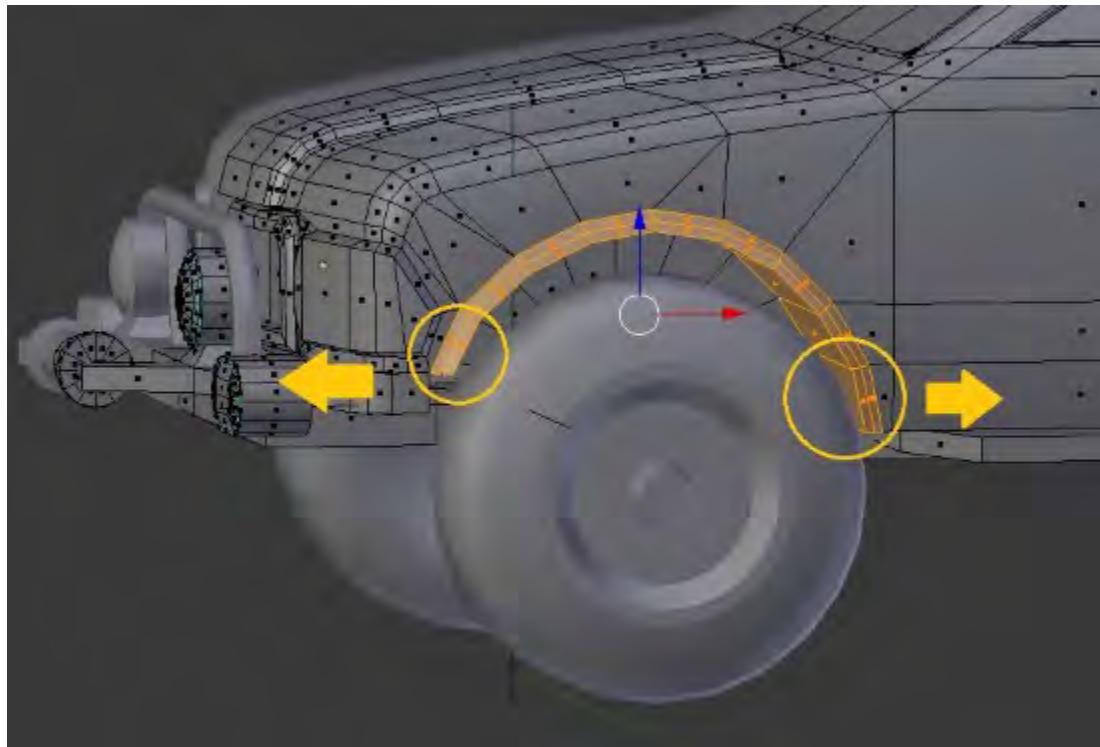
<pagebreak></pagebreak>

Another thing to keep in mind is the level of realism you're going for. If this truck is going to be used in an off-road racing game, you may want to create shock absorbers and various suspension pieces. That would certainly increase the realism.

On the other hand, it will add polygons to your model and require additional work when you set up your game. You'll have to make sure that the various parts all fit and move correctly.

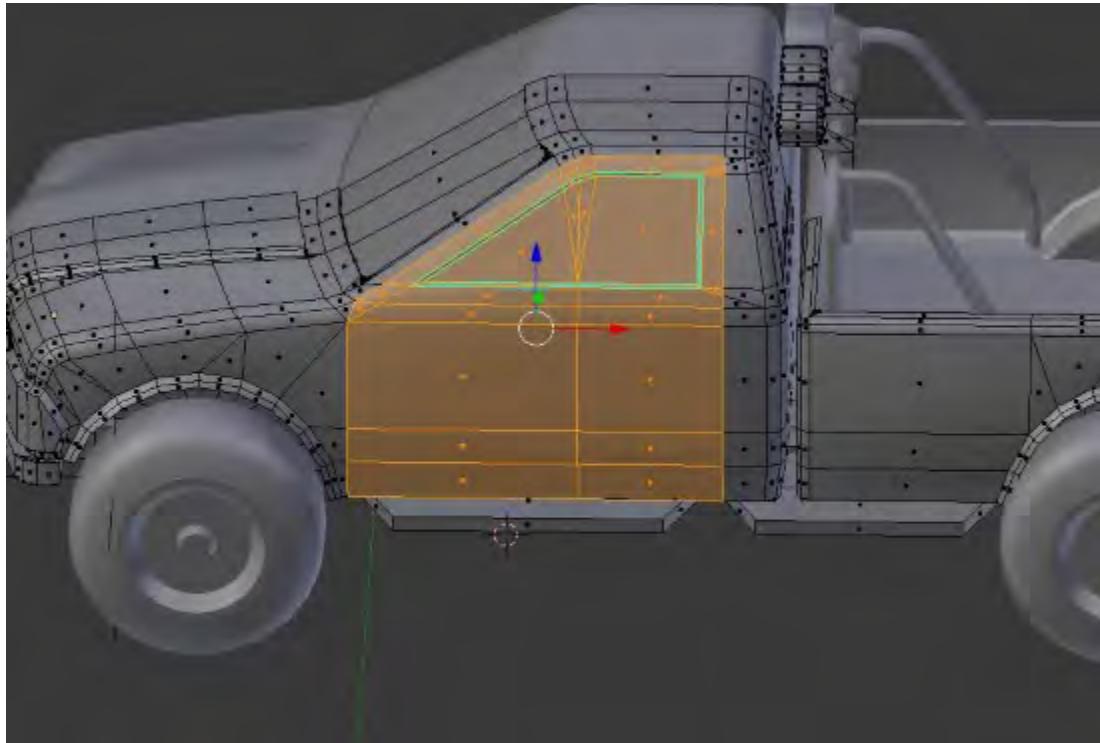
By contrast, you could skip most of the suspension parts and just add an axle and some blocked-out shapes. This will make it much simpler to import the truck into a game, but of course it won't be quite as realistic.

If you choose to use oversized (off-road) tires, you may need to go back and adjust the fenders so that everything will fit:



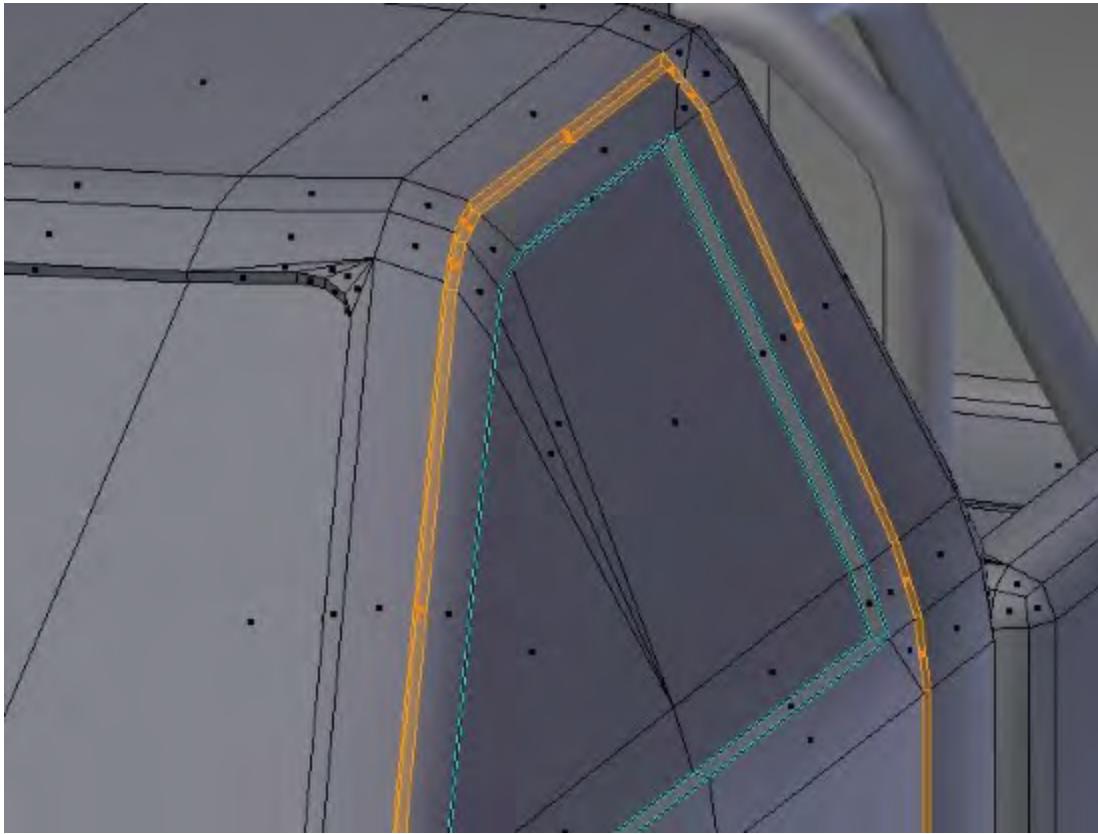
<pagebreak></pagebreak>

The next thing I'll do is select the faces that make up the door area.



Using the **Inset** tool, we can create a slight panel line where the door connects to the body. This is another case where we don't technically need to do that (it uses polygons), but there may be a benefit.

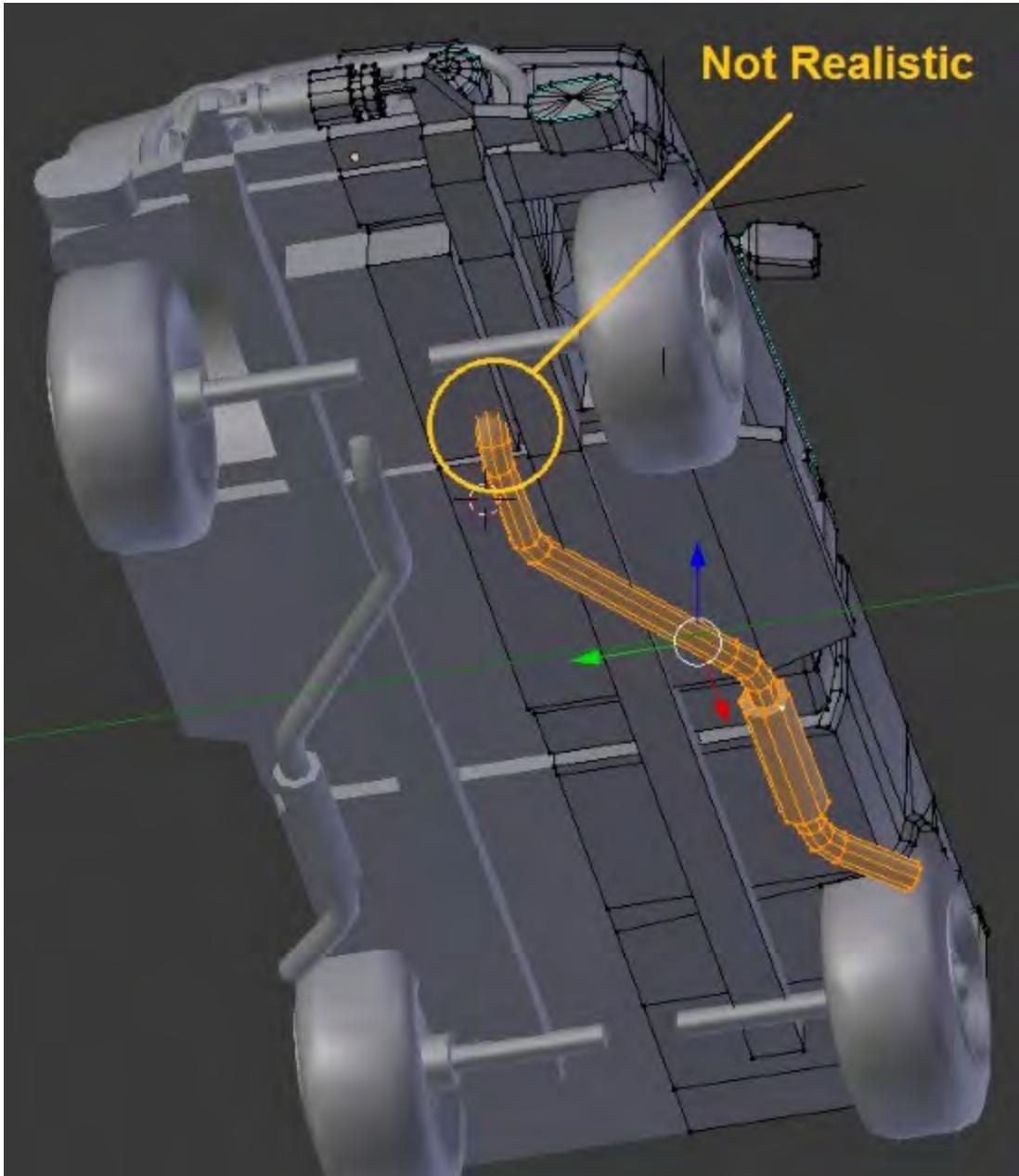
For example, maybe you'll want to show a character getting into the truck, or maybe you want a version of the truck that has no doors. Perhaps the truck even gets damaged during the game and the doors come off. In any of these cases, it's advantageous to have the door physically cut into the truck.



<pagebreak></pagebreak>

Next, we'll add a bit of detail to the bottom of the truck. This is generally a good place to save polygons, since the bottom of a vehicle is rarely visible.

For instance, I've added an exhaust pipe that runs along the underside of the truck, but it just disappears into the engine area.



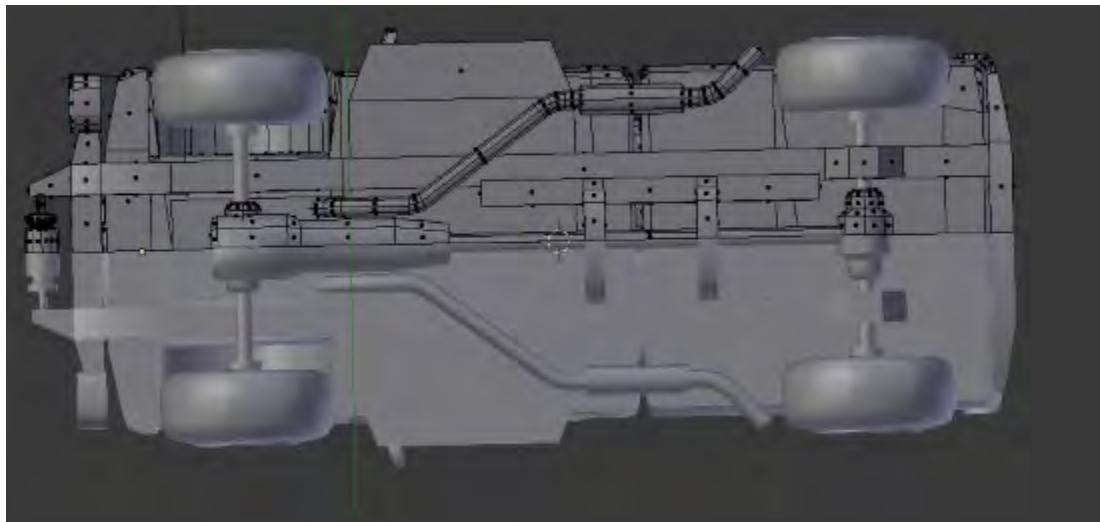
This can be frustrating when you build game models. Cars are a hobby of mine in real life, so I know that the exhaust pipes should run into a header (or exhaust manifold), which in turn should connect to the engine block. My natural instinct is to do that with the model. However, that would probably be a mistake. We'd be adding a lot of geometry to an area of the truck that's rarely seen.

More importantly, we want to keep our detailing consistent. If I created a realistic exhaust manifold, I'd also have to create a realistic suspension, fuel tank, transmission, and so on. Pretty soon we'd be adding thousands of polygons and defeating the purpose of the project.

<pagebreak></pagebreak>

Note

When it comes to game models, it's not a question of how much detail you can add, it's a question of how much you can afford:



Obviously, there's a lot of flexibility in the design of this truck. A quick Internet search will provide plenty of inspiration, and (just like all of these projects) you should feel free to take things in your own direction.

This is what my truck ended up looking like:



The final polygon count was 5,348 faces or 9,946 triangles. That's well within the range of most game engines. In fact, it's really on the low side. If you had a specific application in mind, you might want to test your model out to make sure it will work. If it does, you can always go back and add detail later.

<pagebreak></pagebreak>

Summary

In this chapter, we created a basic vehicle model that can be exported for game use. In the next section, we'll create a UV and texture map for it so that it can be exported to a number of different applications.

Chapter 10. Low-Poly Racer – Materials and Textures

In this chapter, we'll add materials and textures to our model. Then, we'll use UV unwrapping to bake those textures out to an image. That way, they can be used in external applications. Finally, we'll create Cycles-based materials using our image. The following topics will be covered in this chapter:

- Texturing workflow
- Adding materials
- Unwrapping and baking
- Adding detail in an Image Editor
- Checking the Texture Map

Texturing workflow

First, we'll create some procedural textures in Blender. These will be the base textures (not the final product).

After we do that, we'll create and adjust the UV map for our truck.

Then, we'll bake the textures to an image using our UV map. We haven't covered baking yet, but it's pretty straightforward. If you're already familiar with the concept, then great! If not, don't worry, we'll discuss it as we go.

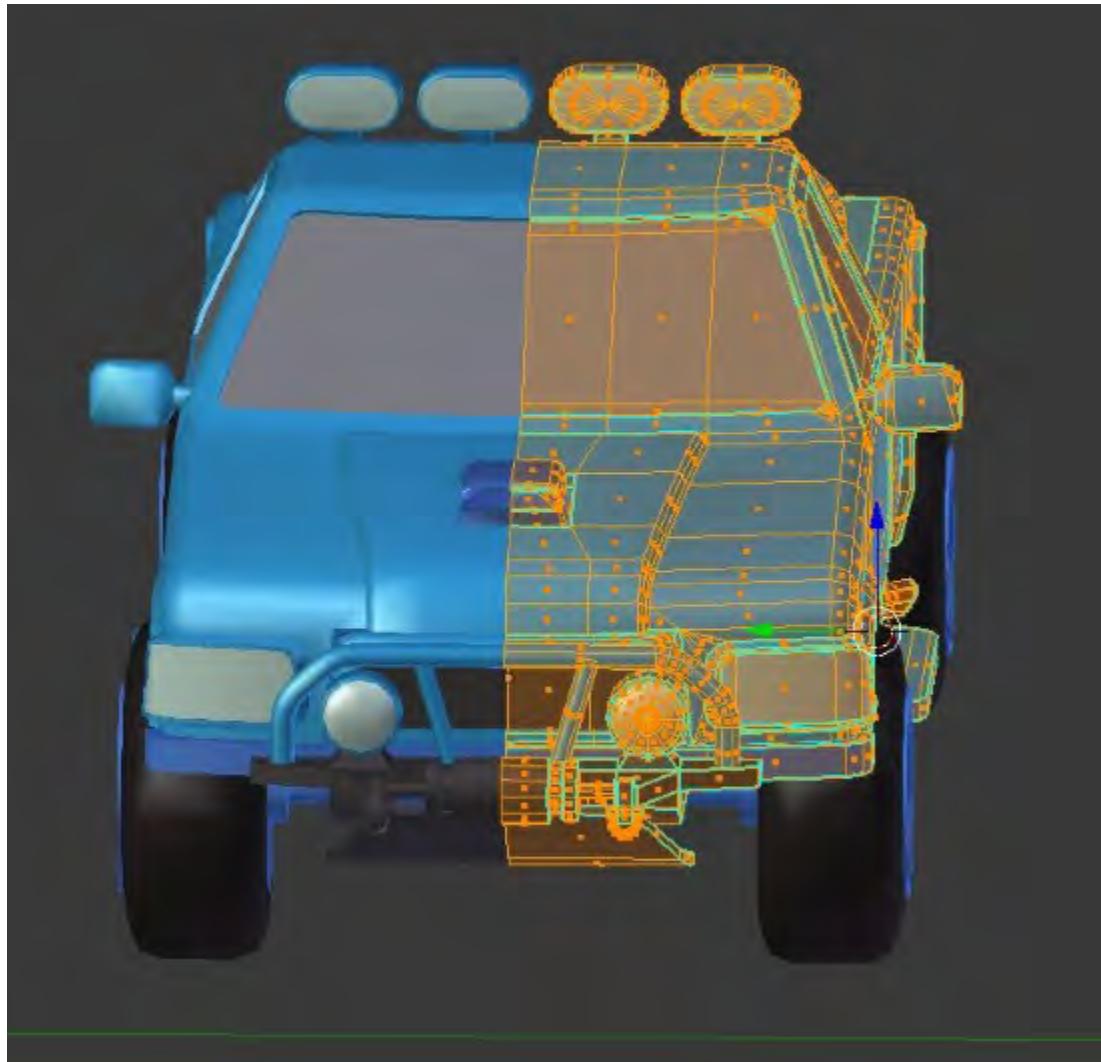
After we have our image-based texture map, we can tweak it in an external image editor and add details.

Finally, we'll use that image map to create Cycles-based materials for the truck. This last step isn't critical, since the truck model is intended for other applications (not Blender). But it's a good way to verify that everything lines up and works as expected.

So, let's get started.

Adding materials

The first thing we need to do is to go through and just add some basic materials. This will be similar to what we did with our spacecraft model—don't worry about actually creating the materials right now, just create the material slots and assign them to the appropriate parts of the mesh:



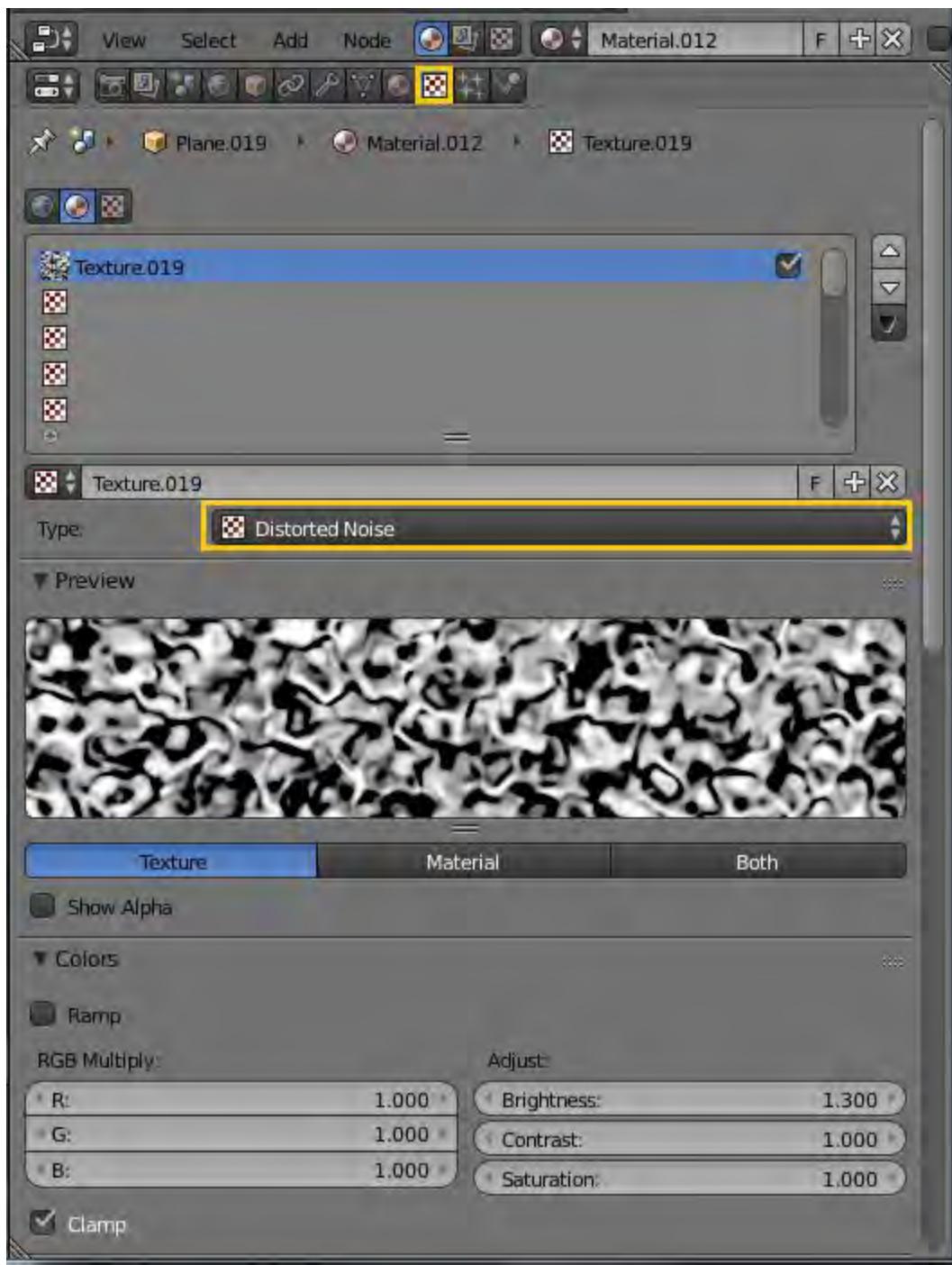
The primary material, of course, will be the truck's paint. We'll use Blender Internal materials to create a stylized paint scheme. First, switch to the internal rendering engine:



Since the introduction of Cycles in Blender 2.5, it has become less common to use Blender Internal for anything. However, there were some interesting features built into the Blender Internal engine. For instance, the materials and texturing system could produce some complex and unique patterns. We'll take advantage of that to create a custom paint job for our truck.

<pagebreak></pagebreak>

Before we continue with the truck, it might help to create a simple object and play around with some textures. In this case, I'll just use a basic **Plane** and add a new material. Then, I'll try out a few different textures to see how they look. This, for example, is the Distorted Noise texture:



You can see that there are plenty of settings to adjust, but we'll leave those alone for now. Below the pattern itself, you'll see options for **Mapping** and **Influence**.

We'll set our mapping coordinates to **Generated**. This means that Blender will apply the texture based on the object's dimensions in the scene.

Note

Textures that use mathematical patterns rather than image maps are generally referred to as procedural textures.

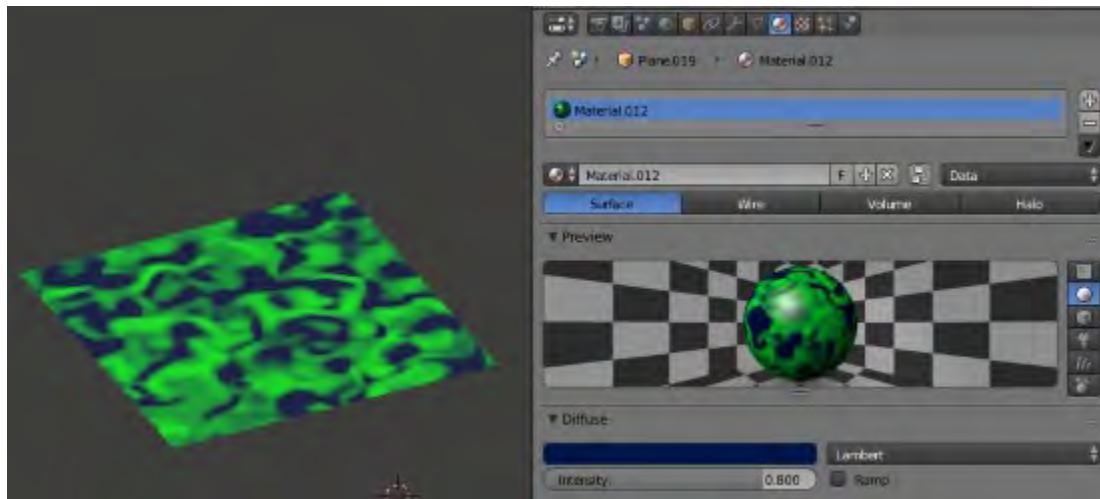
<pagebreak></pagebreak>

After setting our mapping coordinates, we'll select the **Color** option under **Influence**. This means that the texture will affect the color of our material. You can set the color to whatever you'd like. In this case, I've selected green.

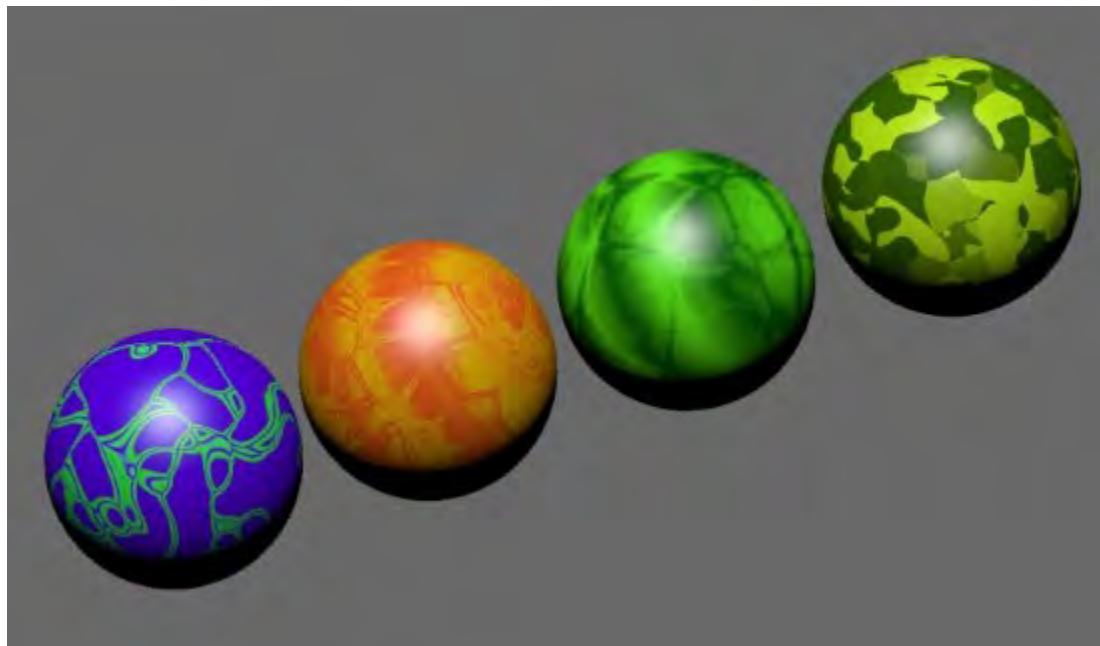


<pagebreak></pagebreak>

With a blue base material, here's how that texture looks:

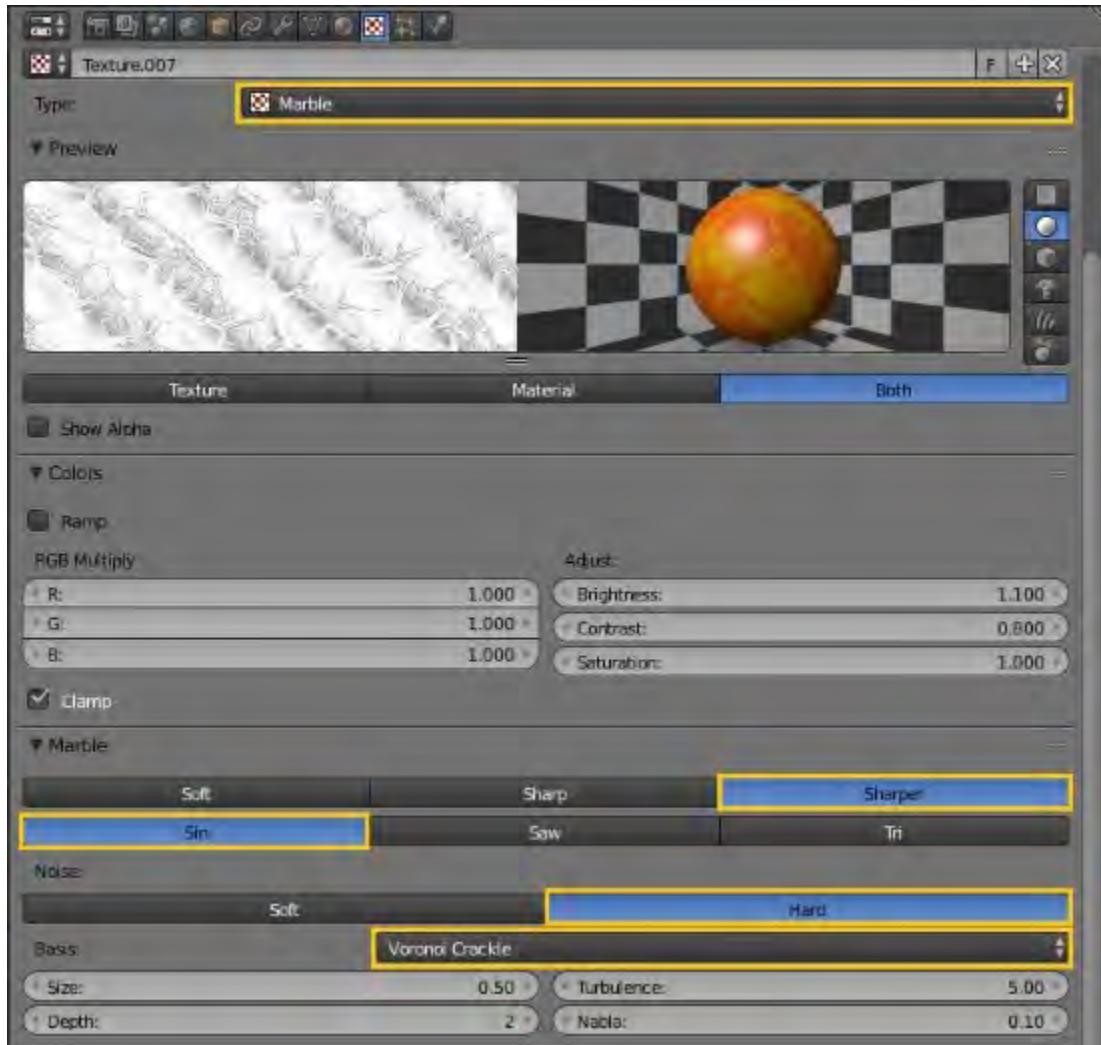


You can try out the various procedural textures and the different options for each one. With a little experimentation, you can get some pretty interesting results:



<pagebreak></pagebreak>

For instance, we can use a **Marble** texture and select **Voronoi Crackle** as our **Noise Basis**:



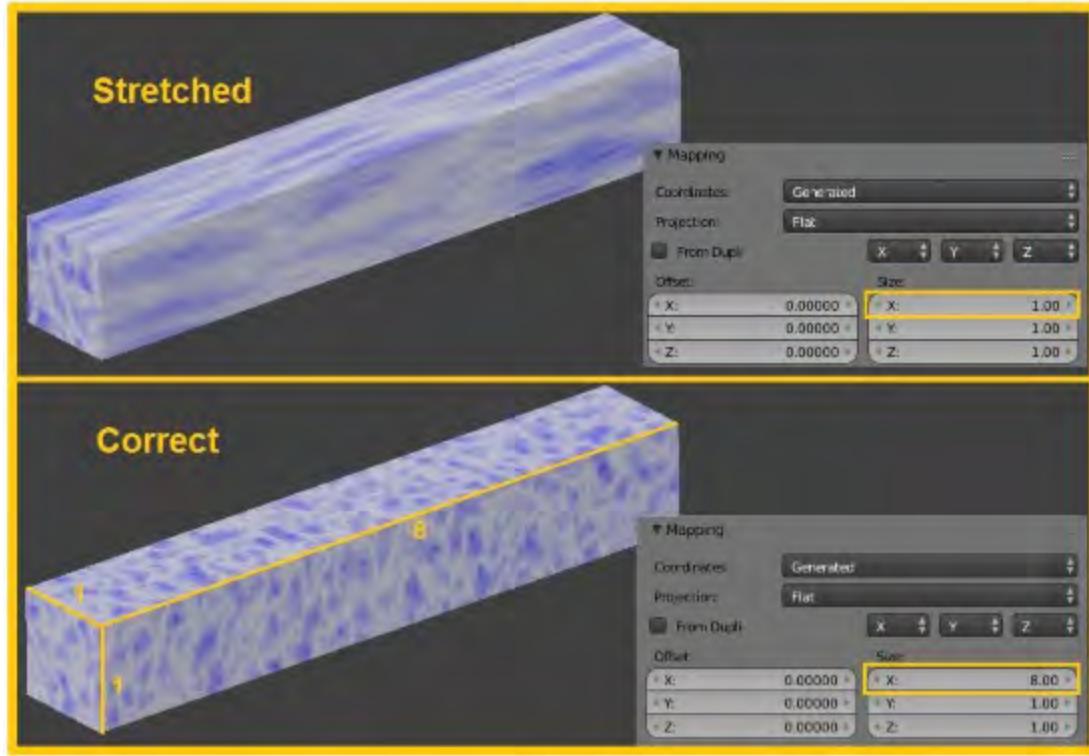
<pagebreak></pagebreak>

After playing with some settings, I think this will look pretty good for the custom paint job on our truck. Of course, you can try various textures (or combinations of textures), until you find one that looks good to you:



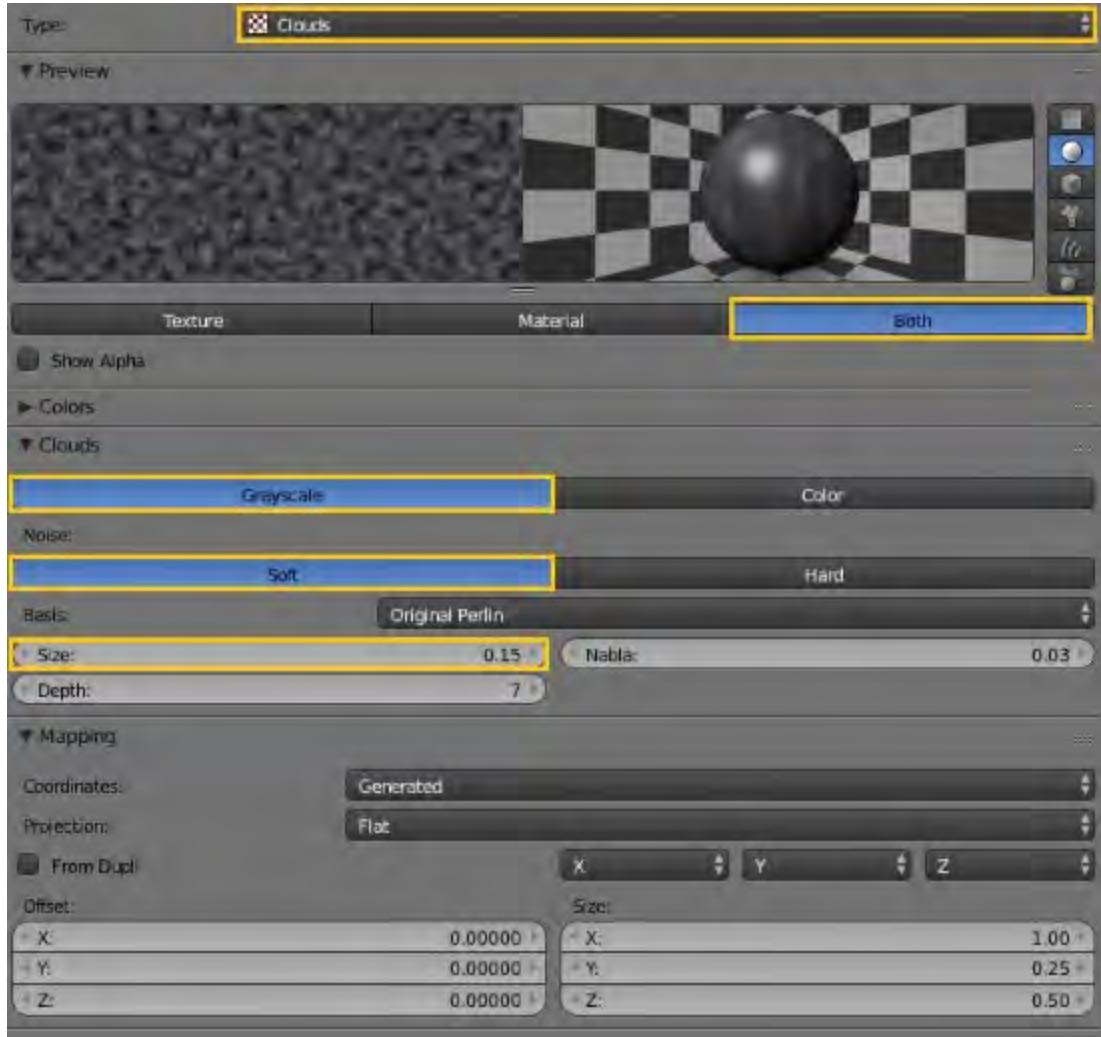
<pagebreak></pagebreak>

Depending on the dimensions of your object, you may need to adjust the size of your texture on the various dimensions. By default, Blender will create the texture with a size of 1 on each axis. If your object is longer than it is wide, or wider than it is tall, for instance, you will need to adjust the texture size to match:



<pagebreak></pagebreak>

Next, we'll use a similar technique to create a texture for the darker mechanical parts. Remember that these are base textures. In other words, you don't need to make them look perfect, you'll be able to adjust them later in an image editor of your choice:



<pagebreak></pagebreak>

A simple cloud pattern is probably good enough for now. As you can see on the bottom of the truck, it doesn't look particularly realistic. However, it does look better than a flat color, and it's seamless—no gaps or edges:

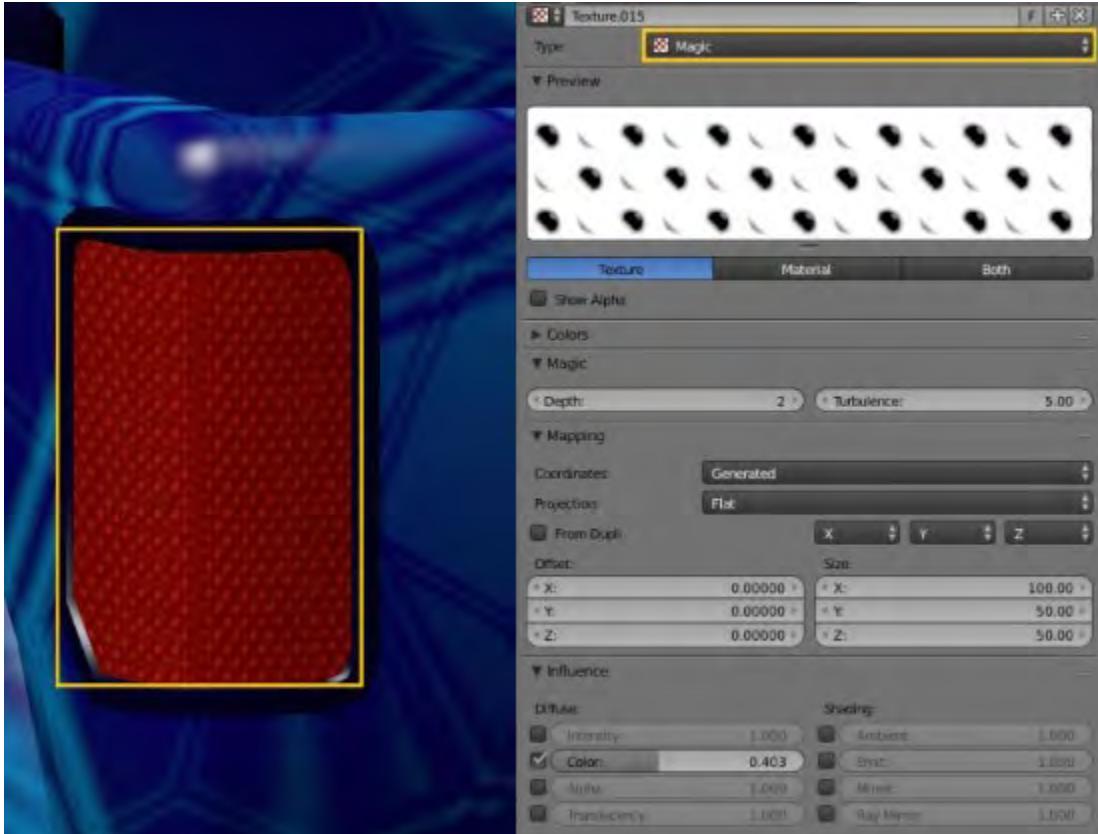


You can add multiple texture layers as well. Try a combination of large and small texture patterns, and just see what looks good to you:



<pagebreak></pagebreak>

We'll repeat this for the various materials. For the taillight, we can use a **Magic** pattern to add a bit of detail:



<pagebreak></pagebreak>

One thing I'll do at this point is delete some duplicate objects. For instance, each half of the truck has three sets of oval-shaped (oval-shaped) lights. Since they're all going to look the same, we can just continue with texturing one light and then make duplicates later. As you'll see in just a minute, that will make things more efficient:

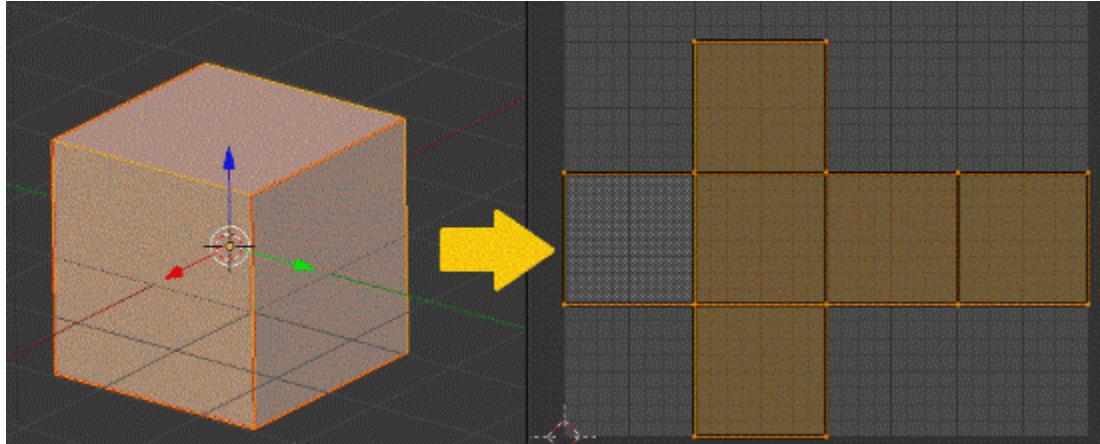


<pagebreak></pagebreak>

Unwrapping and baking

Once you have your basic materials created, we'll need to UV unwrap (or just unwrap) the truck. We briefly touched on this earlier in the book, but it's more important now.

If you're not familiar with the idea of UV unwrapping, it's really pretty simple. Our model exists in 3D space, but it will be using a 2D image texture (eventually). We need to tell Blender how to apply that 2D image to our 3D object. When we talk about unwrapping, you can think of it as a paper model. We need to cut and unfold the model so that it lies flat:

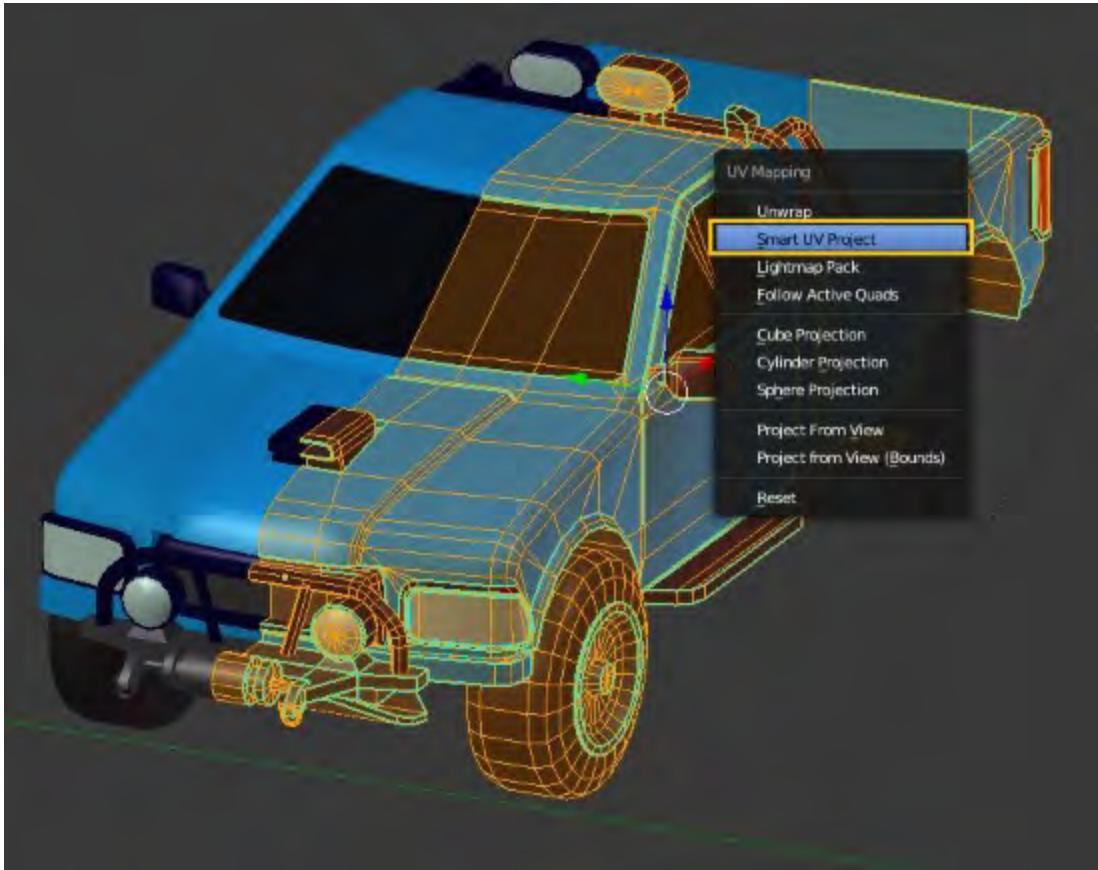


Note

UV unwrapping is a non-destructive process. You don't change or damage your 3D model at all. The traditional coordinates (X, Y, and Z) still define the location of your vertices in 3D space. A second set of coordinates (U and V) will define how those vertices are mapped to a 2D image. This is why it's called a **UV Map**.

<pagebreak></pagebreak>

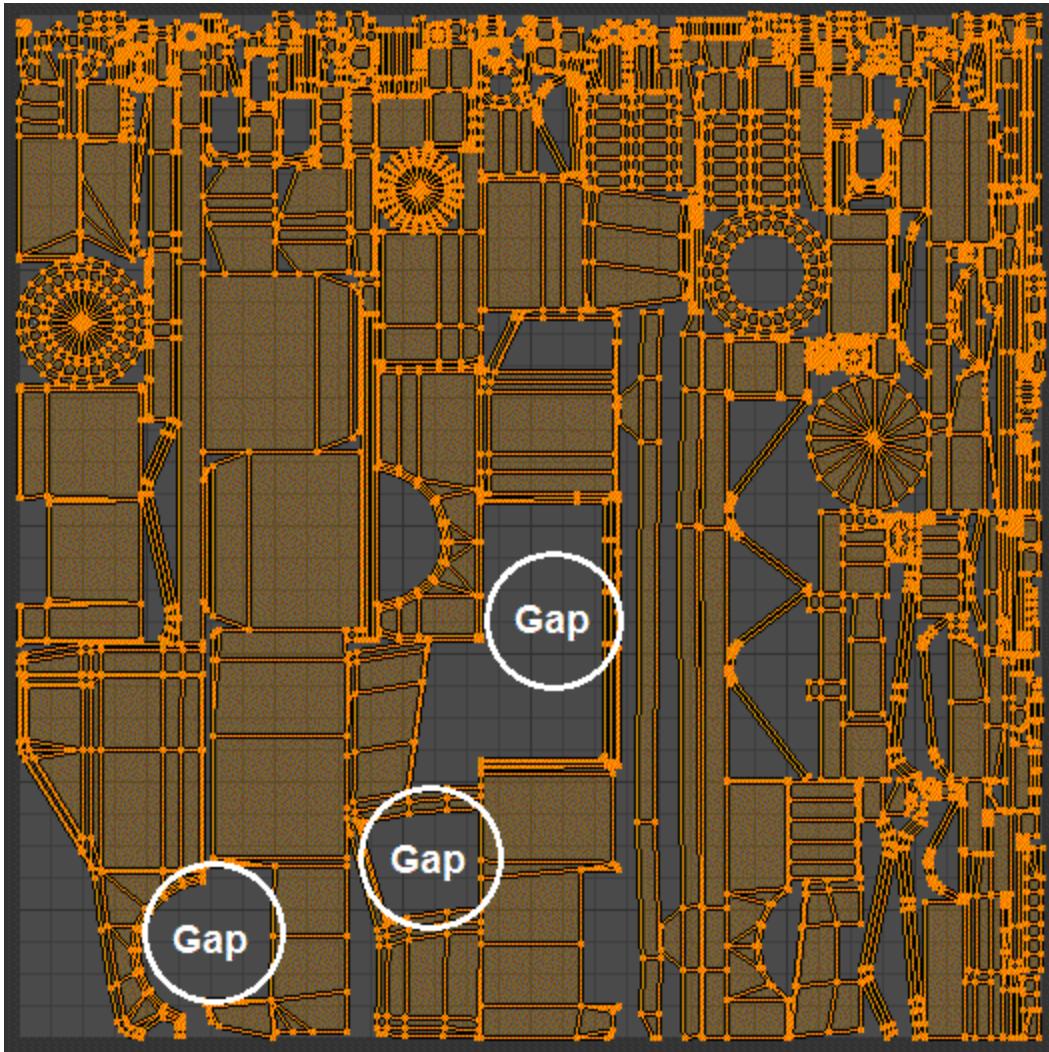
A simple way to unwrap any model is to select everything and then press U. When the **UV Mapping** menu appears, select **Smart UV Project**. You can keep the default options and press **OK**:



In your **UV / Image Editor**, you can now see the unwrapped UV map. Technically, this will work for our project. If you'd like, you can just keep it the way it is. However, there are a few reasons why it's not ideal.

<pagebreak></pagebreak>

First of all, there are a number of large gaps (empty spaces). Since we want to be efficient with our texturing, we want to use as much of our space as possible:



The second reason that this isn't ideal is that it's not intuitive. In other words, you can't look at the UV map and immediately identify the hood panel or the tailgate. If you're going to be painting decals on later, it's quite helpful to be able to look at your UV map and understand what the various parts are. We'll correct that by manually unwrapping the model.

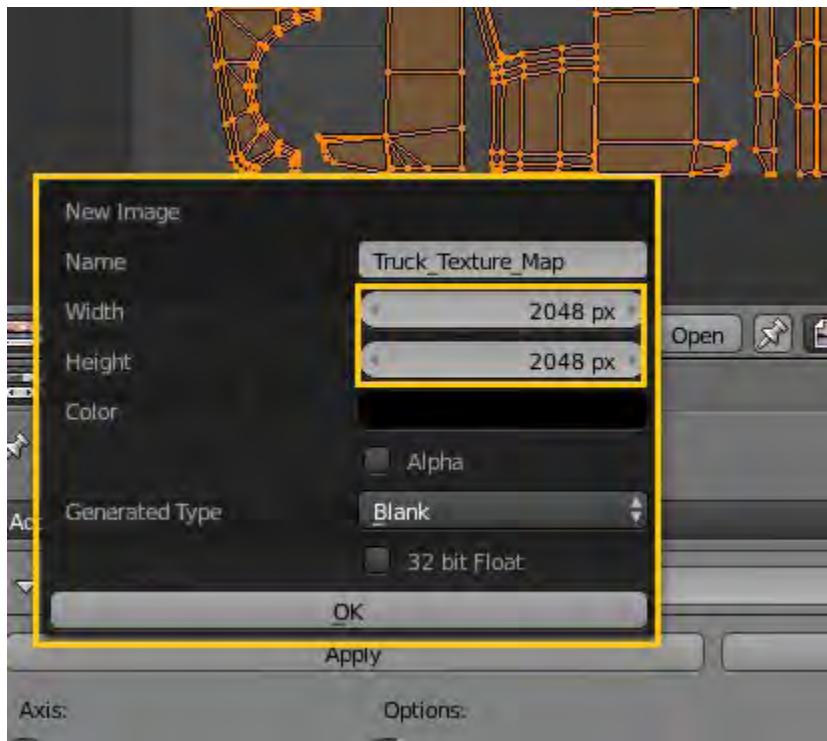
First, however, we'll use this UV map and practice baking our textures to it.

Let's create a new image in the image editor. I'll call it `Truck_Texture_Map.png` (but of course, you can name it anything you want). I'll make it 2048x2048 pixels:

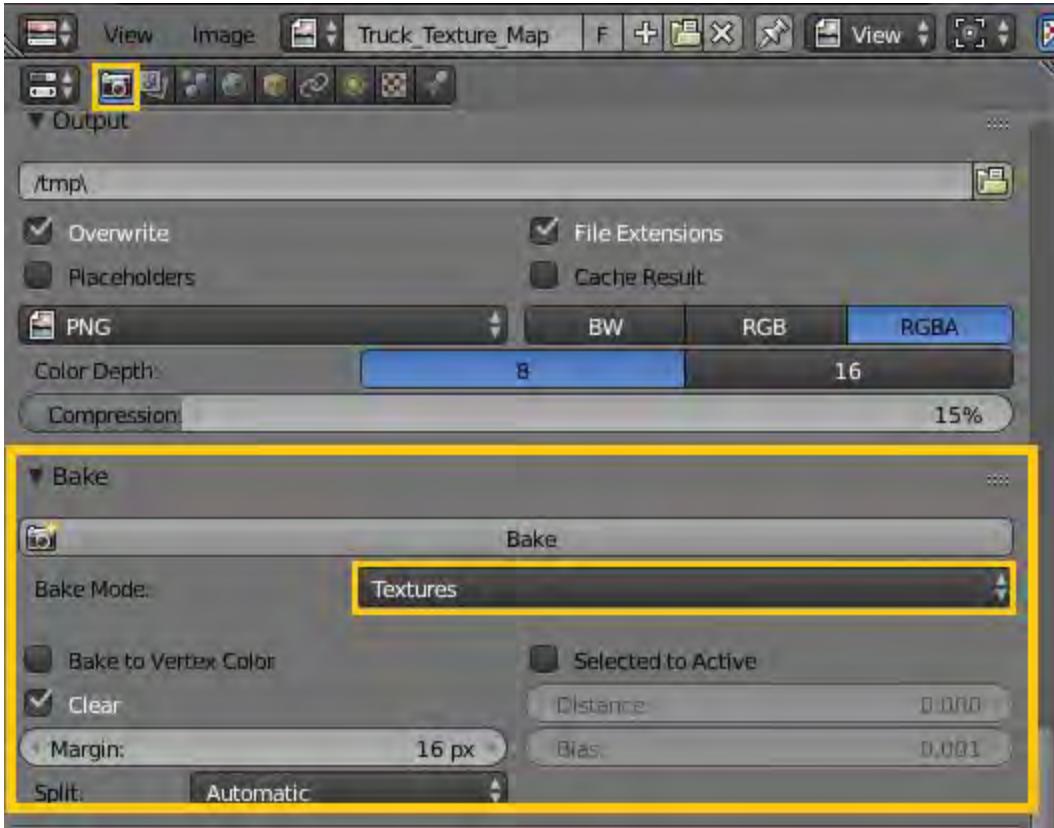
Note

An image that's 2048x2048 is typically referred to as a 2K texture. For many applications, a square texture that's a power of 2 (such as 512, 1024, 2048, 4096, and so on) is typically more efficient for CPU/GPU memory use. Naturally, the larger an image is, the better resolution you're going to get for your model. However, a larger image will also take up more memory and increase your render times. If

you're uncertain what texture size to use, it's best to start by creating a larger texture. You can always scale it down later if it bogs down your software.



Once you've created your new image, you can save it out to any location that you choose. Then, switch to your **Rendering** tab and scroll down to your **Bake** options:

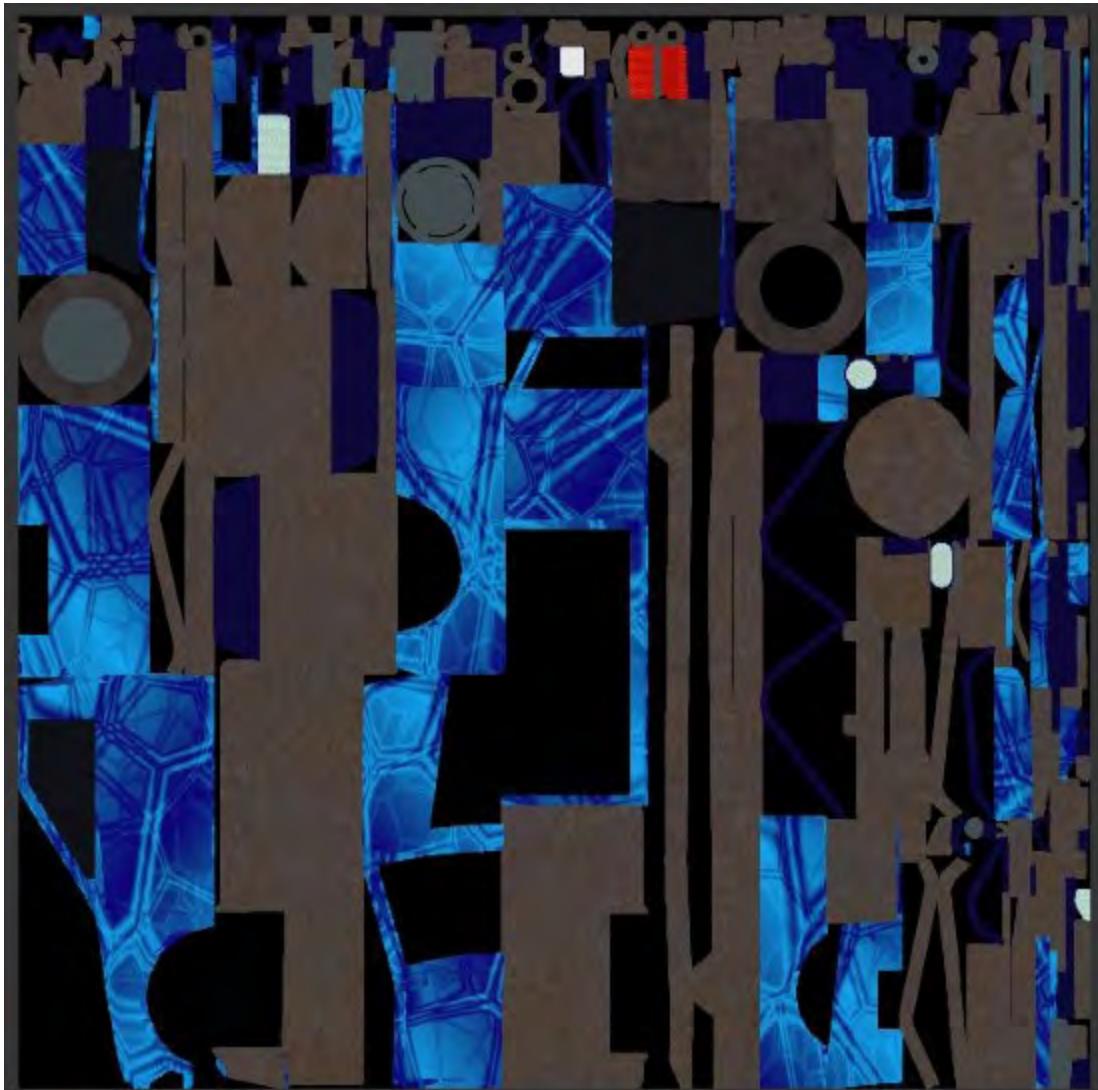


There are plenty of options here, but we'll keep it simple for now. Select **Textures** from your **Bake** mode drop-down menu. Then press **Bake**.

In your **UV/Image Editor**, you will now see that Blender has baked the procedural textures we created to an image.

This is useful for many reasons and necessary if you plan to export models to another format. The procedural texture that we'd been using for the paint, for example, was based on a mathematical pattern (in this case, the **Marble** option). However, that particular pattern (and the generated coordinates that go with it) only work within Blender. Another software program won't understand what a **Marble** texture is or how to apply it.

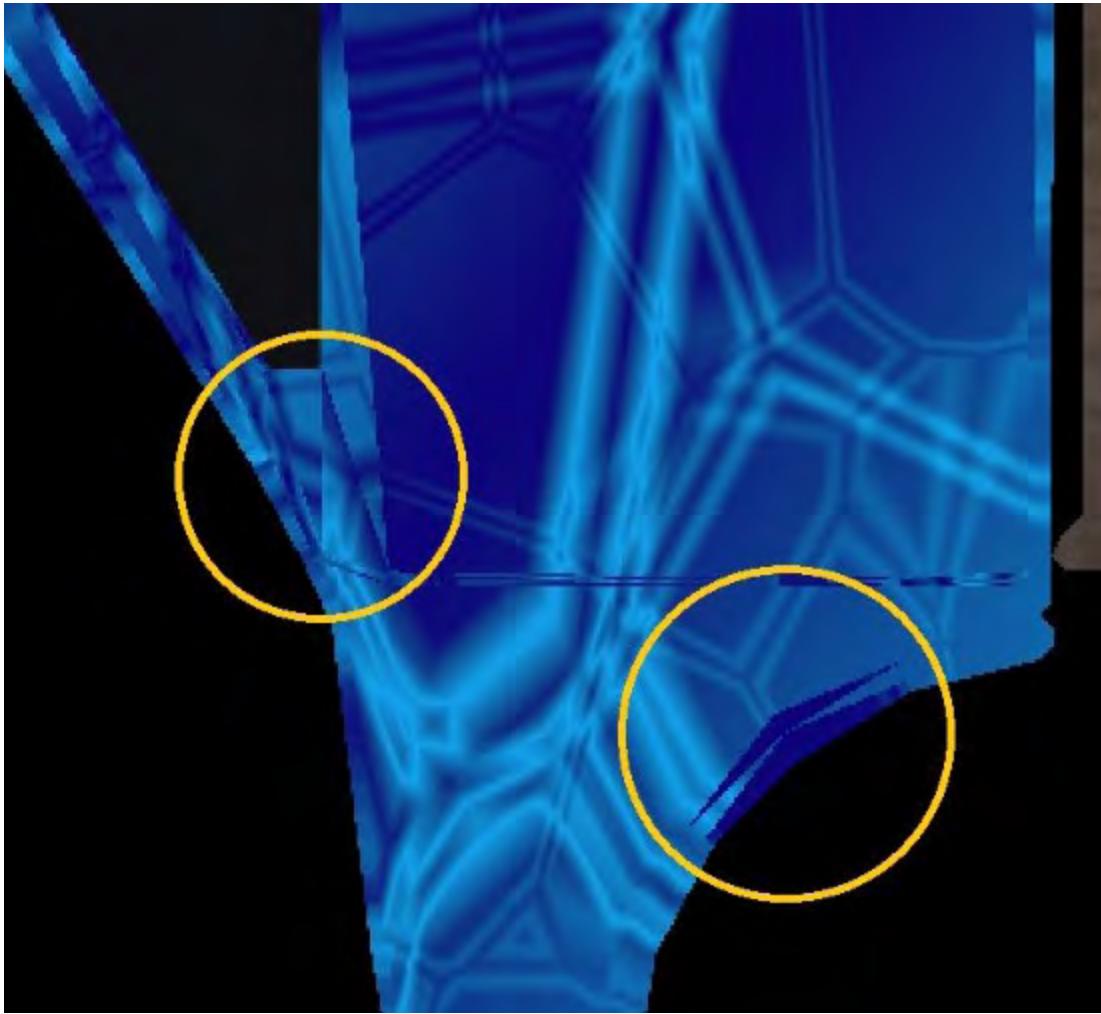
By baking that texture out, we've converted the pattern and material data to a universal format—2D images. This texture can now be opened in (almost) any other 3D or game software:



However, we do have a slight problem with the way we baked it. Because we used a mirror modifier on the model, Blender tried to bake both sides of the truck to the same 2D texture space. However, the paint pattern is not symmetrical (it's different from one side to another). There's no way to bake both halves of the truck to the same 2D space.

<pagebreak></pagebreak>

Therefore, you will see a number of overlaps and problems in the image as Blender tried to combine both sides:

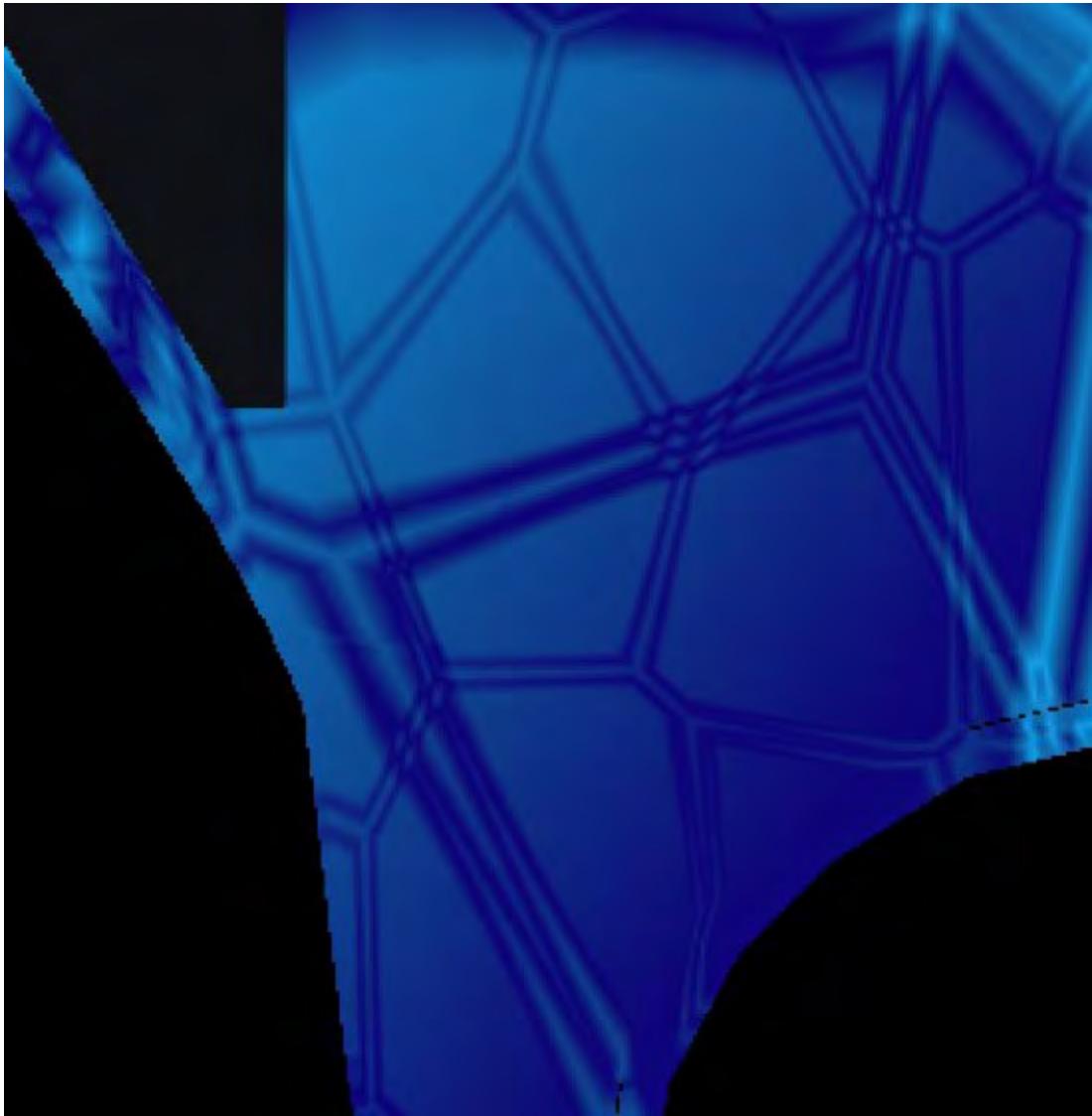


Of course, one way to fix this would be to turn off the **Mirror** modifier on our truck:



<pagebreak></pagebreak>

That will work, in the sense that you won't have odd artifacts during the bake:



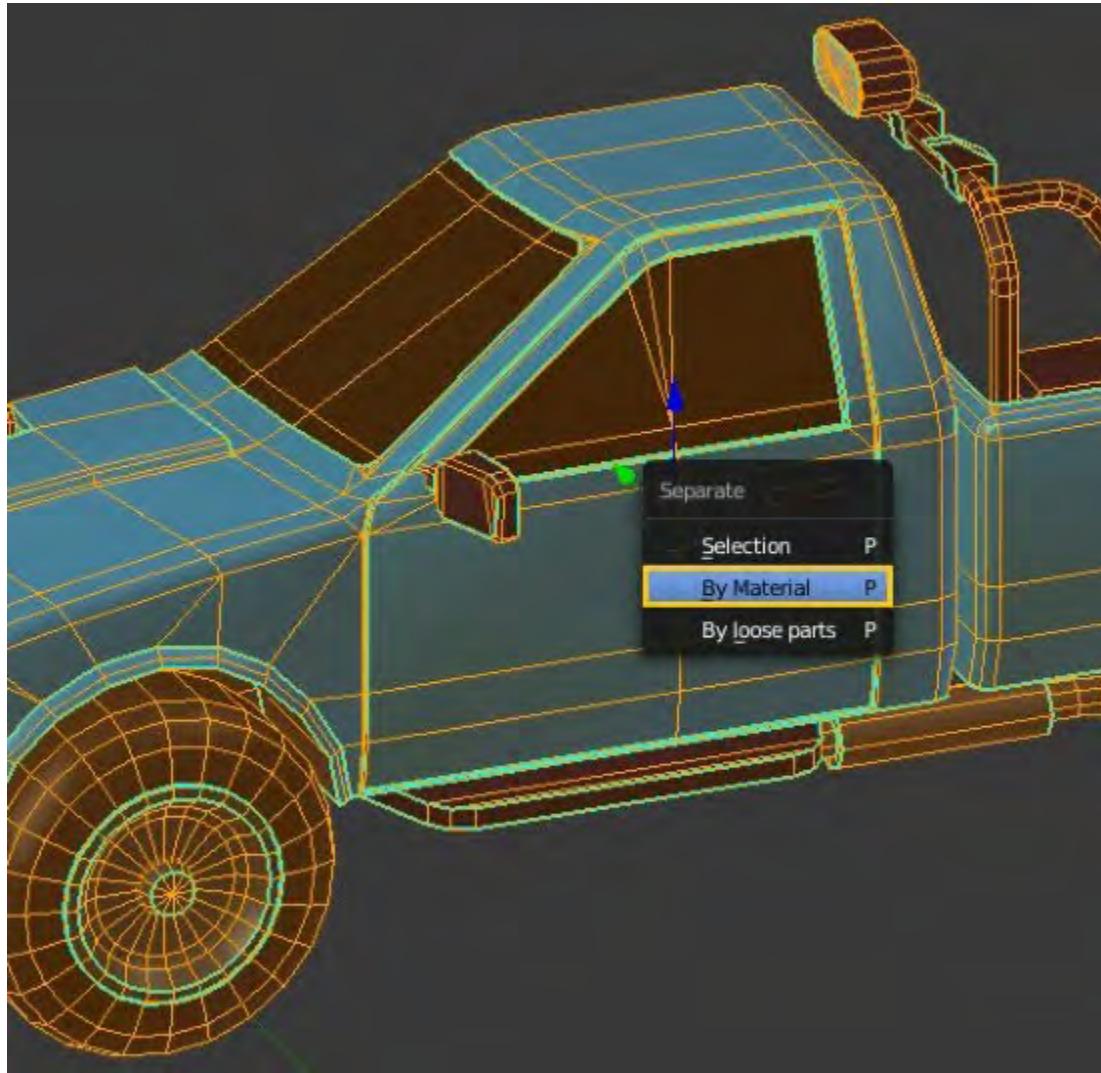
Of course, it also means that both sides of the truck will be identical, which wasn't the way we created it originally. It doesn't necessarily look bad, but it's just something to be aware of. You can get away with it when it comes to a paint pattern, but it will pose a problem in other areas.

For instance, you may want different decals on one side of the truck than you do on the other. Even if you want the same decals (like a number or sponsor logo), they need to be mirror images of each other (so the text faces the right way on both sides). For a lot of applications, you'd need to unwrap both sides of the truck separately. That's what we'll do here.

<pagebreak></pagebreak>

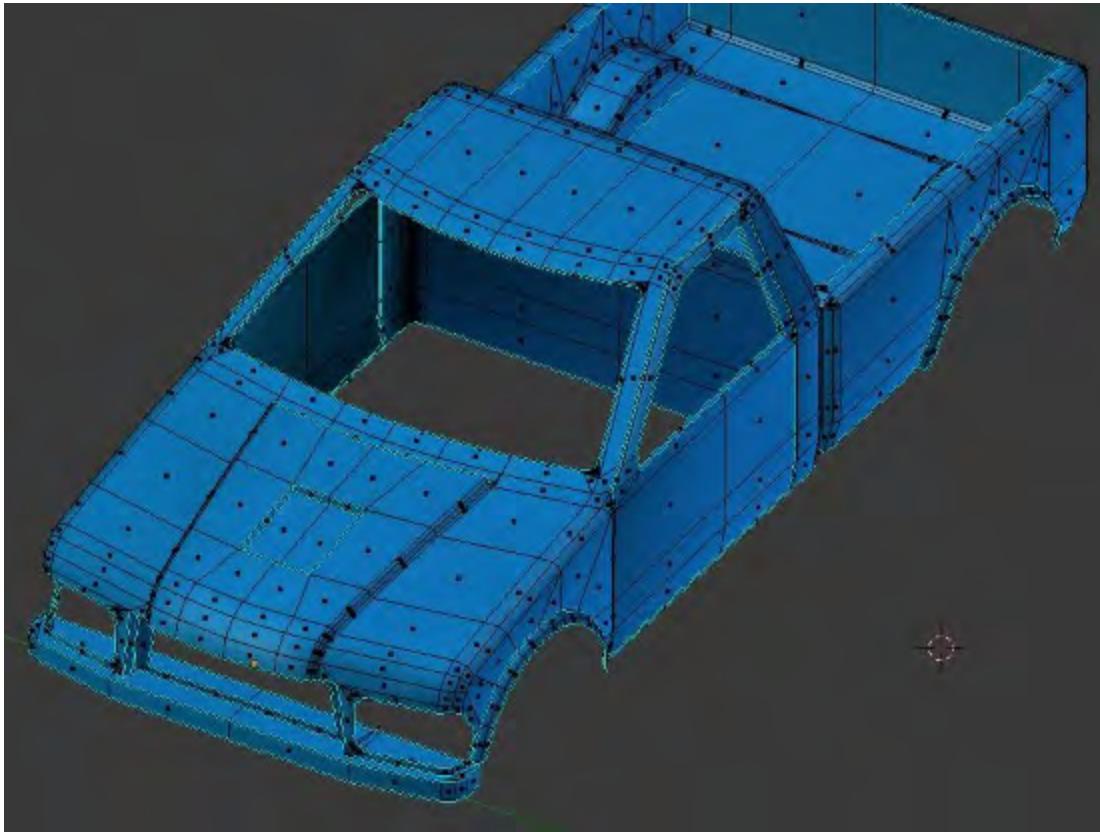
First, I'm going to separate the model into different objects. We'll use the P key again, just as we have before. This time, however, we'll select the **By Material** option. Blender will automatically separate

the truck into different objects based on the material assigned to each part. For instance, all of the painted body surfaces will now be one object, and all of the windows will be another.



<pagebreak></pagebreak>

For those parts that we don't want to mirror (like the main body), go ahead and apply the mirror modifier:



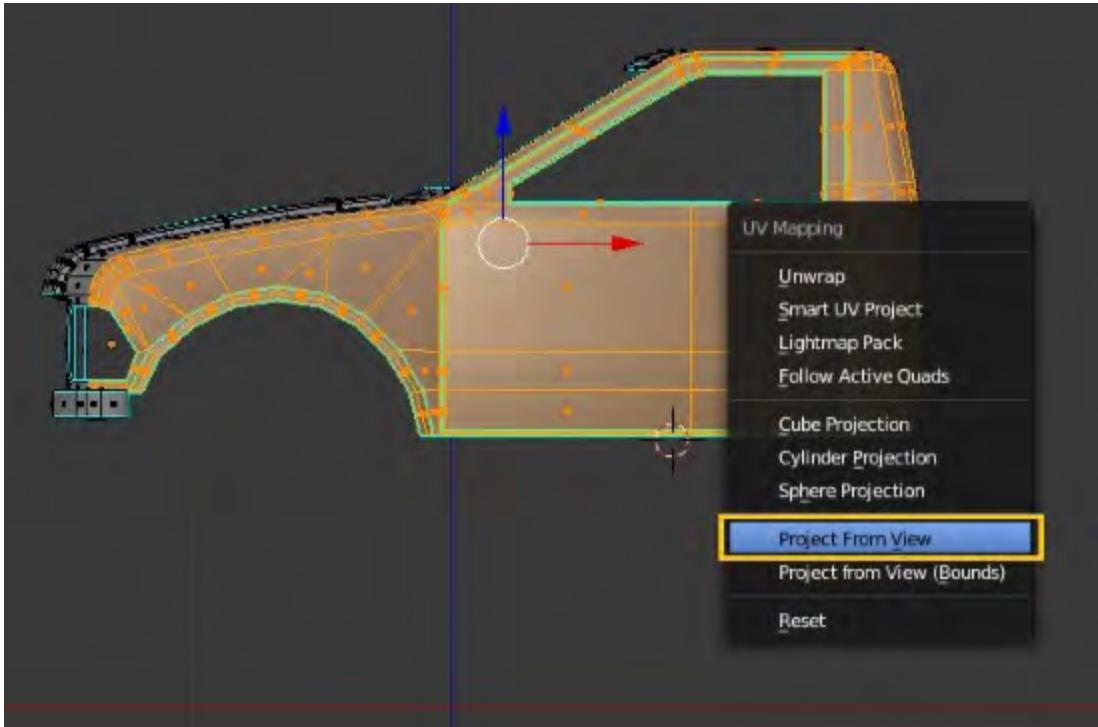
<pagebreak></pagebreak>

Now, we'll go through and unwrap these UVs.

From the side of the cab, I'll pick all of the polygons that are directly facing us and press the U key again. This time, however, I'll pick **Project From View**. The selected faces will now be unwrapped exactly as you are looking at them.

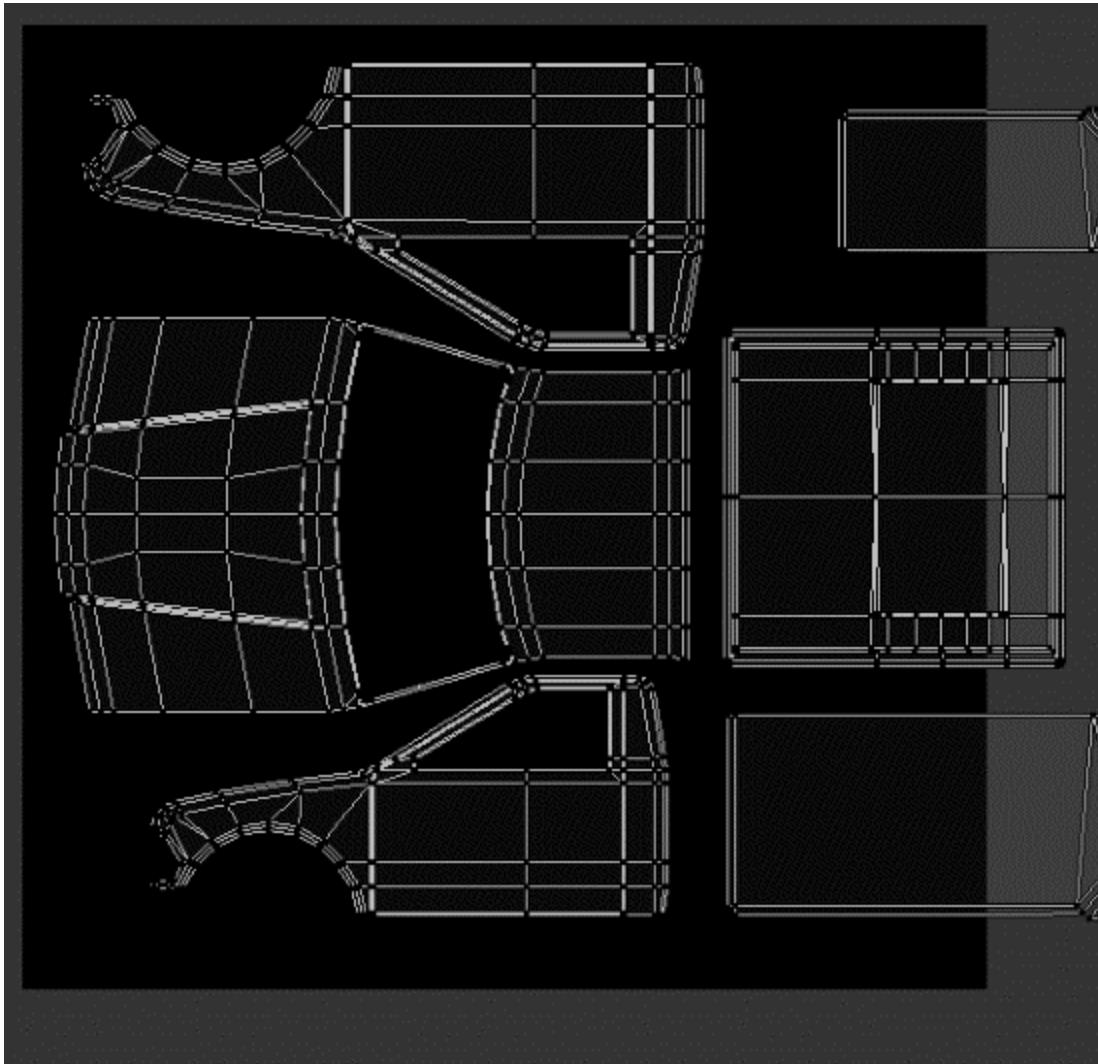
Note

The Project From View option is rarely perfect, since not every face is directly pointing at you when you unwrap it. Thus, there will be a tiny amount of distortion introduced into the UV. For objects like our truck (which have largely flat sides) the effect is negligible.



<pagebreak></pagebreak>

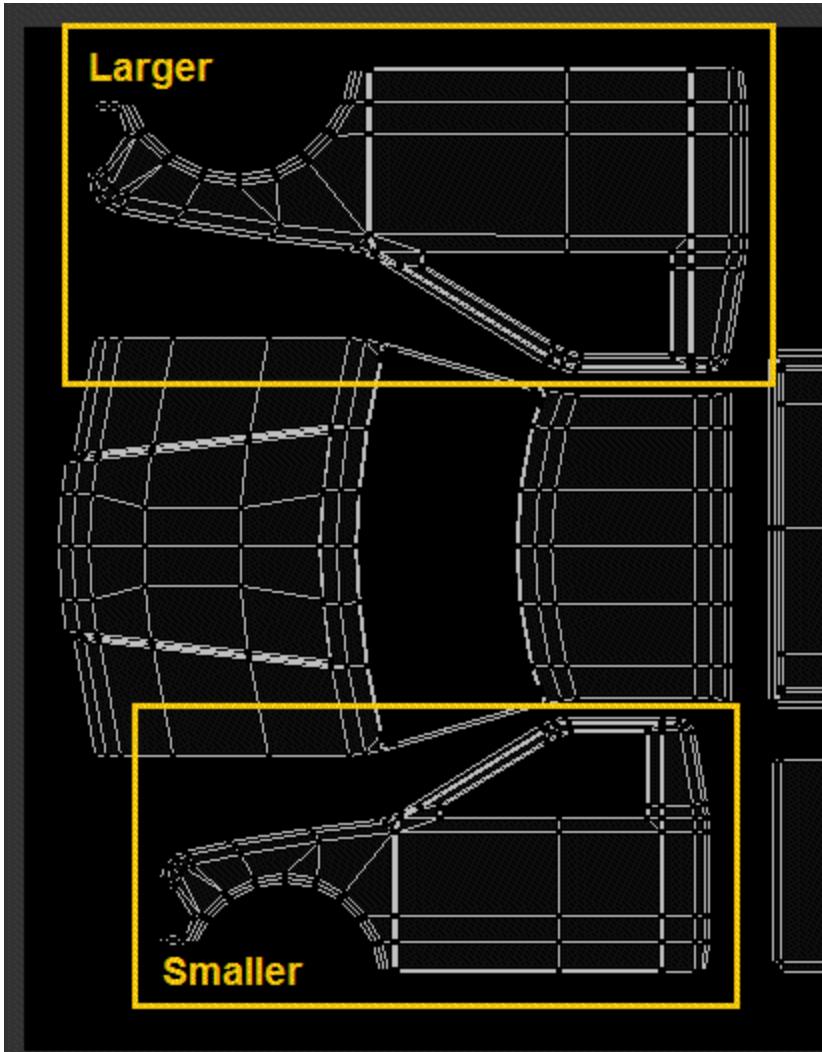
You can repeat this process for the other parts of the body and move the resulting UVs around until they form a rational pattern:



This result still leaves a lot of unused texture space, but it is far more intuitive than the **Smart UV Project** option we used originally. You can look at these UVs and immediately identify the various body panels. That will be extremely helpful when painting textures on later.

<pagebreak></pagebreak>

One downside to unwrapping this way is that you will occasionally have different sizes for your UVs. In other words, two parts of the model that should be the same size (like the truck's doors) may have different sized UV maps. That means that ultimately, one of the doors will have a different texture resolution than the other, so we want to avoid that:



<pagebreak></pagebreak>

One way to fix this is to add a new image in our **UV/ Image Editor**. Under the **Generated Type** option, we'll pick **UV Grid**:

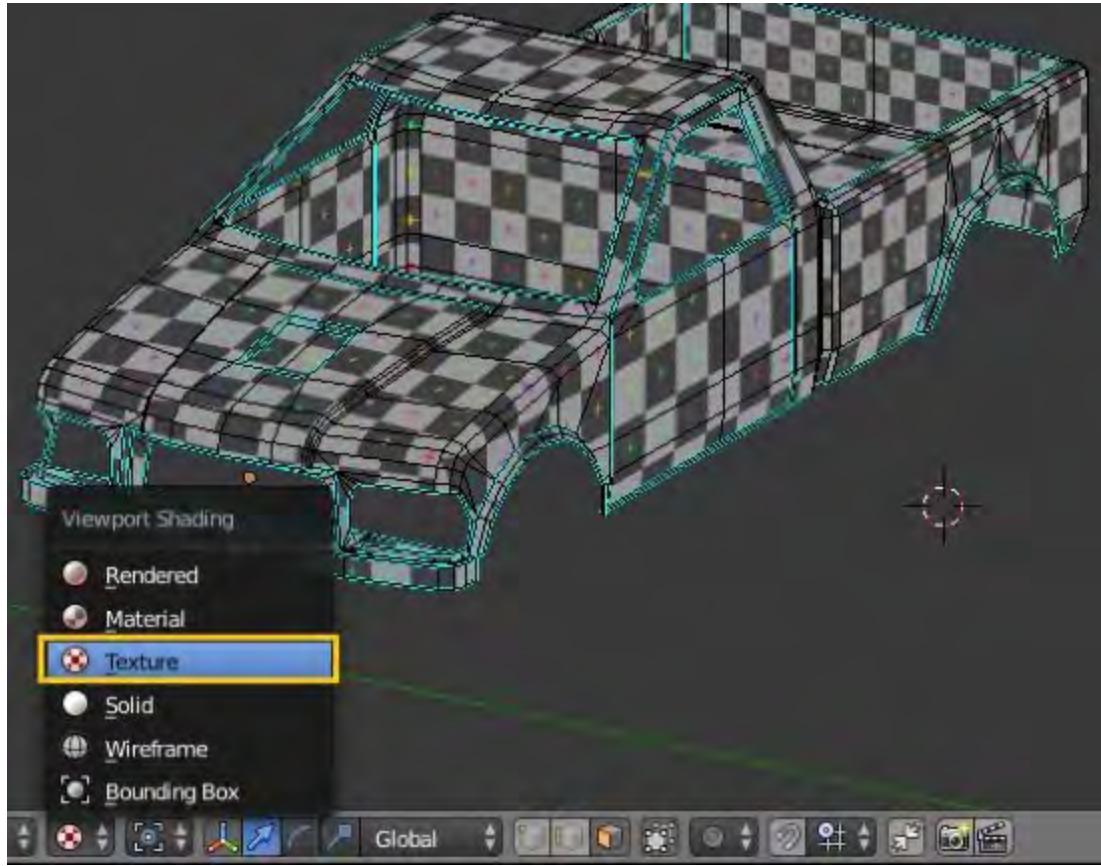


<pagebreak></pagebreak>

Now, you can switch your **Viewport Shading** to **Texture** and take a look at the UV grid. The purpose of the UV grid is to help you check your UV maps (to make sure they're not stretched or improperly sized):

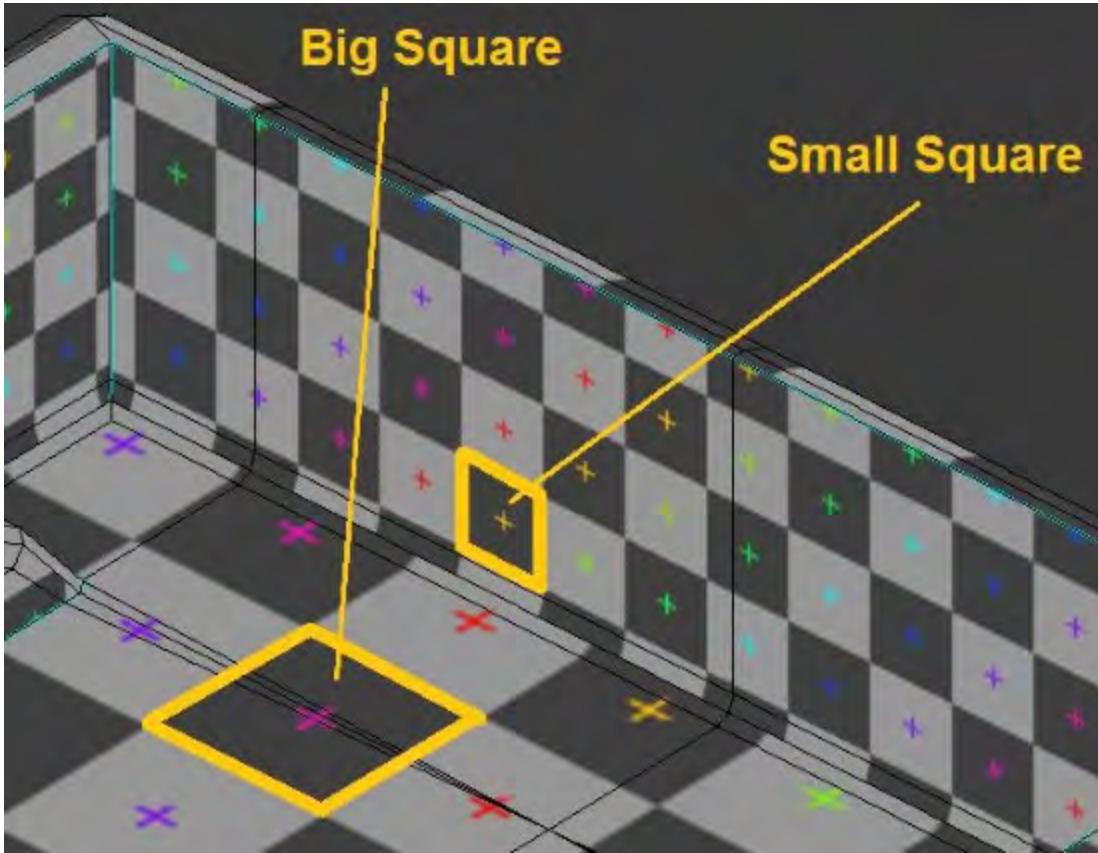
Note

You can also use the **Area Stretch** option to do this.



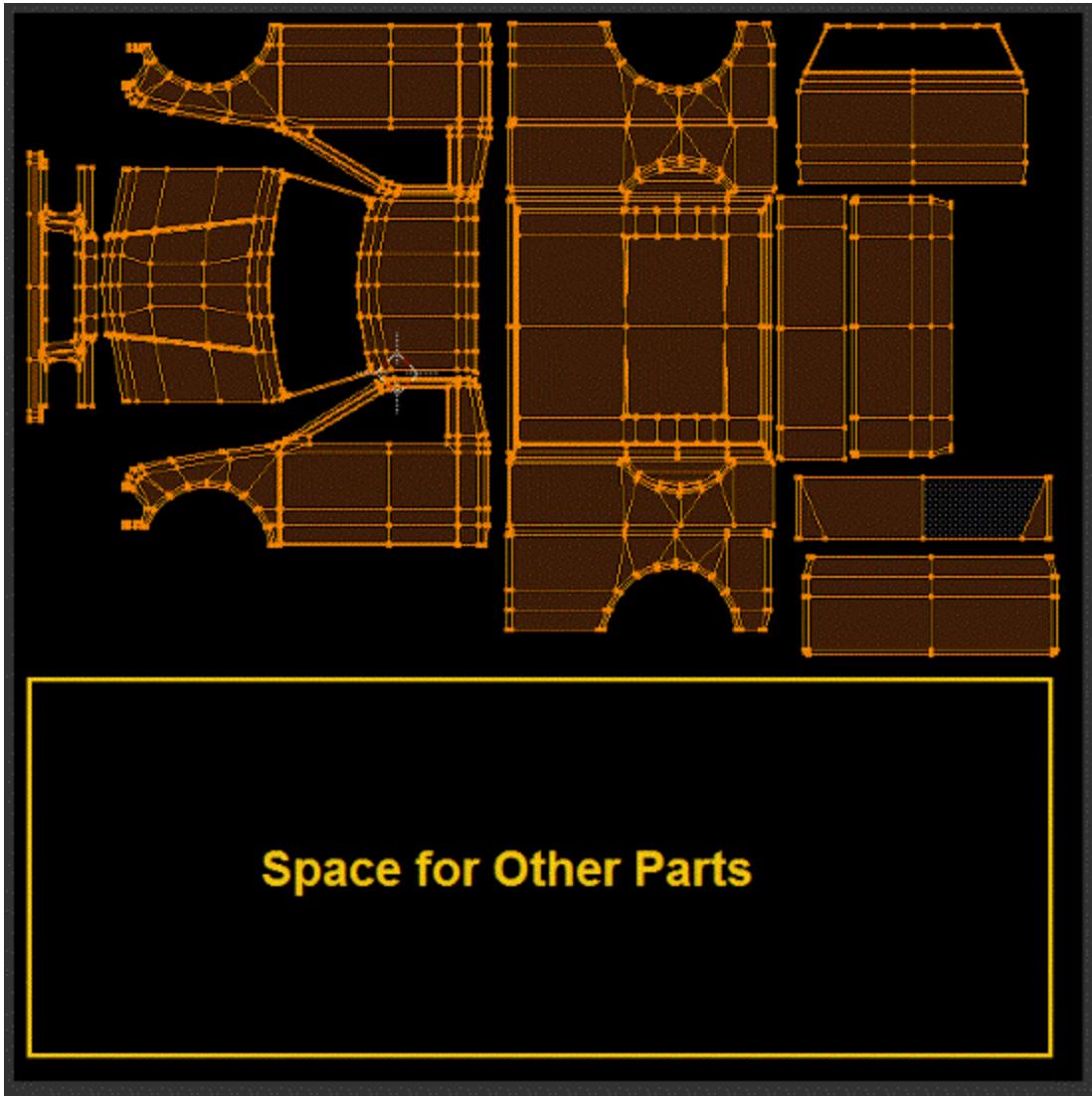
<pagebreak></pagebreak>

For instance, when I look at the inside of the tailgate and the bed of the truck, I can see that I have a problem. The tailgate UV map is too large by comparison, resulting in squares that are too small:



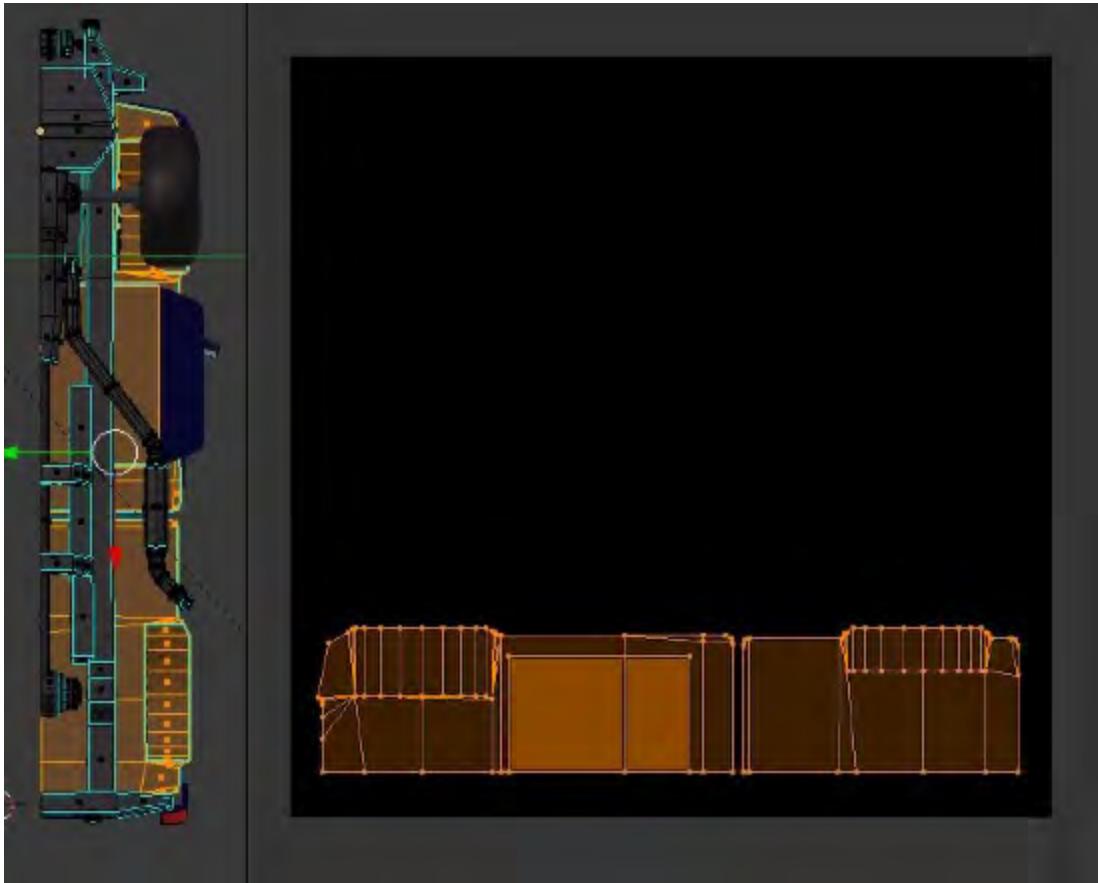
<pagebreak></pagebreak>

Using this method, you can adjust the size of your UV maps. You can also use the Average Island Scale option to do this automatically (Ctrl + A in UV editor). When you've got them all the correct size (or close enough), you can arrange them more logically. Make sure that you leave space for the rest of the truck inside of your square image window:



<pagebreak></pagebreak>

Next, we'll move on and unwrap the frame and mechanical parts at the bottom of the truck. First, we'll unwrap the flat part that makes up the actual bottom of the truck.



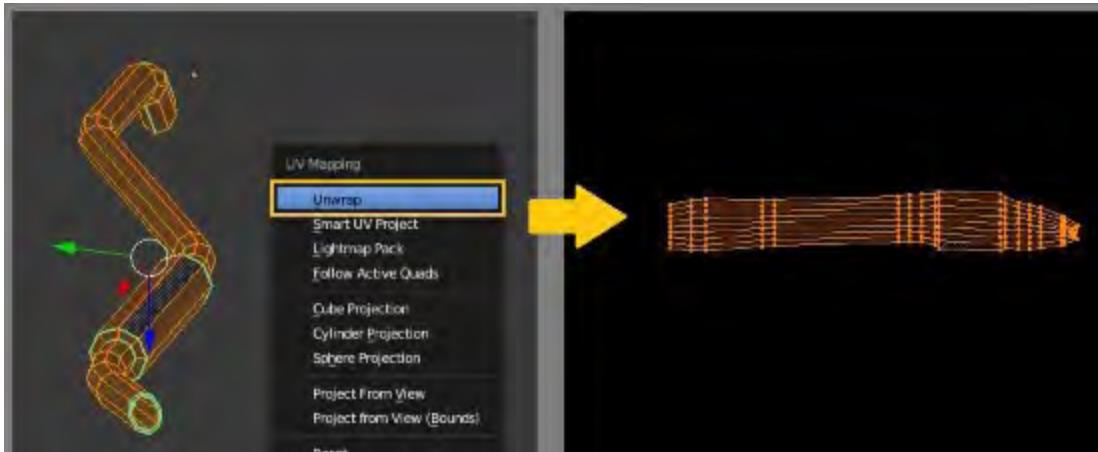
<pagebreak></pagebreak>

Next, we'll unwrap the exhaust pipe. You can do this by selecting one of the long edge loops along the pipe and marking a seam. A seam tells Blender where to cut the model in order to unwrap it. By specifying which edges are going to be seams, you gain a lot more control over the unwrapping process. In order to mark a seam, you simply hit **Ctrl + E** and select **Mark Seam**:



<pagebreak></pagebreak>

Now that you've marked your seam, you can unwrap with the first option (**Unwrap**):



The resulting UV map may look a bit strange, but it does make sense when you think about it. Because the different parts of the exhaust have different diameters, the UVs will have to be a little taller/wider in some places to make up for that.

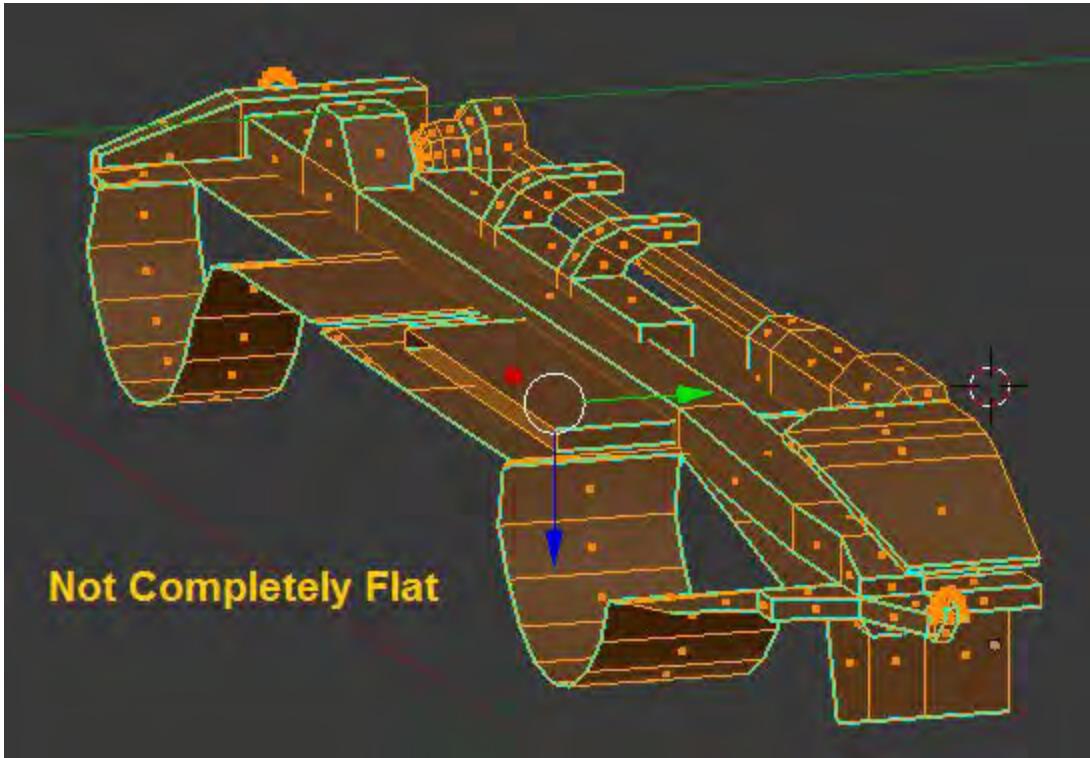
If you like, you can continue unwrapping the various bottom parts by hand, using seams, project from view, and more. This process can be quite time consuming, but it's really the best way to do it.

<pagebreak></pagebreak>

However, sometimes you'll want to save time and you may not care about 100% accurate mapping. In this case, there are a few little tricks you can use:

Note

What I'm about to demonstrate would not be considered a good technique. It's a quick way to get the job done (and therefore worth knowing about), but I advise you to use it sparingly.

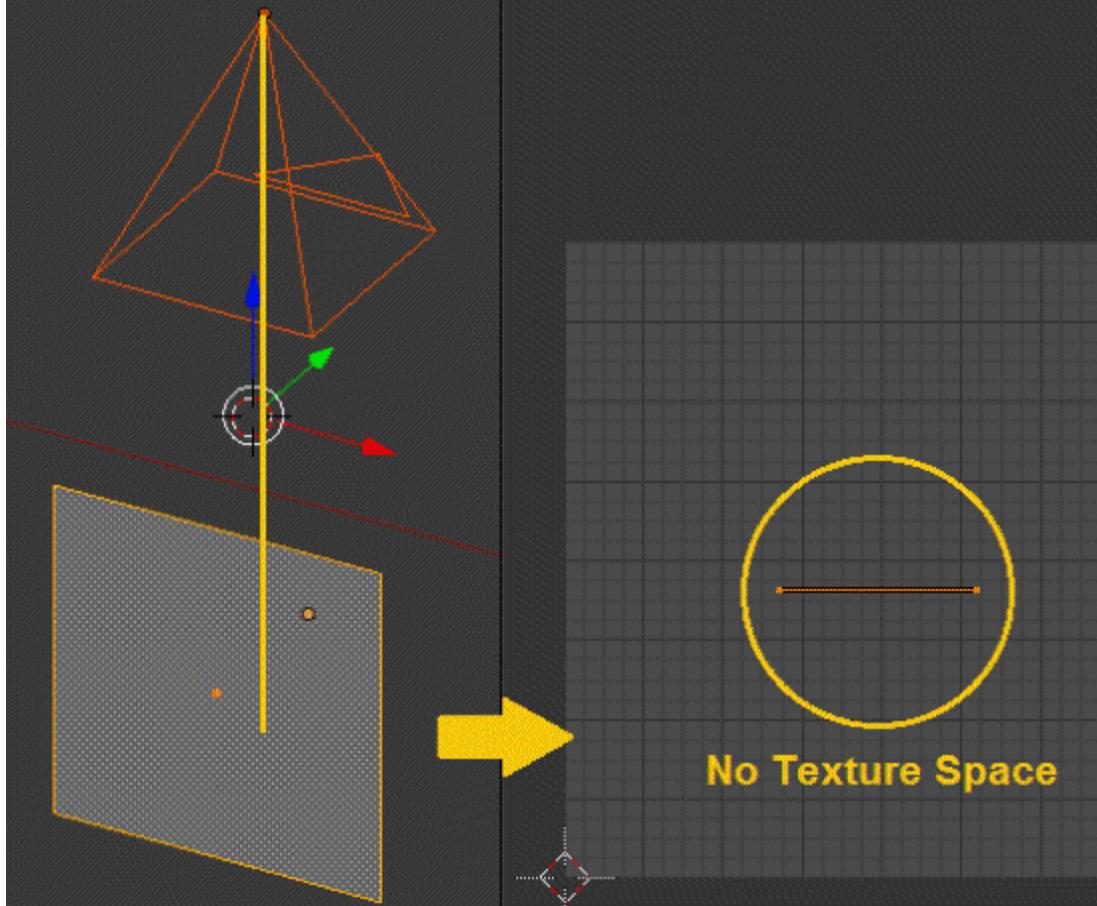


You can see that bottom of the truck is almost flat. In other words, it looks like it could mostly be unwrapped from the bottom view. That's very tempting, as it would save a lot of work. However, you can see that there are a few little parts that would need to be unwrapped from the side view.

<pagebreak></pagebreak>

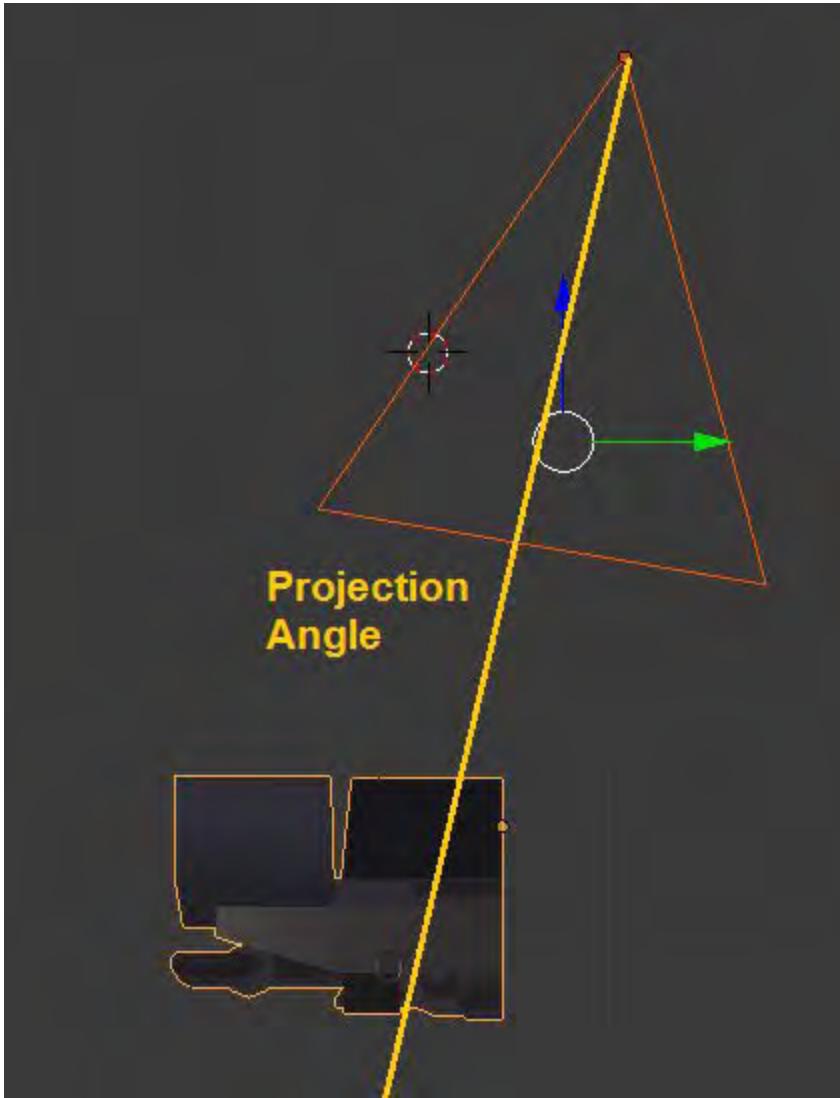
Naturally, unwrapping a part from the wrong view will result in having no texture space available for it:

✓ Improperly Unwrapped from Top View



<pagebreak></pagebreak>

To avoid that problem, you can unwrap the model from almost the bottom view. If you unwrap it, say, 15 degrees off the axis, you should get an adequate result:

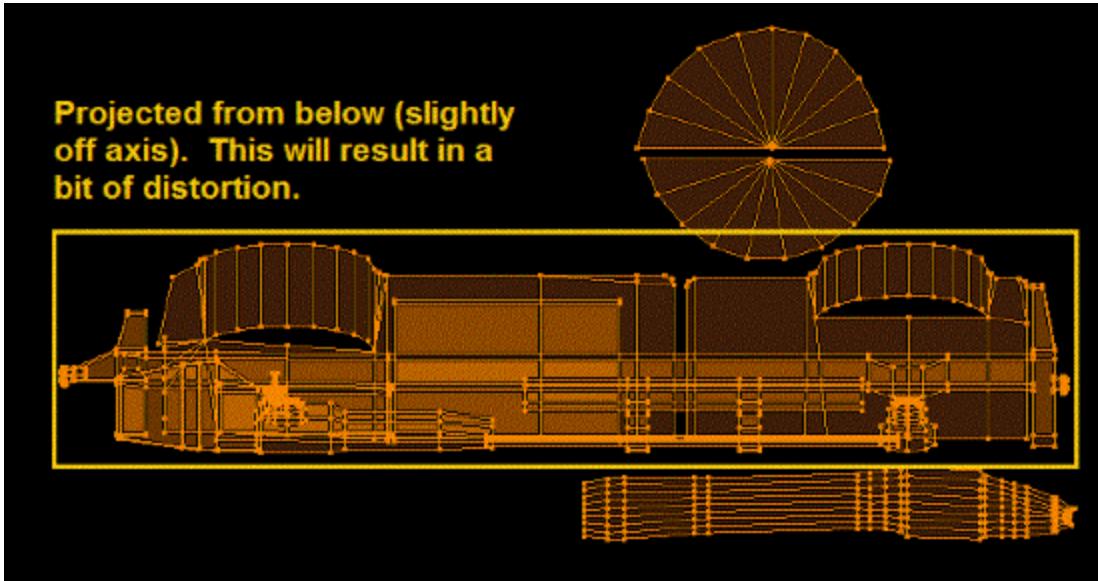


<pagebreak></pagebreak>

You will still have a bit of distortion:

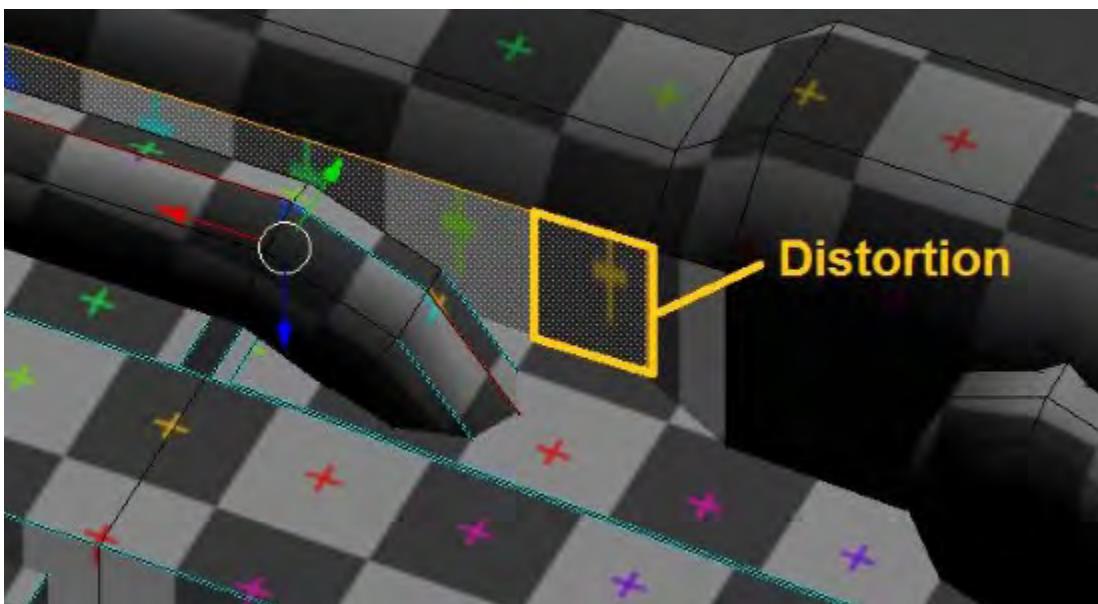
Note

In this case, I only rotated slightly on the X-axis. That means that there is no texture space available for either the very front or very back polygons of the truck frame. To correct that, you'd want to be slightly off-axis on both the X and Y axes.



Let's look at the downside of this technique. As you can see, the portions that aren't completely flat (from the bottom view) have a little bit of distortion to them. You need to decide whether you can live with that or not. If not, you'll want to go through and unwrap each part manually.

In this case, however, I think we can let it slide. This is the very bottom of the truck. It will rarely be seen on screen for any length of time (maybe during a jump or roll), and it will be blocked by the exhaust pipe and/or shadows. Again, it comes down to how much work you're willing to put in and how much realism you expect to gain.



<pagebreak></pagebreak>

Next, we'll unwrap the tire:

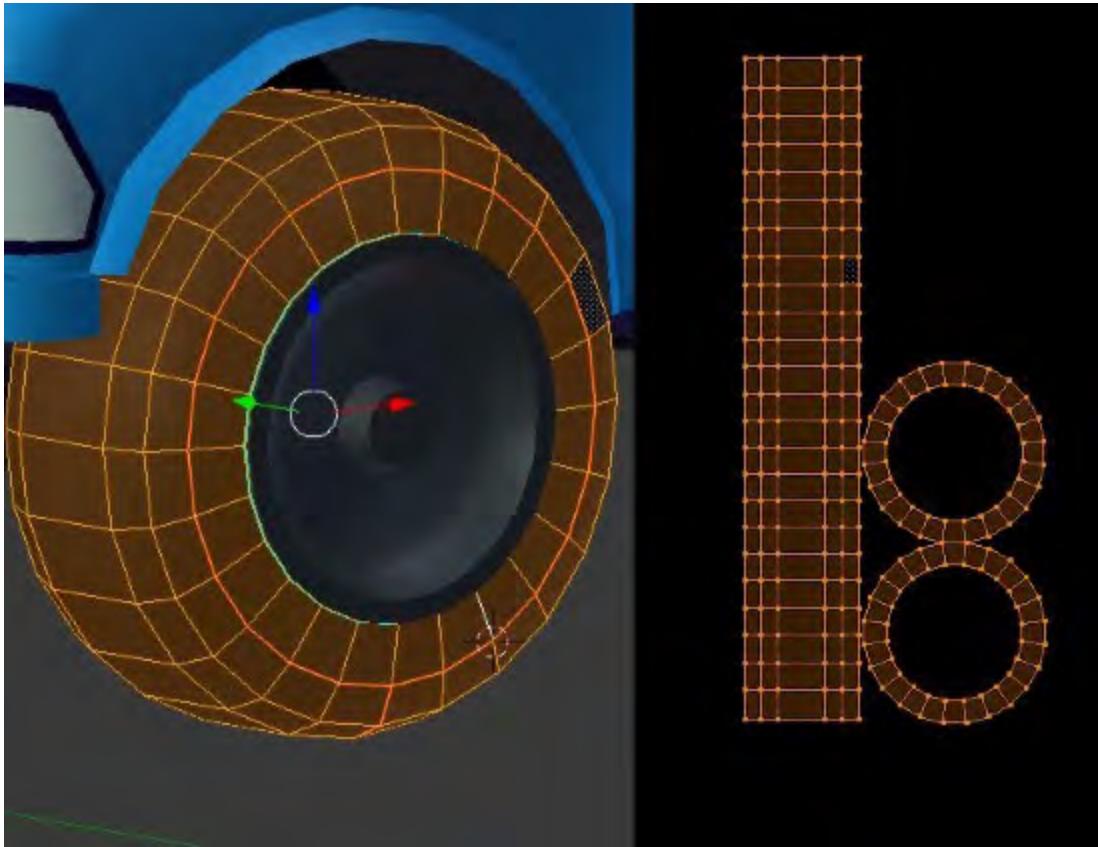
Note

I've deleted all but one wheel/tire so that all four of them eventually use the same texture space. Once we've UV unwrapped one tire, we can just duplicate those polygons, and they'll be assigned to the same portion of the image. This is helpful for saving space.



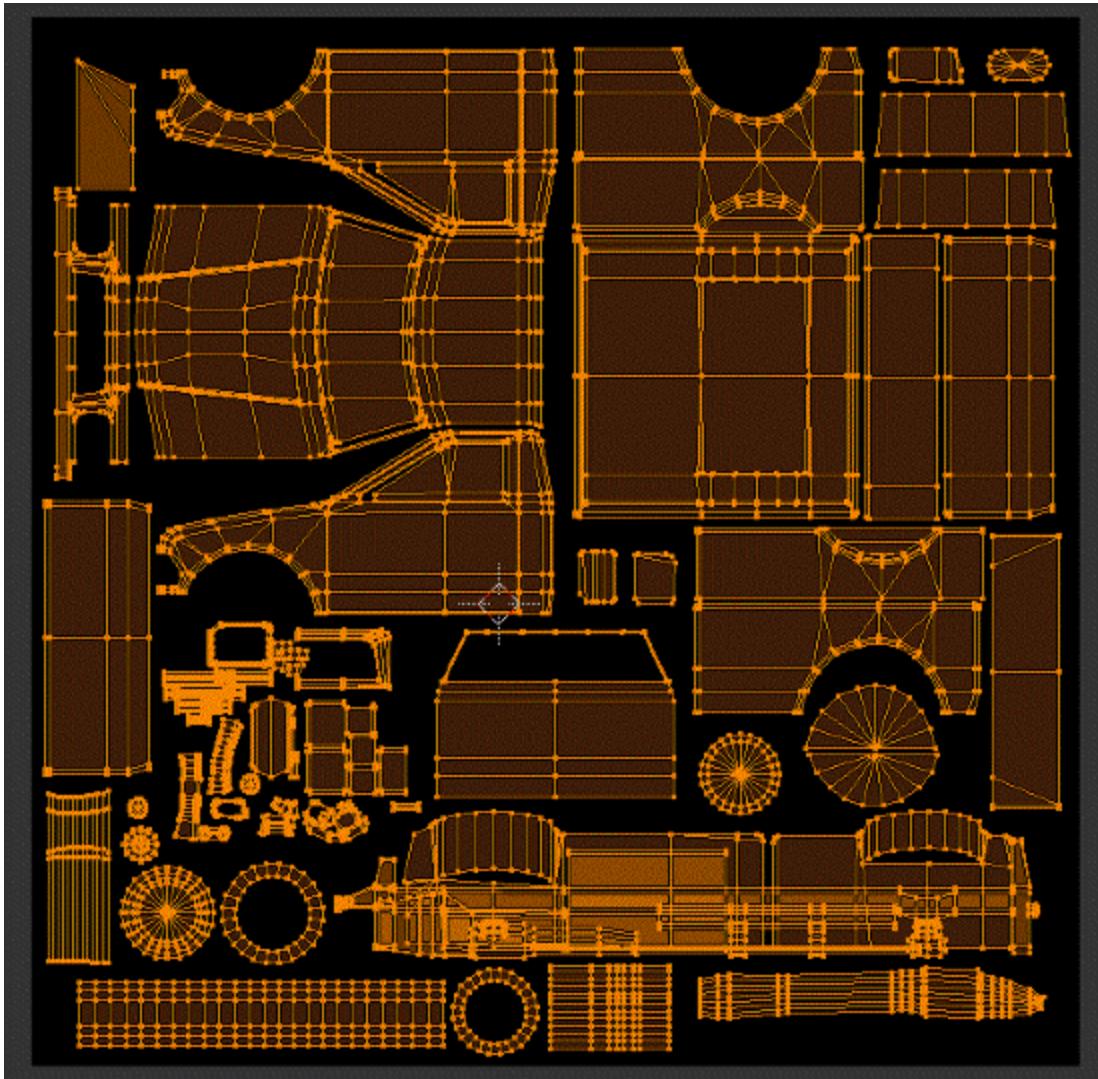
<pagebreak></pagebreak>

I've marked a seam here as well. Since we'll want to draw a tread pattern onto our image later, we need to make sure that the tire is unwrapped appropriately. Also, unwrapping the sidewall portion from the side view will give you the opportunity to add a bit of text or other details to the side of the tire:



<pagebreak></pagebreak>

Using these same techniques, you can now go through and unwrap the various parts of the truck. Here's what my final UV map ended up looking like:

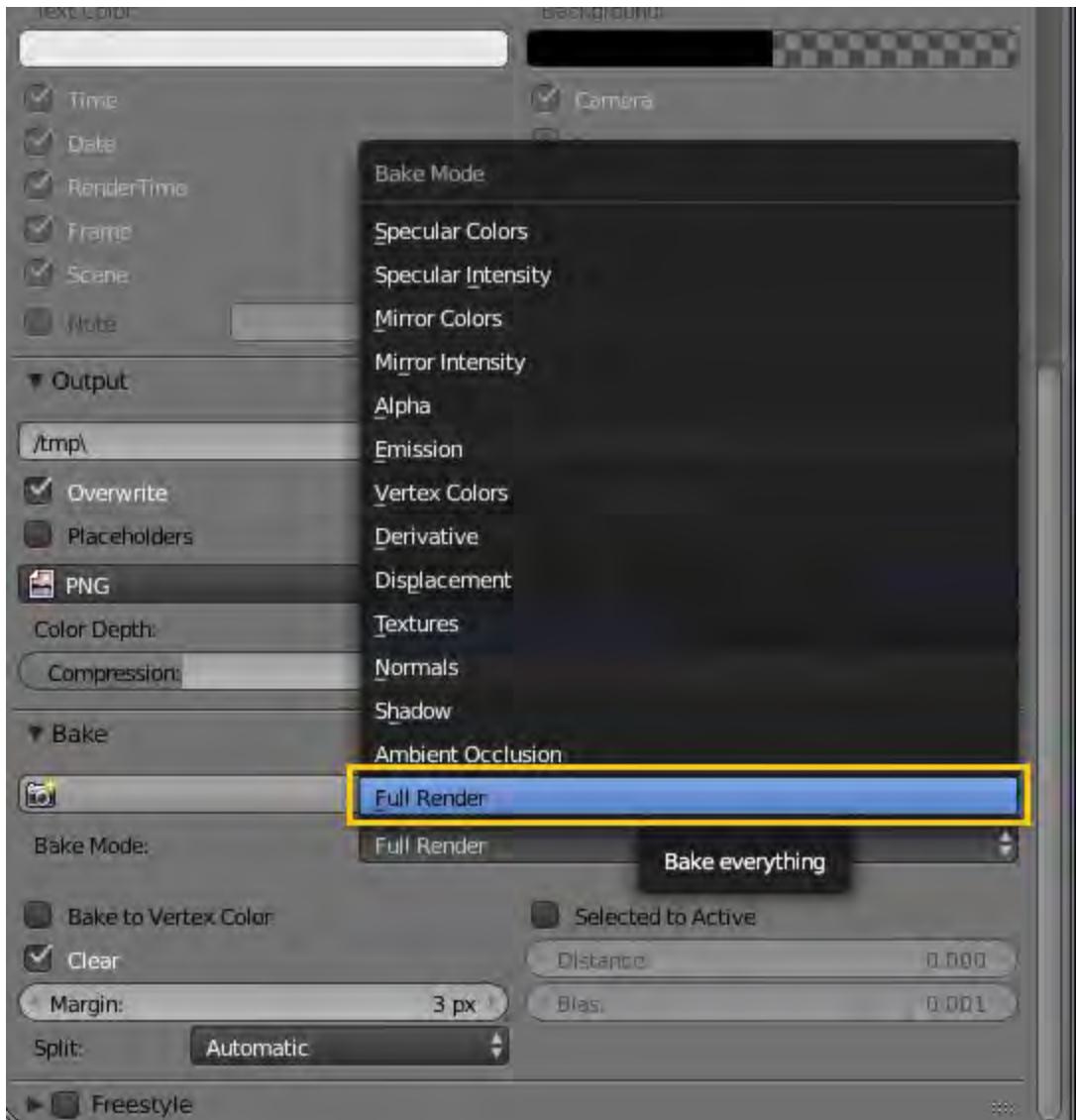


This is far from perfect. As you can see, there's plenty of empty space in there, and things are not unwrapped with 100% efficiency. UV unwrapping is an art, and you could easily write an entire book on the different techniques. Since we don't have room for that here, we've stuck to the basics. I certainly encourage you to take it further!

Now that we've got our UV map created, we can bake out our textures again. Last time, we just stuck with the **Texture** option for our **Bake Mode**. This time, just to experiment, we'll use the Full Render option.

<pagebreak></pagebreak>

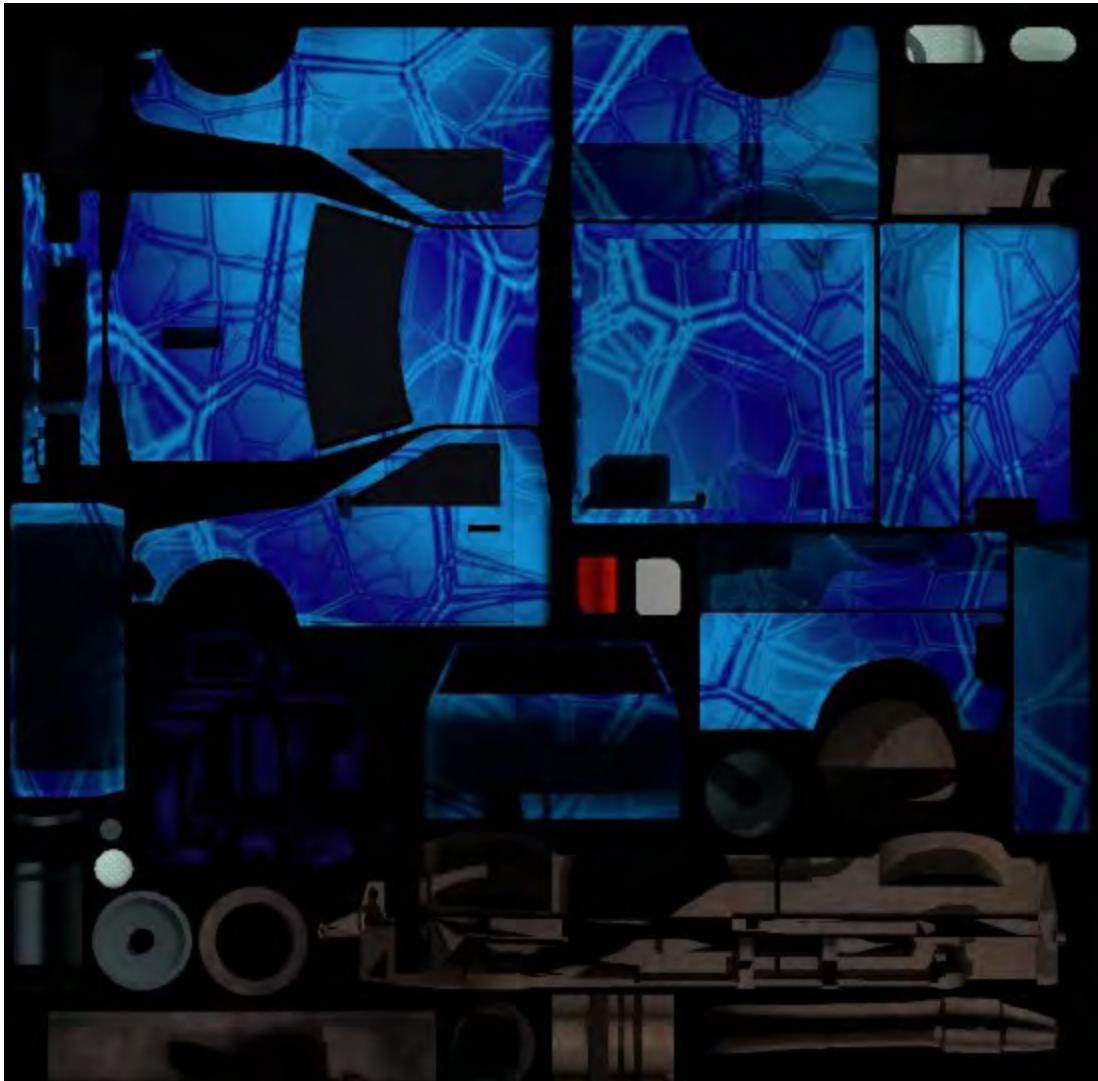
When you use this option, Blender will bake not only the textures, but the light, shadow, and other aspects of your model as well:



Make sure you add a few lights to your scene to test it out (otherwise the image will just be black). When you're ready, hit the **Bake** button.

<pagebreak></pagebreak>

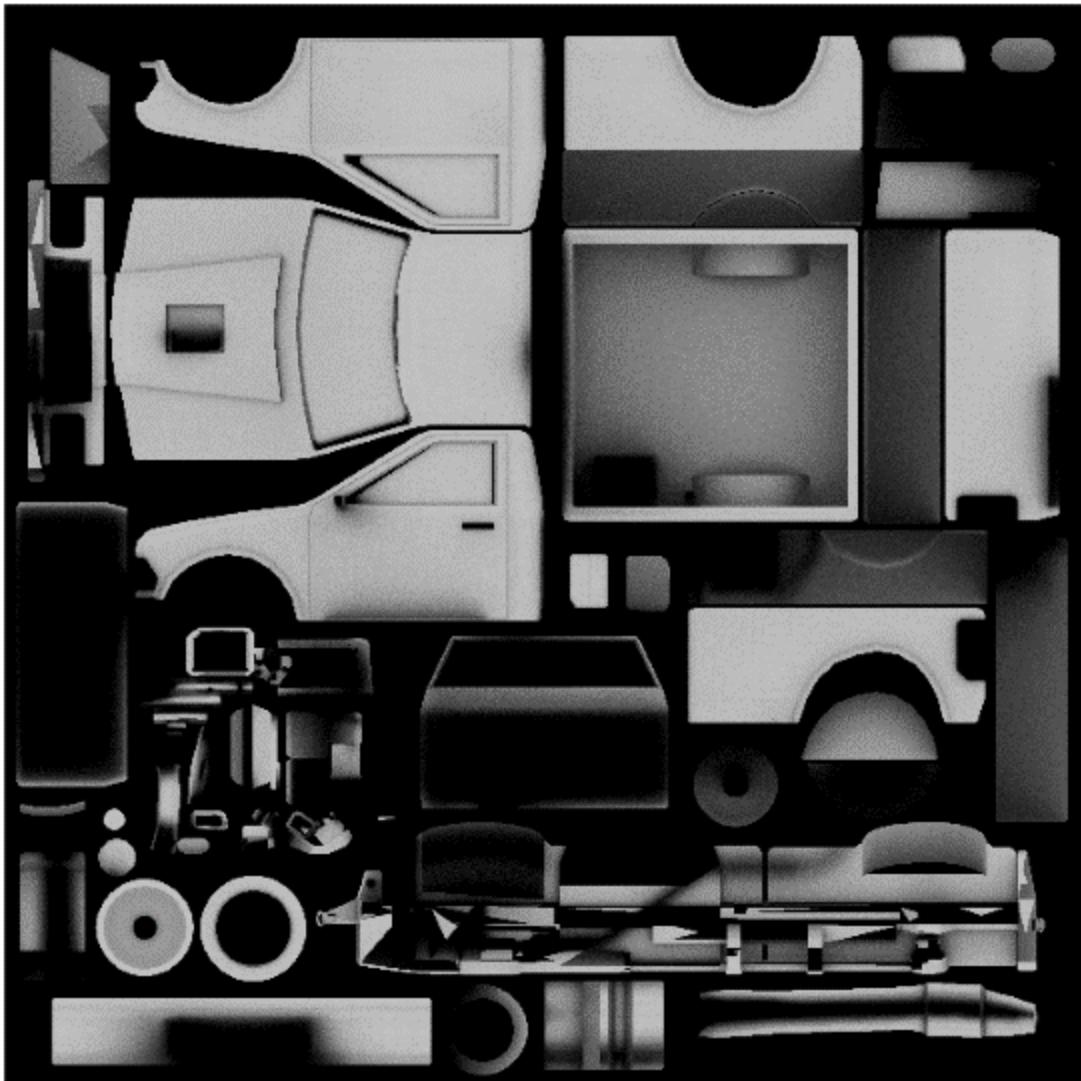
You can tell that this result looks a bit different from the first one:



Another option that you have for your **Bake Mode** is **Ambient Occlusion**. This sounds complex, but it's not. Imagine a perfect sphere encircling your model and casting light evenly from every direction (this idea is called **Global Illumination**, incidentally). Even though you have light coming from everywhere, some parts of your model will block light from reaching other parts. There will be minor shadows and highlights where parts come together or get in the way of each other.

<pagebreak></pagebreak>

In order to understand this better, go ahead and test it:



This image should give you a good idea what ambient occlusion is all about. By combining it with your **Texture** version (or even with your **Full Render** version), you can quickly add some detail and complexity to your final image texture.

Adding detail in an Image Editor

I recommend saving your various baked renders as separate images, and then combining them in a 2D Image Editor that supports layers (such as GIMP, Photoshop, and so on.).

Once you've combined them together in a way that you like, you can also go through and manually paint highlights, shadows, and decals onto your texture map.

<pagebreak></pagebreak>

For instance, I've quickly painted a basic tread pattern onto the tire:



This doesn't look particularly realistic (as a tread pattern), but it's just an example. You can easily find real tread patterns online or paint your own.

Another thing you can do is take photographs of real car parts and overlay them onto the texture image. For example, you can take a photo of a real fog light and use that:



Note

With other Blender projects, we wouldn't want to do this. The 2D image of a light will never be as good as a properly modeled lens with transparent materials. For a game model, however, we can't afford the geometry we'd need to model it; most game engines won't give you that level of realism for glass surfaces anyway.

<pagebreak></pagebreak>

You can also take this opportunity to add any decals or logos that you'd like:

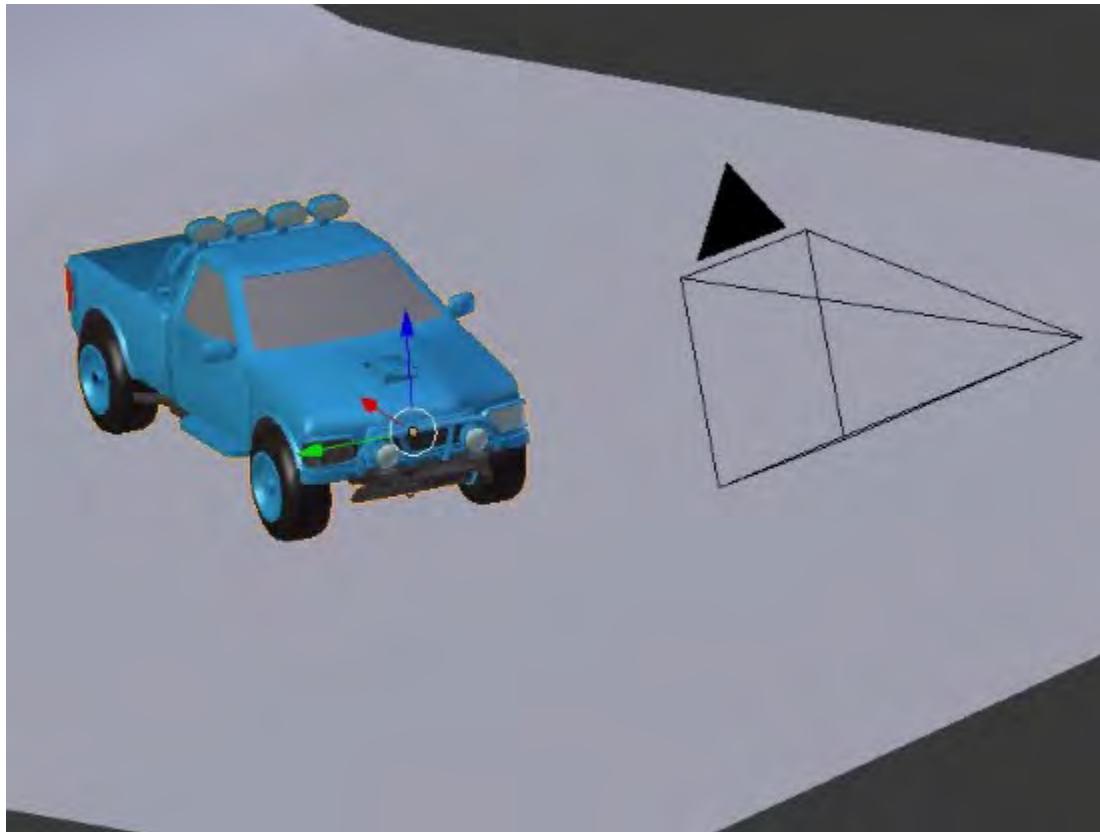


With a little time and practice, you can create some pretty nice texture maps with these techniques.

Checking the Texture Map

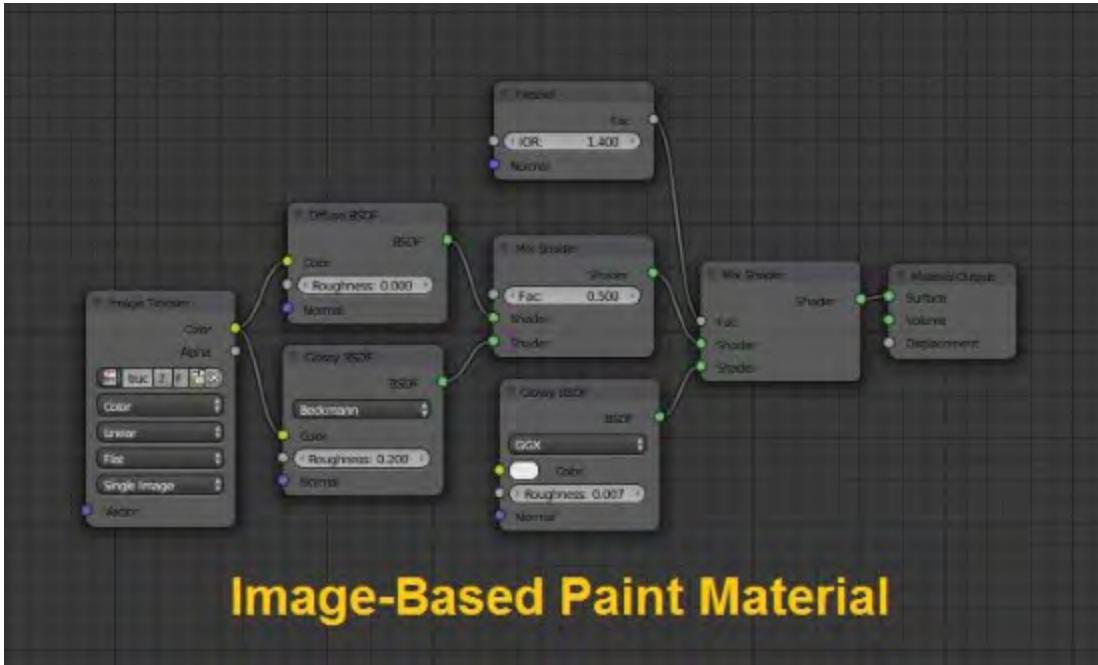
After we're done, we can test out the material/textures in Blender. We'll use the texture map to create some Cycles materials and ensure that everything lines up properly.

You can start by just adding a basic background, lighting setup, and camera:



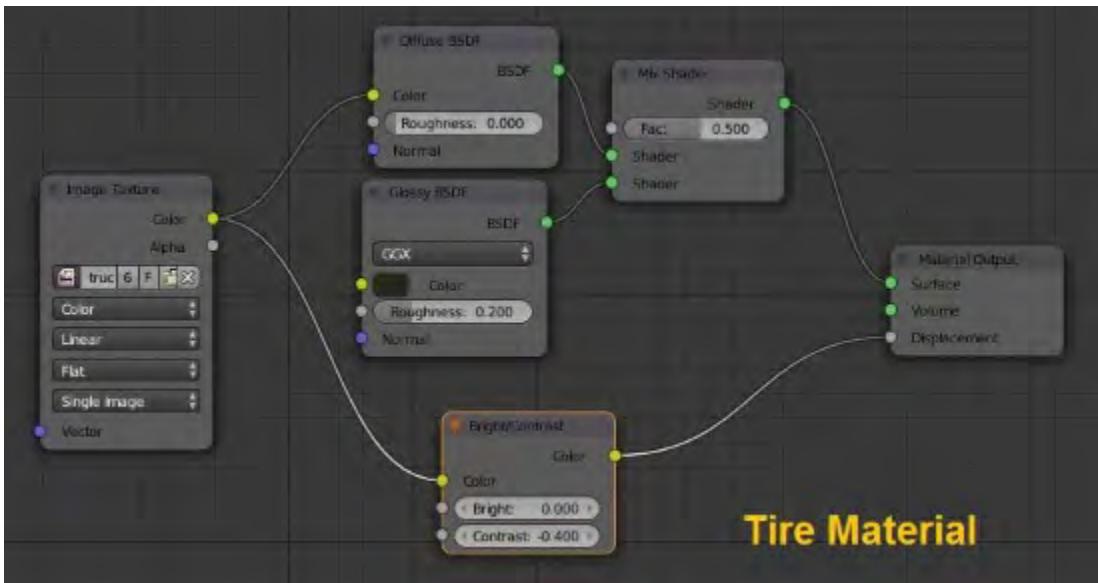
<pagebreak></pagebreak>

Then, we'll create our materials. For instance, here is a quick car paint material using the texture image:

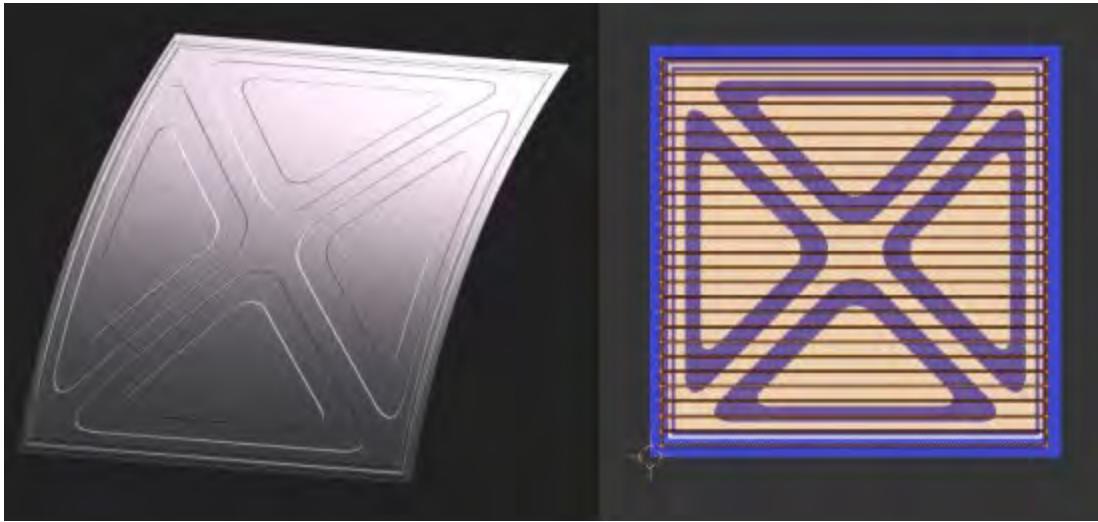


<pagebreak></pagebreak>

For the tire, we can also use that texture image to map to the **Displacement**:



For increased realism, you can also create a separate image map for this purpose. Rather than showing colors/textures, it will only show the Normals or Displacement that we want on various parts of the truck. Here's a quick example of a normal map in use:



<pagebreak></pagebreak>

And here's a quick version of the truck that uses normal maps to add damage/detail to the body (Render Credit: James Clayton):



Feel free to experiment with separate Displacement/normal maps. However, keep in mind that every image you have to load will increase render time and place additional stress on the CPU/GPU. Only use a separate map when you're sure that your game or other 3D program can handle it.

<pagebreak></pagebreak>

With basic Cycles materials, here's what the truck may look like:



Again, this doesn't really matter; the point of the model is to be used in another application, so don't spend a lot of time fine-tuning the materials in Cycles. All we're really looking for is any alignment or texture issues and generally making sure the model looks okay.

<pagebreak></pagebreak>

When you're satisfied, make sure that you delete unneeded objects from your scene (lights, background, cameras, and so on). Then you can use the **Export** menu to **Output** the model into a format of your choice:



Summary

In this chapter, we created basic materials for the truck. We then used the Blender Internal engine to create basic procedural textures. After unwrapping the model, we baked our textures out and used that image to create and test materials in Cycles. As with the previous project, we've just covered the basics of this topic. However, it should provide a good baseline for you to take it further.

Appendix 1. All About Packt



Thank you for buying Blender 3D Incredible Machines

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 that focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website at <http://www.packtpub.com>.

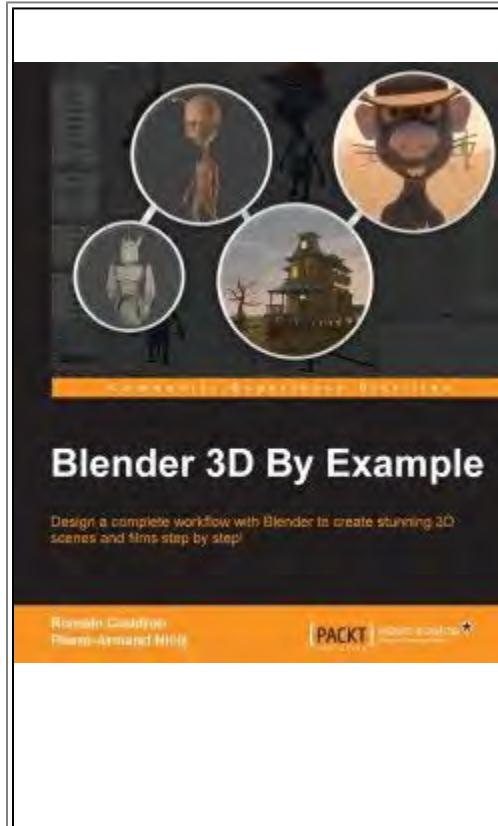
About Packt Open Source

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 licenses, 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, then please 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.

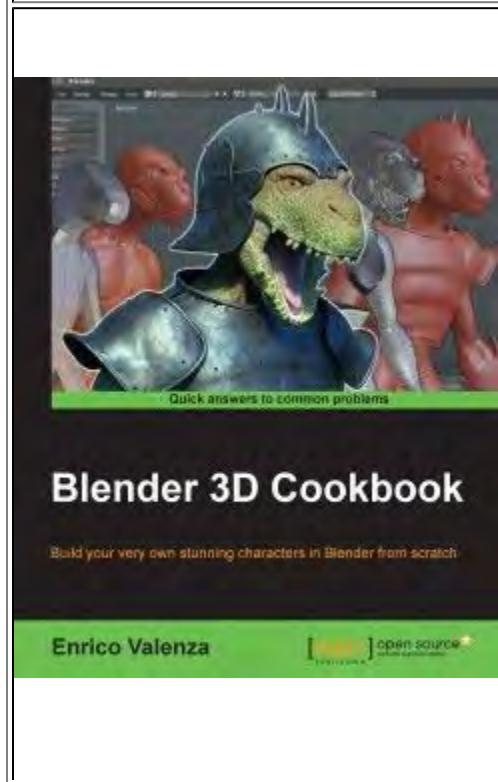


Blender 3D By Example

ISBN: 978-1-78528-507-3 Paperback: 334 pages

Design a complete workflow with Blender to create stunning 3D scenes and films step-by-step!

- Make use of the powerful tools available in Blender to produce professional-quality 3D characters and environments
- Discover advanced techniques by adding fur to a character, creating a grass field, and fine-tuning a shot with post-processing effects to enhance your creations



Blender 3D Cookbook

ISBN: 978-1-78398-488-6 Paperback: 608 pages

Build your very own stunning characters in Blender from scratch

- Establish the basic shape of a character with the help of templates, and complete it by using different Blender tools
- Gain an understanding of how to create and assign materials automatically, working in both the Blender Internal engine as well as in Cycles

- Familiarize yourself with the processes involved in rigging, skinning, and finally animating the basic walk-cycle of the character



Blender Cycles: Materials and Textures Cookbook Third Edition

ISBN: 978-1-78439-993-1 Paperback: 400 pages

Over 40 practical recipes to create stunning materials and textures using the Cycles rendering engine with Blender

- Create realistic material shaders by understanding the fundamentals of material creation in Cycles
- Quickly make impressive projects production-ready using the Blender rendering engine
- Discover step-by-step material recipes with complete diagrams of nodes



Blender 3D 2.49 Incredible Machines

ISBN: 978-1-84719-746-7 Paperback: 316 pages

Modeling, rendering, and animating realistic machines with Blender 3D

- Walk through the complete process of building amazing machines
- Model and create mechanical models and vehicles with detailed designs

- | | |
|--|---|
| | <ul style="list-style-type: none">• Add advanced global illumination options to the renders created in Blender 3D using YafaRay and LuxRender |
|--|---|

Please check **www.PacktPub.com** for information on our titles

Part 3. Module 3

Blender 3D By Example

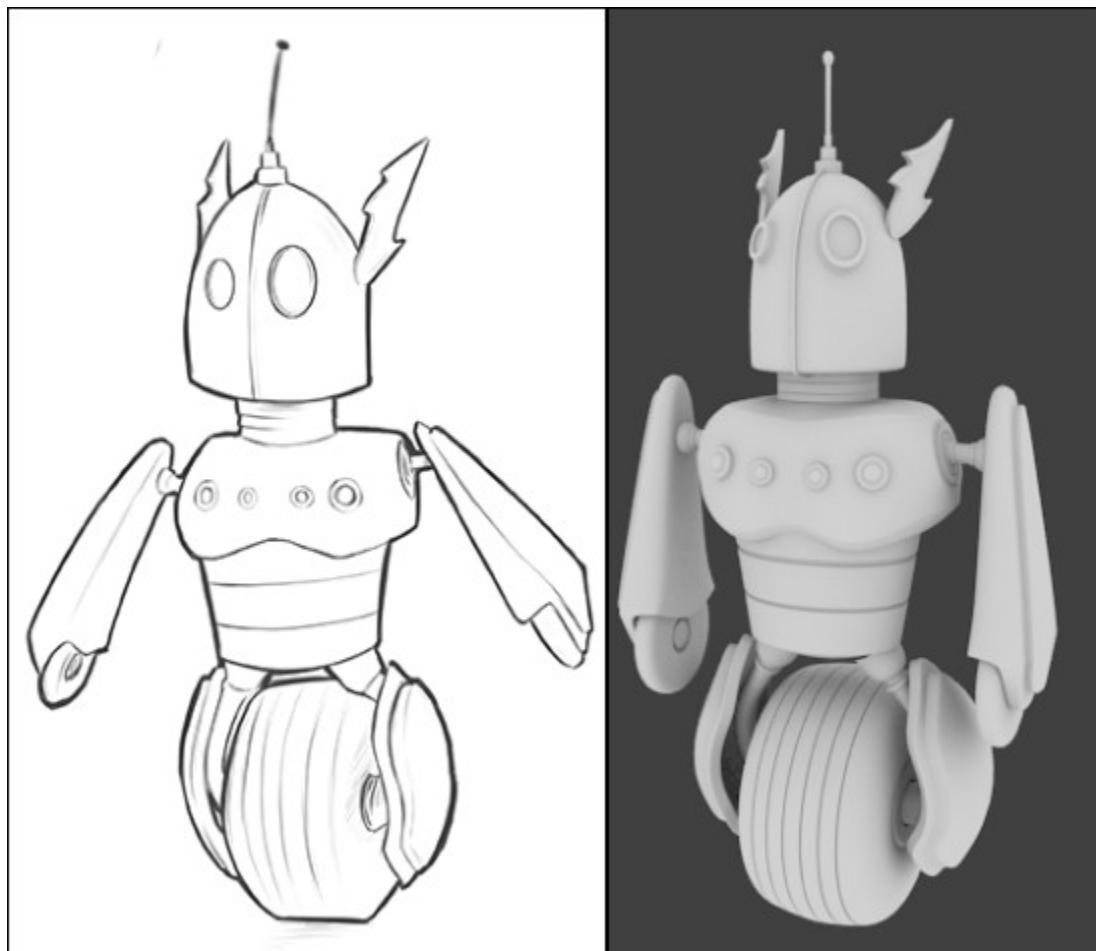
Design a complete workflow with Blender to create stunning 3D scenes and films step-by-step!

Chapter 1. Robot Toy – Modeling of an Object

In this chapter, we will start our first project in order to discover the fundamental modeling tools of Blender. We will create a little robot that is inspired by vintage toys with a drawing image reference. You will learn polygonal modeling workflow, which will be useful for your future 3D productions. The head will be created with a simple cylindrical primitive that we will modify to give it the right shape. Then, in the same way, starting from a primitive, we will model the rest of the body, always with a good topology in mind. Indeed, we are going to maximize the number of quads (polygons with four faces) and organize them so that they best fit the shape of each part. In the end, we will do a quick render with the Blender internal render engine. Without further ado, let's enter the marvelous world of 3D modeling! In this chapter, we will cover the following topics:

- Adding and editing objects
- Using the basic modeling tools
- Understanding the basic modifiers (such as mirror and subsurface)
- Modeling with a proper topology
- Creating a quick preview with Blender Internal

In the following screenshot, on the right, you can see the 3D robot modeled using a sketch, shown on the left as a reference, with Krita, which is another open source tool for 2D art:



Let's start the modeling of our robot toy

We will now start the modeling of the robot toy by adding the first object to the scene. The robot will be modeled from a simple cylinder.

Preparing the workflow by adding an image reference

In order to start the modeling of the robot, let's have a look at the following procedure:

1. We will add the robot image reference in a new **UV/Image Editor**.
2. After dividing the view and selecting the right editor (by clicking on the RMB on the edge of an editor and selecting **Split Area**), go to the **UV/Image Editor** header and select **Open Image** to choose the corresponding reference in the file browser.
3. To pan or zoom in this editor, use the same shortcuts as the 3D view. This reference will serve as a guide during the modeling process. Refer to this in order to get the main shape right, but don't rely on its details.

Adding the head primitive

When you start modeling an object, you need to start with a basic 3D shape that is close to the shape you want to model. In our case, we will use a cylindrical primitive to start modeling the head. To do this, follow these steps:

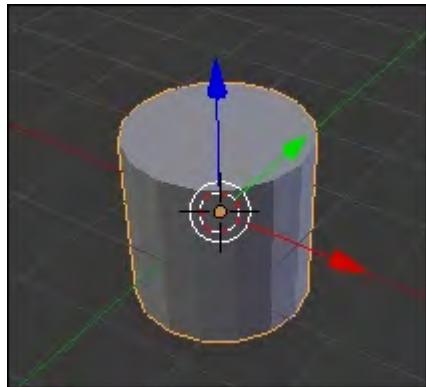
1. First we will need to remove the 3D cube that is placed by default in any Blender starting file. The cube is selected if it has an orange outline. If this isn't the case, you can right-click on it. This is the main selection method in Blender. If you want to select or unselect all the objects present in the 3D view, you can press the **A (All)** key.
2. You can now remove the selected cube by pressing the **X** key or the **Delete** key. It's now time to add the cylindrical primitive.
3. All the primitives are going to spawn at the 3D Cursor location. We will ensure that the cursor is at the center of the scene by pressing **Shift + C**.
4. We can now use the **Shift + A** shortcut, and select **Mesh | Cylinder** to create the primitive at the center of the scene.
5. Our new object has too many details, so we will decrease the number of vertices in the left 3D view panel. If you can't see this panel, press the **T** key. At the bottom of this, you can see the preferences of the currently active tool (the mesh creation, in our case), and you can change the number of vertices of our cylinder to **16**.
6. We will now set the 3D view focus on the newly created object by pressing the dot numpad key or by selecting **View | View Selected** in the 3D view header.

Note

About naming shortcuts

Most of the shortcuts correspond to the first letter of the tool's name. For instance, the Grab tool can be activated by the **G** key and the Scale tool can be selected by the **S** key.

If you want to explore all Blender's shortcuts, visit <http://www.shortcutsheaven.com/>.



The cylinder located at the cursor position (center of the world) that we will use as a base for the head of the robot.

The Edit Mode versus the Object Mode

Currently, we cannot access the components (vertices, edges, and faces) of our cylinder because we are in the **Object Mode**. This mode allows you to do basic things on objects such as moving, rotating, or scaling them. Let's perform the following set of steps:

1. If you want to edit an object, you need to use the **Edit Mode**. To switch between these modes, press the *Tab* key or go to the **Modes** drop-down menu in the 3D view header while any object is selected. In the **Edit Mode**, you can choose the type of components to select by pressing *Ctrl + Tab* or by selecting the component type in the 3D view header.
2. Let's go into the **Face Mode** and select the top face of the cylinder by right-clicking on it. As you can see, in Blender, faces (or polygons) are represented by a little square in the middle.
3. Now you can go into the orthographic front view (the *3* numpad key and *5* numpad key for perspective/orthographic views respectively), and use the *z* axis of the Gizmo tool to move the selected face a little bit down.

Note

In Blender, we don't encourage you to use gizmos as there is a much faster method to move, rotate, or scale your selection. To move a selection, press the *G* (Grab) key and, if you want to constrain your move to a certain axis, press the corresponding *X*, *Y*, or *Z* keys. You can even hide the Gizmo tool by pressing *Ctrl + Space*.



The top face moved down in the **Edit Mode** with the z axis of the Gizmo tool or by pressing the G + Z shortcut.

Using the basic modeling tools

In the following, you will learn the powerful usage of the main modeling tools of Blender, such as the Extrusion, Bevel, or Loop Cut tool, while creating your little robot toy.

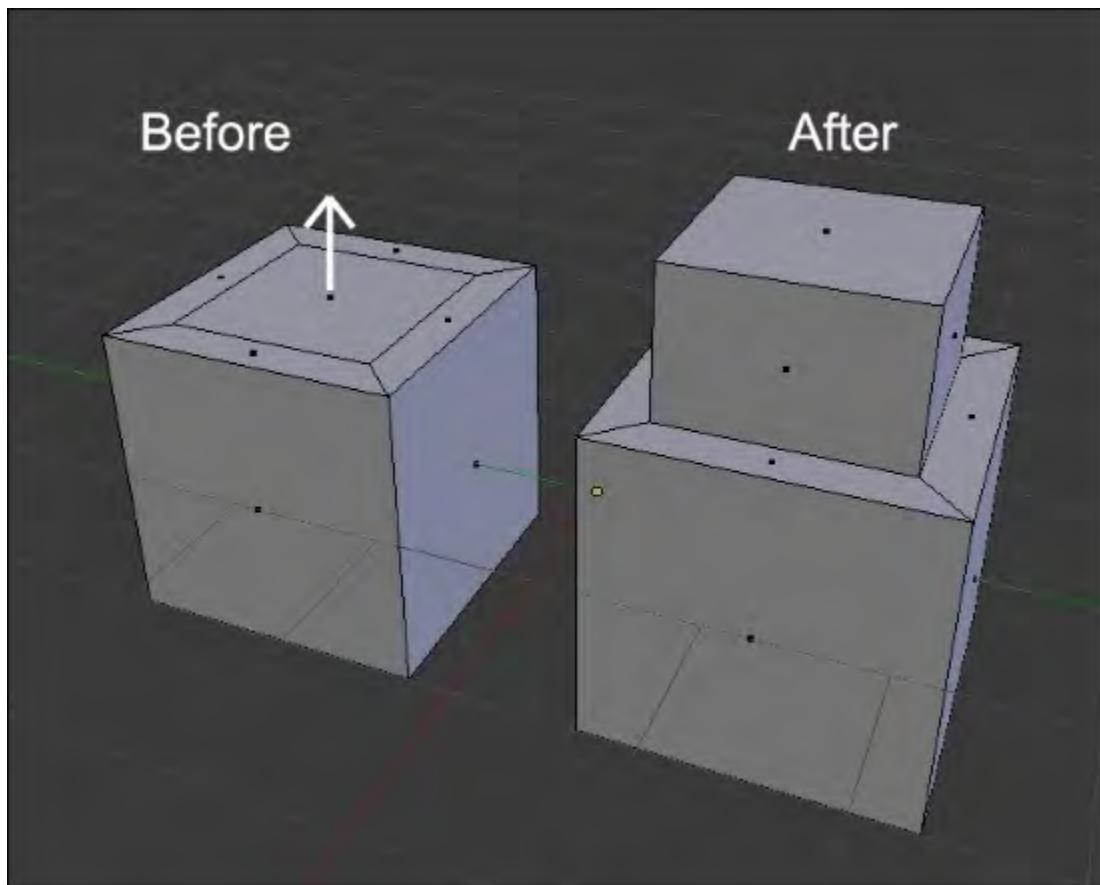
Modeling the head

We will now use the basic modeling tools in order to form the shape of the head. As you may have understood, we are going to add new geometry gradually to approximate the shape in 3D. One of the useful tools is called **Extrusion**.

Note

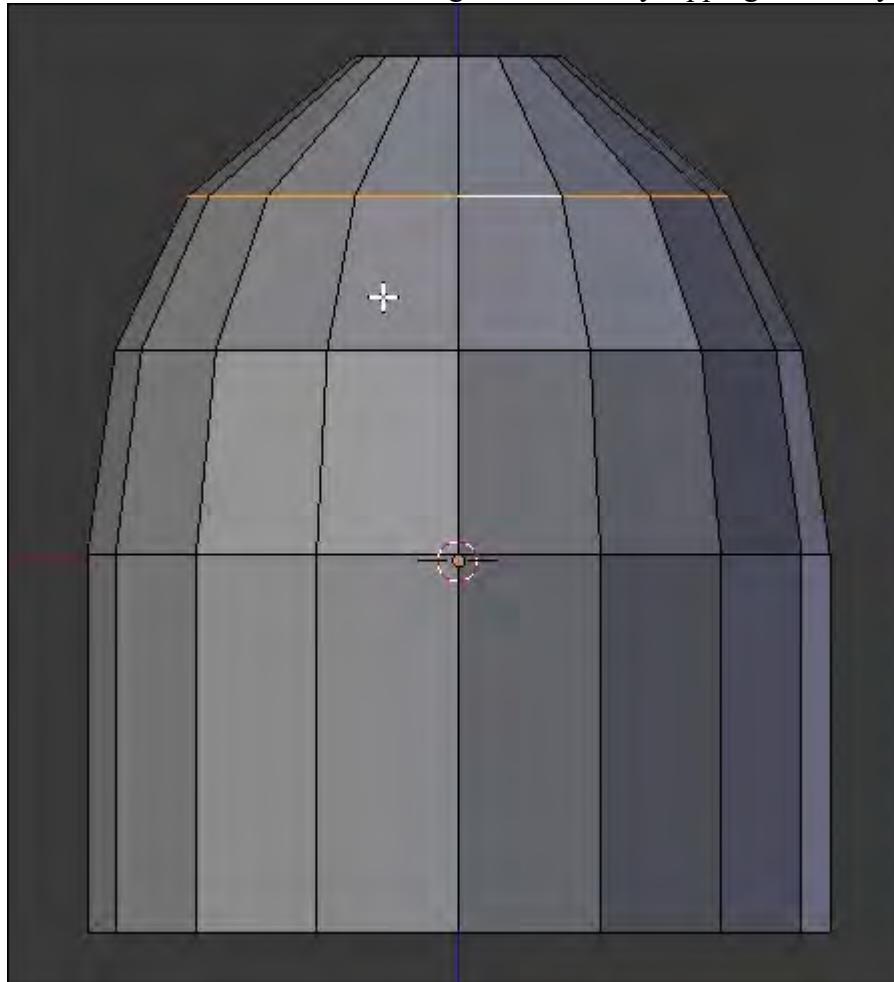
What is an extrusion?

This is the process of creating new geometry by extending (and optionally transforming) selected components.



1. While the top face of the cylinder is selected in the **Edit Mode**, we will press the *E* key in order to create a new geometry from that face. Then, we will need to position and validate the extrusion.

2. We now have two choices in order to confirm the extrusion. If you want, you can move the extruded geometry, and after this, press LMB in order to validate its position. The other choice is to press RMB, in order to place the extruded geometry at the same position as that of the selected component(s). In our case, we will place the extrusion just over the selected face.
3. We can now scale our extrusion by pressing the *S* key, and repeat the process of extruding the top face and scaling it three times in order to have a bell shape. We can always go back by pressing *Ctrl + Z* and redo these steps.
4. We can also go into the **Edge** component mode (*Ctrl + Tab*), and select the edge loops that came from the different extrusions by placing the mouse pointer over them and pressing *Alt* and *RMB*.
5. After this, we can move them along the volume by tapping the *G* key twice.

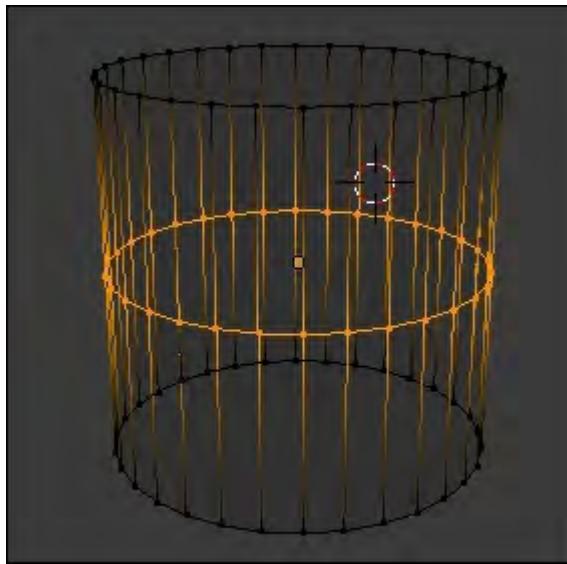


Shaping the head with extrusions.

Note

What is an Edge Loop?

An Edge Loop is a set of edges that are connected together and form a loop. You can also get face loops to follow the same principle but with faces. They are essential to construct the shape of an object.



Modeling the antenna

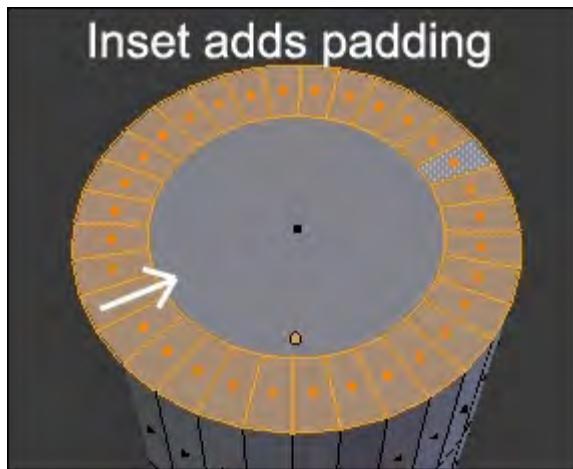
In order to create the antenna, we will start from the head and detach it later. Follow these steps to create the antenna:

1. In the head **Edit Mode**, we will select the top face and extrude it a little bit to make the base of the antenna.
2. Then we will make an inset from the top face of the base by pressing the *I* key.

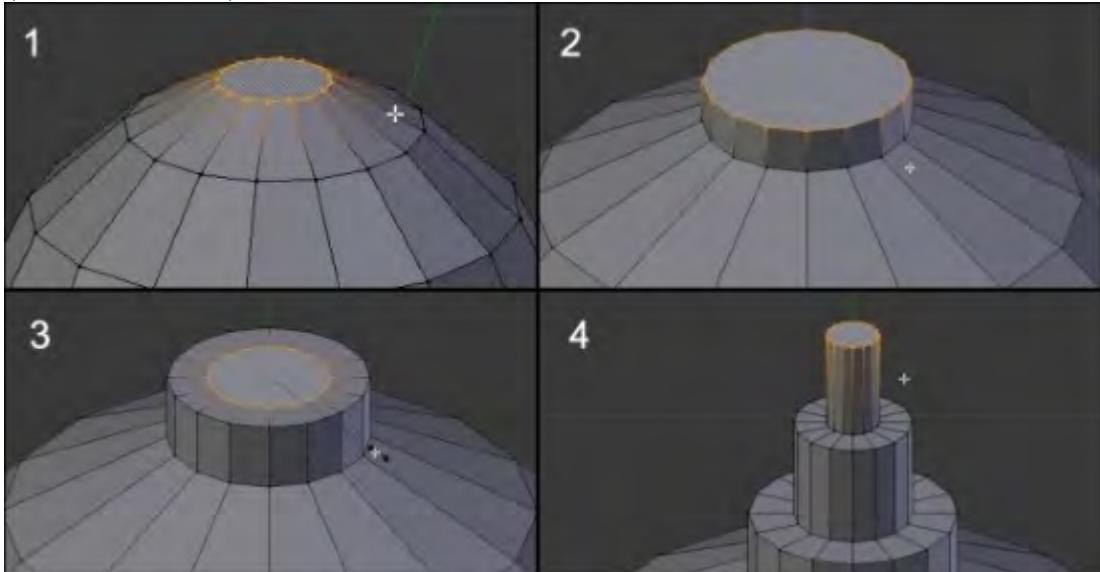
Note

What is an inset?

An inset allows you to add some padding on a face:

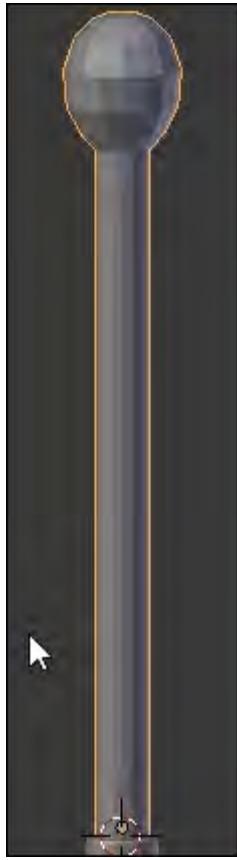


3. With the inner face of the inset selected, we are going to repeat the process one more time (extrusion + inset).



The different steps to model the base of the antenna. A succession of insets and extrusions.

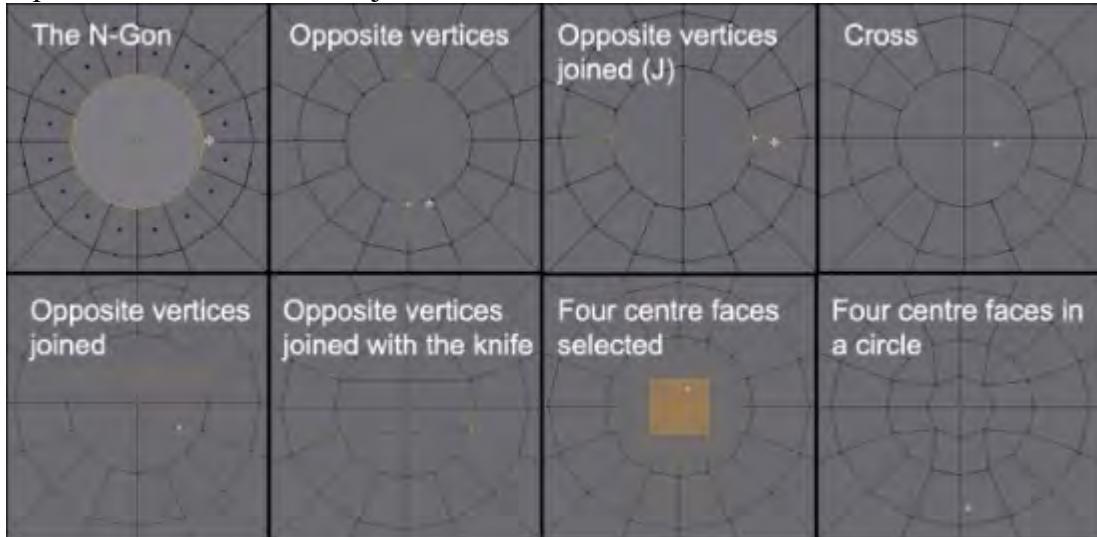
4. After this, we will add the stem of the antenna by moving the cursor to the top of the antenna's base, selecting the top face, pressing **Shift + S**, and selecting **Cursor to Selected**.
5. Now we can add a cylinder in the **Object Mode** that will pop up on the cursor. This cylinder will represent the stem, so scale it accordingly and end it with some extrusions that you will form in a sphere shape by scaling them and following the same process as that of head modeling.
6. You can also select the top part of the stem and use the smooth option (by pressing the **W** key and selecting **Smooth**) to relax the geometry.



The stem with the different extrusions that we have shaped like a sphere with the Smooth tool

7. We now have an **N-Gon** at the top of the stem. An N-Gon is a polygon that has more than four edges. It is considered a bad practice to have these kinds of polygons in a 3D object. We are going to solve this by going into the top view (the 7 numpad key) and by doing a small inset on the object in order to maintain the border.
8. After this, we will connect some vertices together in order to have only quads (polygons that have four sides).
9. Then, we select the two opposite vertical vertices by right-clicking on the first one and pressing *Shift* and right-clicking on the second one. Pressing *Shift* and invoking any selection method allows you to add new items to your current selection.
10. After this, we join them to a new edge with the *J* key (to connect the vertex path tool) to separate the N-Gon into two equal parts.
11. Now we ought to select the two opposite horizontal vertices and join them to form a cross. If you look closely, we haven't resolved the N-Gon problem yet, because we have four more of them.
12. As we can't leave them in the mesh, we are going to repeat the process by joining the other facing vertices in order to have only quads. If you want, you can also use the Knife tool in order to cut in the geometry by pressing *K*. With the knife we will have to click on the vertices that we want to connect together and when we finish, we can press the *Return* key in order to validate.

13. At this point, we can use the **LoopTool** add-on that we installed in the first chapter. We can select the four middle faces (in **Face Mode**) and use the **LoopTool** circle option (press **W** then select **LoopTool | Circle**). This allows us to form a circle with the selected components.
14. It's time to detach the antenna. In order to do this, we select the loop at the base of the antenna (press **RMB** and **Alt**) and press **V** to rip the loop. Blender will give us the choice of moving the ripped part, but we won't. So we cancel the move by clicking the **RMB**.
15. Now, we will detach the geometry of the antenna to form a new object. First we deselect all the components (**A**), then we move our mouse pointer over the antenna and press **L** to select the linked geometry.
16. After pressing the **P** key and choosing **Selection** in the pop-up menu, the selected part will be separated to form another object.



N-Gon correction with the Join tool and the Knife

17. We will have to clear three N-Gons: one at the bottom and at the top of the head, and one at the top of the base of the antenna. We have decided to resolve them with the previously explained method.

An introduction to the Subdivision Surface modifier

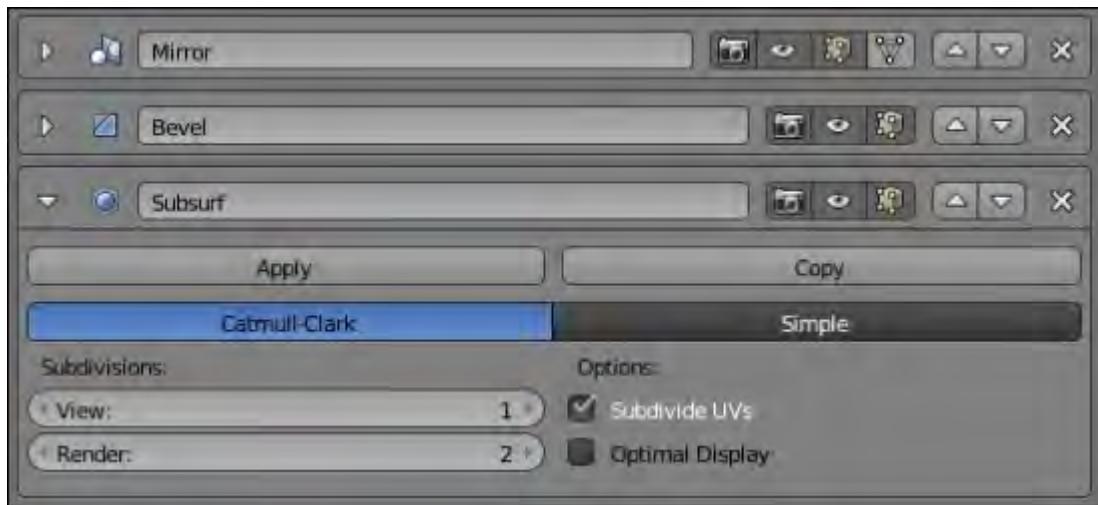
We will now smooth the geometry of the robot head and the antenna using the following steps:

1. First we go to the left 3D view panel (**T**), and with both objects selected in the **Object Mode**, we click the **Smooth** button under **Shading**. This will create a blend between the faces but not round our objects. In order to round our geometry, we will need to use a modifier called **Subdivision Surface**.
2. Let's go into the Properties editor and select the adjustable wrench. Then, we choose a **Subdivision Surface** modifier in the **Add Modifier** drop-down menu. All we need to do now is repeat the process with each object.

Note

What is a modifier?

A modifier is a tool that applies to the entire object. You can push new modifiers on the modifier stack of the object where the top modifier will take effect before the bottom one. You can also reorganize their order using the up and down arrows. You may hide a modifier using the Eye button. If you want to collapse a modifier, use the left-hand side horizontal arrow. You can also apply the behavior of the modifier with the **Apply** button. Always save your work before doing this.



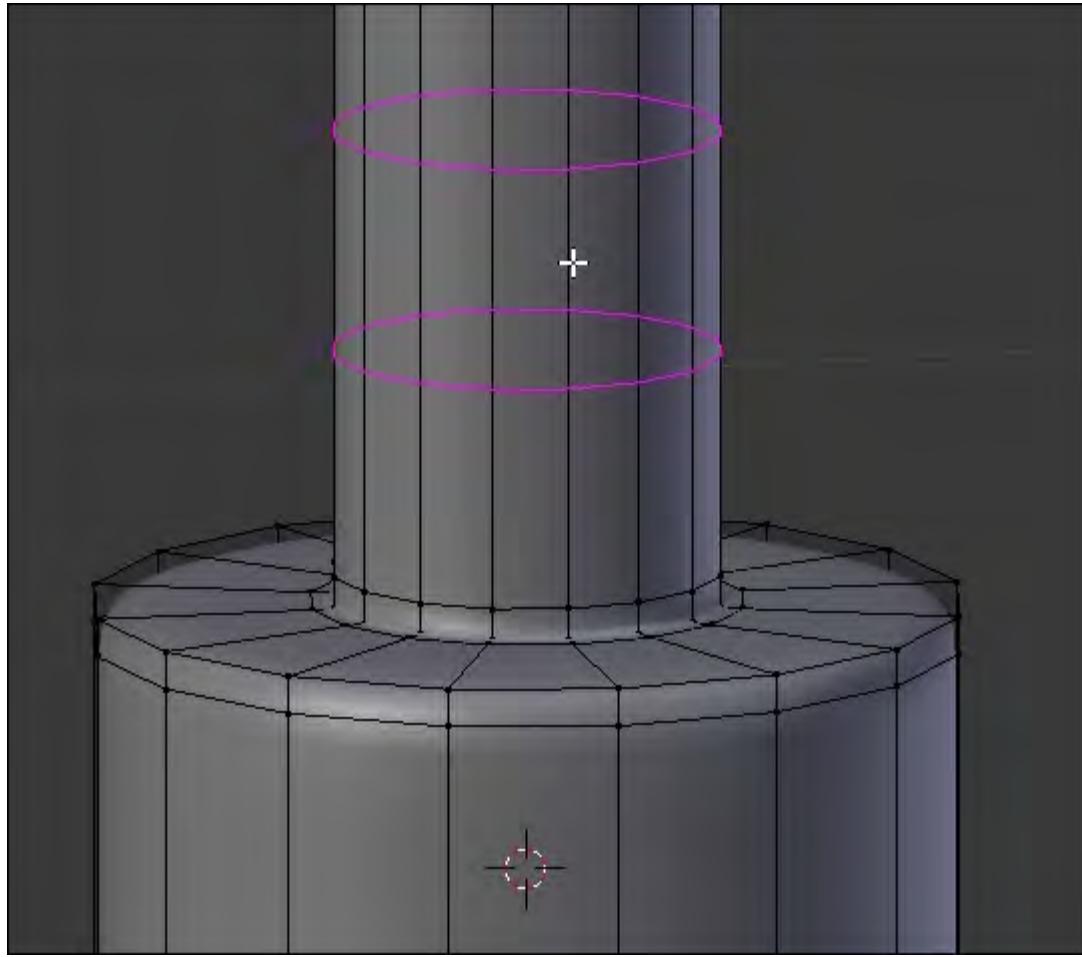
The stack of three object modifiers. The **Subdivision Surface** applies over the **Mirror** and the **Bevel** modifier.

3. As you may have seen, the subdivision divides all the polygons by four and tries to do an interpolation by smoothing them. If you want more divisions, you can increase the View slider under Subdivisions.
4. The shape looks better but needs to be sharp at some points. In order to do this, we will maintain a border by adding edge loops with *Ctrl + R*. The LoopCut tool is very useful; it allows us to add edge loops where we want and as many as we want.

Note

About the LoopCut tool

To add an edge loop, use the *Ctrl + R* shortcut and move your mouse cursor perpendicular to where you want to add a new edge loop. You will see a preview of the new cuts. You can add multiple loops at the same time by scrolling your mouse wheel or by pressing the + or – keys. After you have validated the cuts, you will need to position them and validate their location by left-clicking or right-clicking. The later will center the cuts.

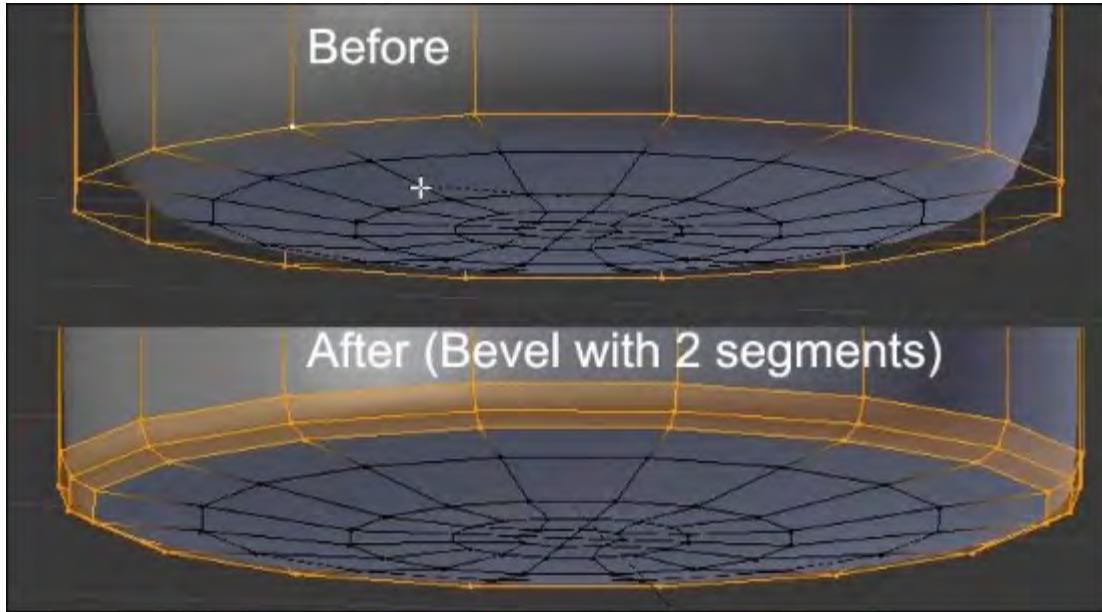


5. We can also sharpen the edges by selecting them and using the Bevel tool. Remember that the nearer the edge loops are, the sharper the result will be.

Note

About the Bevel tool

The Bevel tool allows you to split one edge into multiple edges. When you activate it with *Ctrl + B*, you can choose the number of splits that you want by scrolling your mouse wheel or by pressing the + or – keys. You can also decrease the speed of the tool by pressing the *Shift* key. As always, you can validate your placement by left-clicking or cancel it by right-clicking.



Improving the head shape

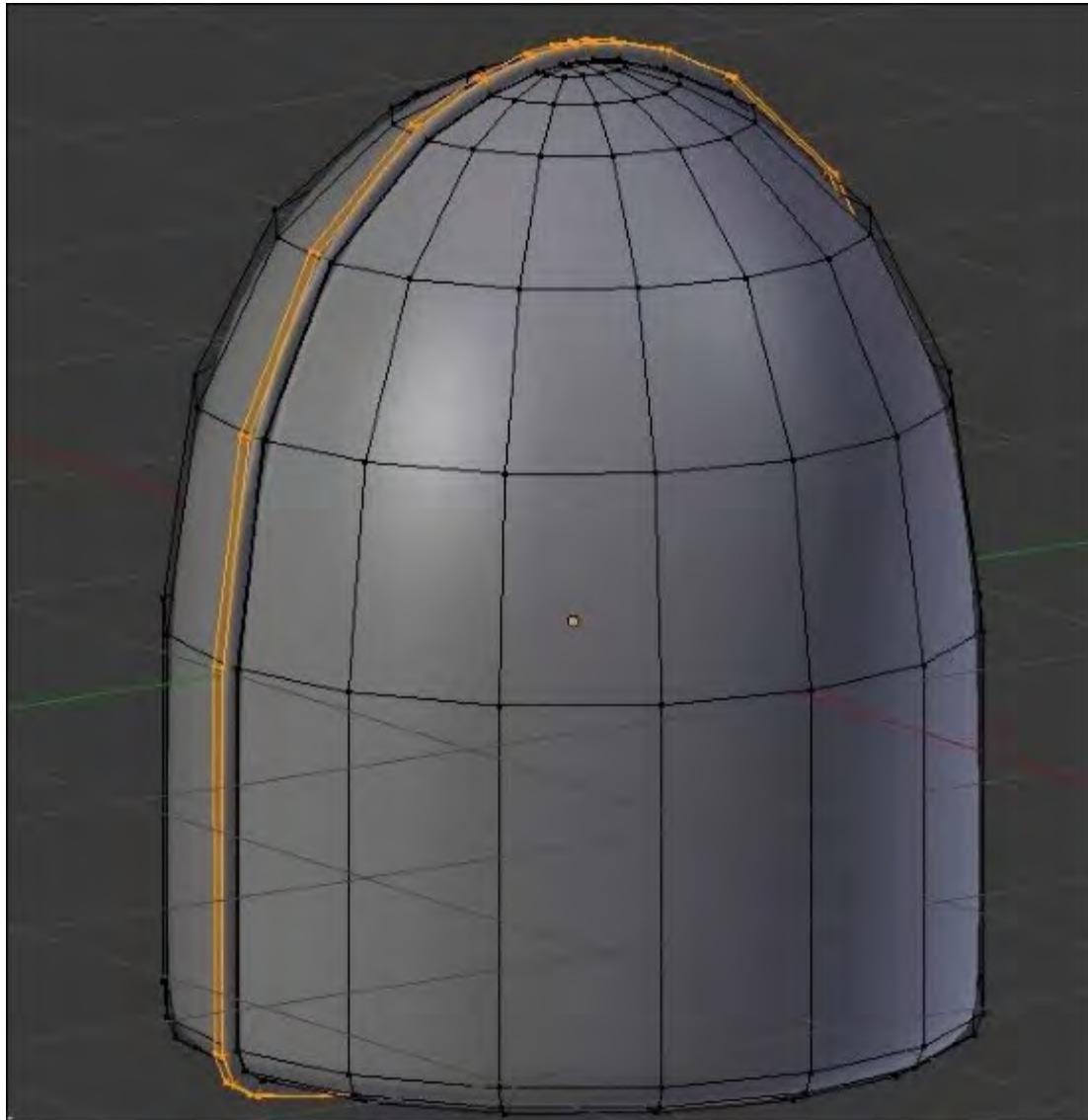
Let's select the head and go into the **Edit Mode**.

1. From the front view, we will now select the central edge loop. One way to do this is to use the wireframe mode by pressing the **Z** key. This mode allows us to see through the mesh and the selected components that are behind it.
2. We can now use the Box Selection tool with the **B** key in order to draw a rectangle area around the vertices that we want to select. If you want, you can hide the antenna and the stem by selecting them and pressing **H** (Hide). The Bevel tool will help you to create a thin base in the middle of the head. This bevel will be maintained with two new cuts. To do this, we can do an extrusion without moving it (cancel the move with RMB).
3. We can now scale the newly extruded faces on the **x** axis using the **S + X** shortcut. The thickness is added by selecting the inner face loop and extruding it at the same place.
4. In order to push the extrusion according to the normals, we use the **Alt + S** shortcut.
5. We will also have to maintain the shape of the head by adding multiple edge loops (with **Ctrl + R**, for instance).

Note

Save your work!

After all the work you've done, it's very important to save it! To write your blend file to your hard disk, go to the **File** menu and press the **Save** option or use the **Ctrl + S** shortcut. You can now choose which directory you want to place it in. A nice trick is to press the **+** or **-** key to add or remove one unit from the name of your file.



The head shape without the antenna.

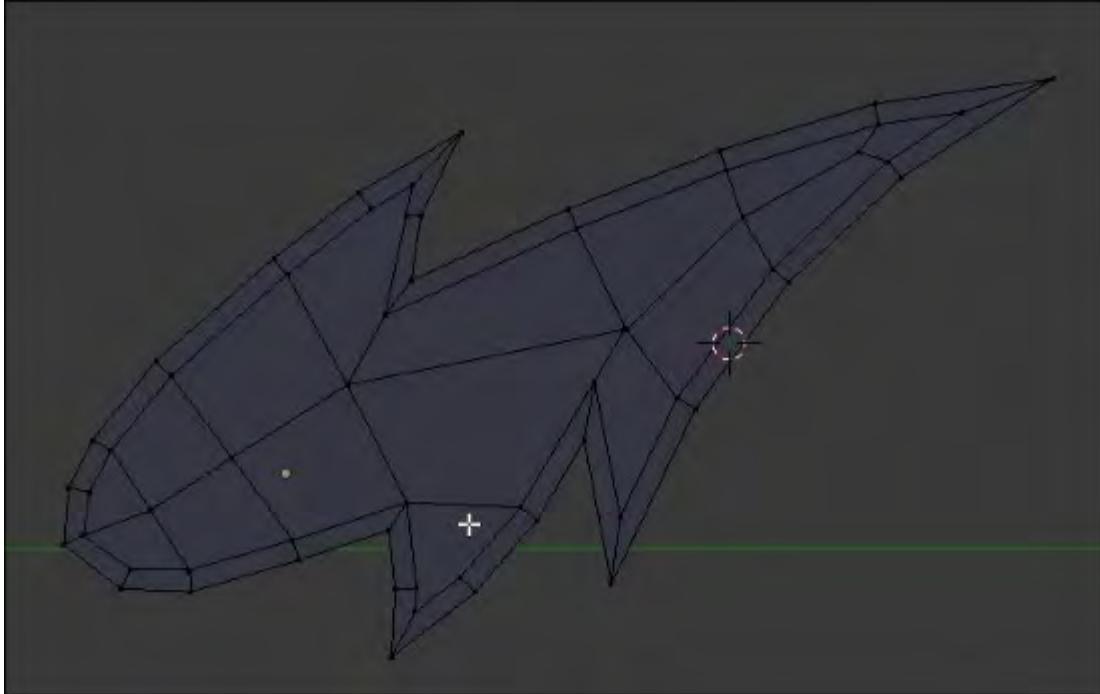
6. You can unhide the antenna and the stem in the **Object Mode** by pressing **Shift + H**.

Modeling the thunderbolts

It's now time to start modeling the thunderbolts; let's have a look at the following steps:

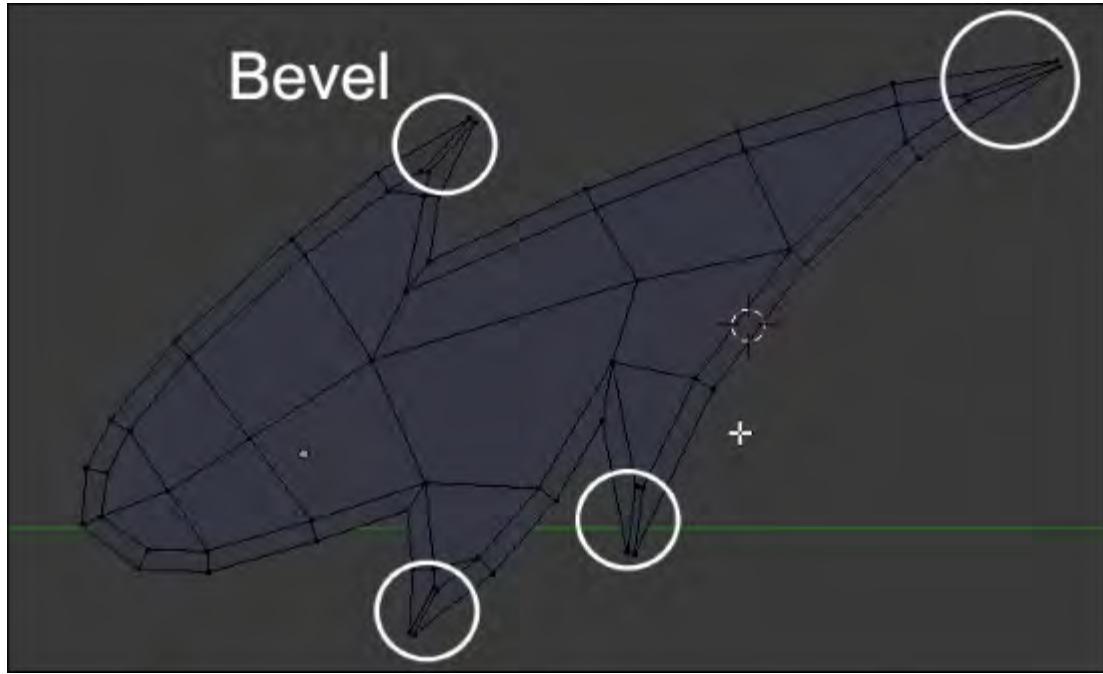
1. We will start by going into the Side Orthographic view (the 3 numpad key) and by placing the cursor next to the head with a simple left-click.
2. Then we will add a plane and in the **Edit Mode** we will remove all the vertices with the **X** key.
3. In the **Edit Mode**, we are going to create a chain of vertices that matches the thunderbolt shape of the image reference.
4. Pressing **Ctrl** and **LMB**, we will add new vertices and create the silhouette of the thunderbolt.

5. In order to close the shape, we select the first and last vertices and press the F key to fill them with an edge.
6. If you want to add more details to the shape, select two connected vertices and with the LoopCut tool (*Ctrl + R*), place a new vertex in the middle of both the connected vertices.
7. We can then select all the vertices (A) and fill the shape with an N-Gon (F) that we are going to resolve later.
8. We can now add an inset (I) in order to keep an outline.
9. After we've done this, we will have to clean the mesh by replacing the N-Gon with quads using the join tool (J) or the knife tool (K). If you have one triangle or N-Gon, it's not a problem for now as it could be solved later.



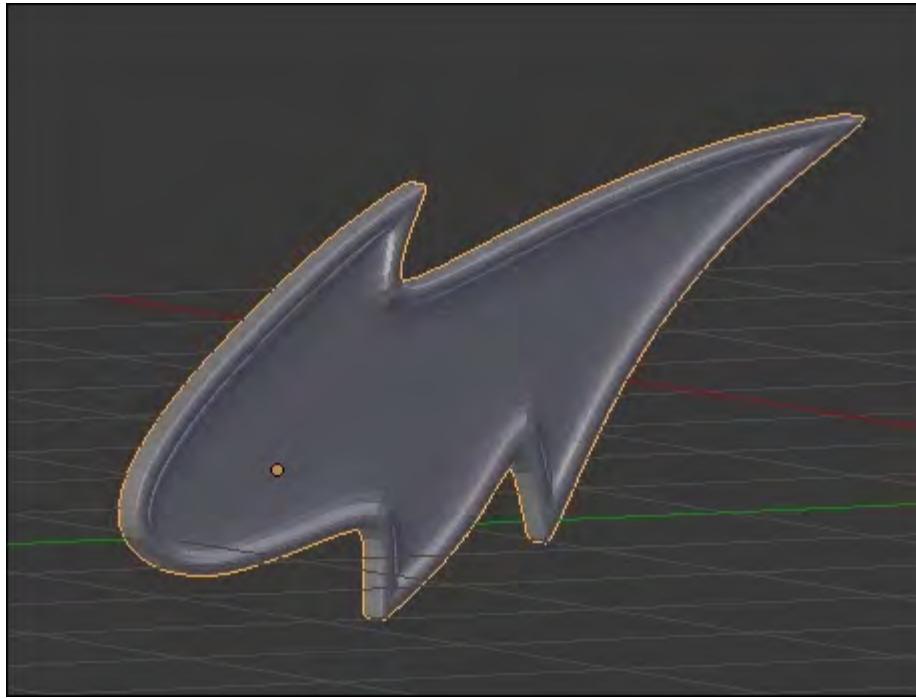
The thunderbolt shape.

10. We can now add a **Subdivision Surface** modifier in the **Object Mode**.
11. We will have to sharpen the spikes using tight bevels.
12. Of course, we will have to clean the mesh by removing the N-Gons.



Maintaining the spikes with bevels.

13. It's now time to extrude the whole thunderbolt by selecting all the faces (*A*). You may get a lighting error with black faces. It means that the normals are pointing inward and can't catch the light. You can verify this by opening the right panel of the viewport (*N*) and, under Normals, you can check the face icon. If the normals are not pointing outward, then you will need to recalculate their direction by selecting all the components and pressing the *Ctrl + N* shortcut.
14. We can now select the inner faces on the outside of the thunderbolt with either *Shift + RMB* or using the *C* key, which allows you to paint and select the components that you want according to the current view. With these faces selected we can create a small inner extrusion, and maintain the shape with the LoopCut tool (*Ctrl + R*).



The finished thunderbolt with a view 2 Subdivision Surface.

15. In order to mirror the thunderbolt on the other side of the head, we will use a **Mirror** modifier with the head as the center of a pivot. Place and rotate the thunderbolts according to the image reference.

Note

The Mirror modifier

This is an easy way to make a symmetry from your 3D model. The basic symmetry is based on the positioning of the pivot point and the x axis. All of these are configurable by changing the axis. It is strongly advised you use the **Clipping** option if you want to weld the components that are on the symmetry axis of your geometry. By doing this, you will avoid holes. With the **Mirror Object** option, you can choose to base the symmetry axis on the pivot point of another object in your scene.

16. The last thing we may want to do at this stage is to correctly name our objects in the outline editor that is situated in the top-right corner of the interface by default.

Note

The outliner

The outliner displays a list of all the entities that make up the current scene. When you select an object in the 3D view, it will be highlighted in the outliner and conversely. You can rename any item in the list by double-clicking on its name. The outliner also gives you control of the

visibility of any object with the Eye icon button. The mouse cursor button can be toggled on or off to allow the selection of the corresponding object in the viewport.



Modeling the eyes

It's time to finish the head of our robot by adding a pair of eyes on it. This is done with the following steps:

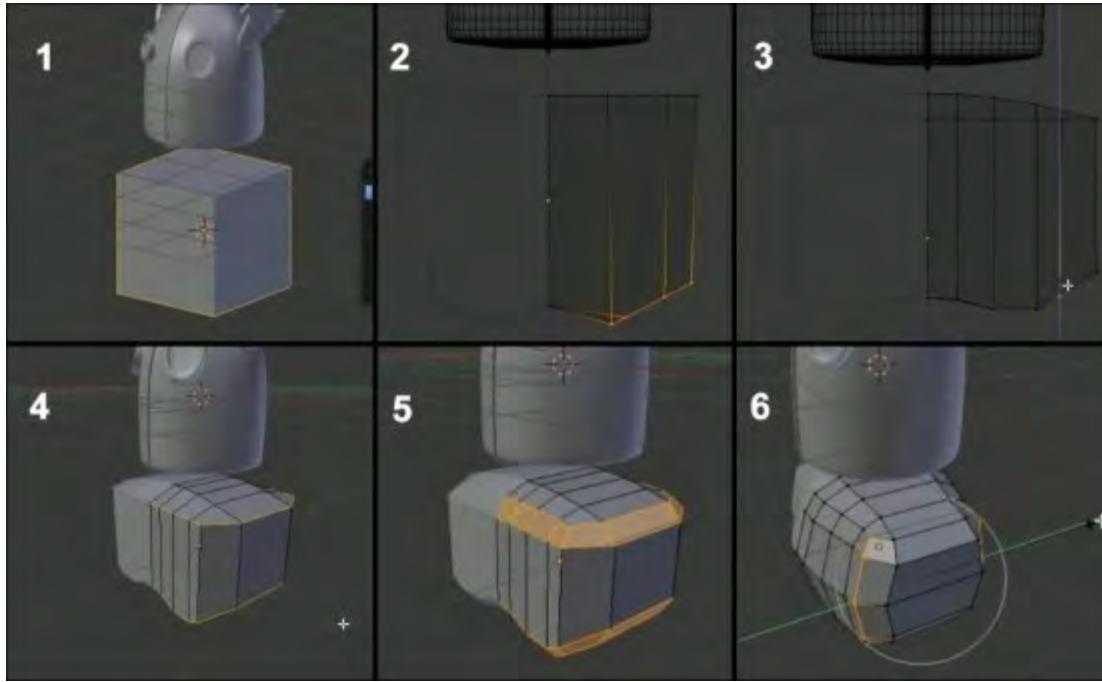
1. In the **Edit Mode** of the head, we select the edge that is roughly positioned at the eye location.
2. We use a bevel to add more geometry.
3. It is now possible to slide the top and bottom edges of the bevel according to the volume by selecting them in the **Edge Mode** and pressing G twice to form an ellipsoidal shape.
4. After this, we can use the F key to fill the eye. It will remove the two vertical edges that come from the bevel.
5. Now we can do a series of insets and extrusions to pop out the eye.
6. Applying the same method to the other side, we will add a little cartoon effect.
7. As always, we now need to clear the geometry by removing N-Gons using the knife.



Modeling the chest

It is time to model the chest. The following steps will help you do so:

1. Now we put the 3D cursor at the center of the space (*Shift + S*), then we add a box (*Shift + A*) for the base shape of the chest.
2. We can adjust the position under the head (*G + Z*) in the front view, and switch to the **Edit Mode** to start the modeling.
3. It is much faster to work with the symmetry, so we are going to cut the cube at its center with an edge loop (*Ctrl + R*). We select all the vertices on the left-hand side with the box select tool (*Ctrl + B*) in the wireframe shading mode (*Z*) and we delete them (*X*).
4. In the object mode (*Tab*), we add a mirror modifier to work with symmetry.
5. More polygons can be added to create the basic shape of the chest. We, therefore, add two vertical edge loops (*Ctrl + R* and scroll the mouse wheel up) from the side view and slide these on the *y* axis (*S + Y*).
6. Next, we move the polygon situated at the center of the chest from the side view (*G + X*) and add another vertical edge loop from the front view.
7. In the orthographic (5) front view (*I*), and with Wireframe (*Z*) Shading activated, we can easily work the shape by moving (*G*) and rotating (*R*) the selected vertices (refer to **2** and **3** in the following screenshot). Then we move the top face up a little (*G + Z*).
8. We can see that our central edge loop is not very well aligned in the front view. Therefore, we select and replace it by applying a zero scale on the *x* axis (*S + X + 0* on the numeric keypad), which perfectly aligns our selected vertices.
9. In order to have a less angular shape, we are going to activate the edge mode and select the edges at the top and bottom of the chest (press *Shift + Alt* and the LMB) to finally do a bevel (*Ctrl + B*) (refer to **5** in the following screenshot).
10. To balance the polygon flow of the mesh, we add a horizontal edge loop to the center (*Ctrl + R*) that we will extend with the Scale tool on the *y* axis (*S+Y*) (refer to **6** in the following screenshot). The goal is to use the maximum area of the drawing as a reference. Always remember to check the silhouette of your object.
11. For a more curved shape, we will use **Proportional Editing** that can be activated by the small circle icon located at the header of the 3D view.
12. The **Proportional Editing** tool will help us to shrink the side of the bust. Be careful to check the behavior of the clipping option of the mirror modifier for the vertices located on the axis of symmetry. They don't have to be merged at the center. To smooth the model, we will add a **Subdivision Surface** modifier and will check the **Smooth Shading** option in the **Transform** panel (*T*) in order to remove the flat aspect of the polygons.



Note

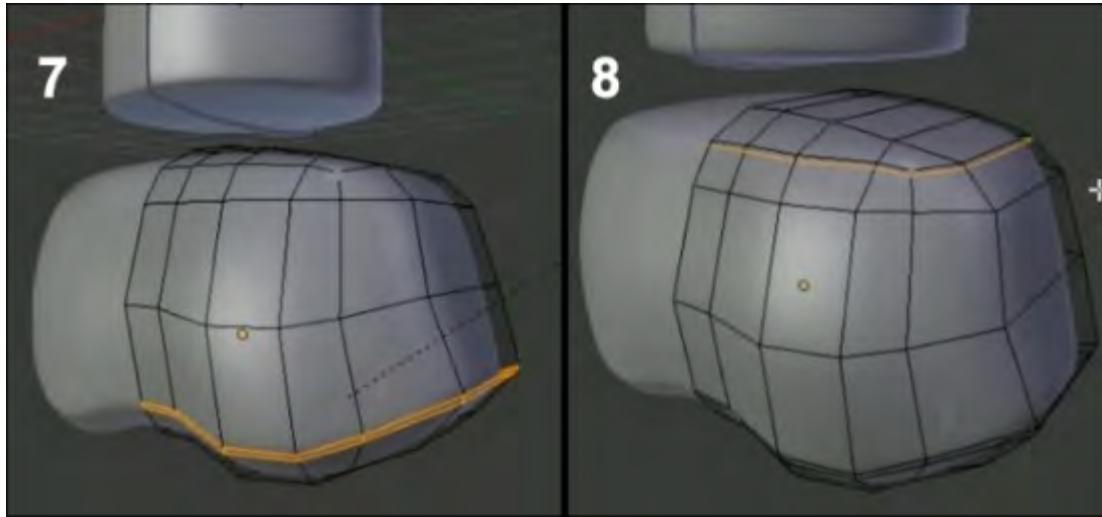
The Proportional Editing tool

This allows you to change the shape of an object in the **Edit Mode** in a smooth manner. It acts like a magnet for unselected components that are inside a circle of influence that you can adjust by scrolling the mouse wheel. This is perfect when you want to move a set of components and when the geometry is too compact.

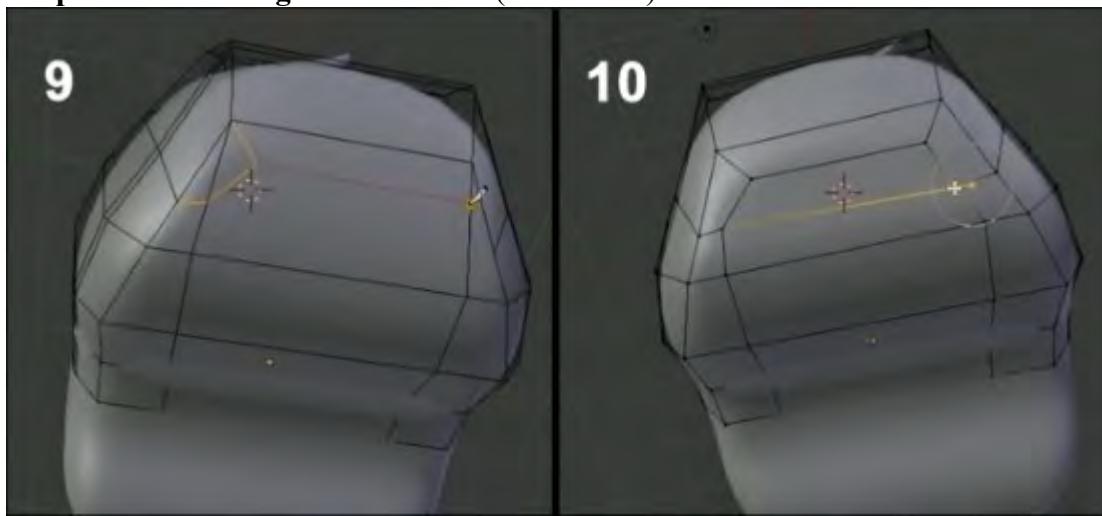
The **Connected** option allows you to limit the scope to the geometry connected to the selection.

The **Falloff** option offers a series of attenuation curve profiles.

13. With some bevels and additional edge loops, we will just sharpen some curves (refer to **7** and **8** in the following screenshot).

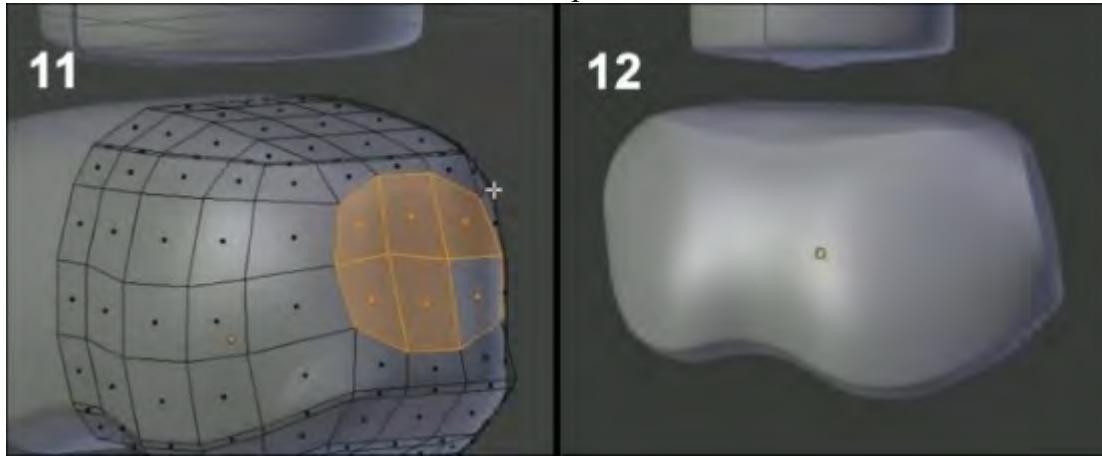


14. To flatten the top a little, we will select the highest faces and scale them on the Z axis with the **Proportional Editing** tool turned on. (S+Z and O).



15. From the bottom view, we use the Knife Topology tool (**K**) to change the organization of our vertices, edges, and faces (refer to **9** and **10** in the preceding screenshot). The process of arranging the components in the order that they best fit the shape is called "searching for a good topology". An easy way to grasp good topology is to remember that the edge loops are going to wrap around the object you want to create. Another thing that we already talked about is using only quads because they are easier to manage.
16. We form a loop to allow a better subdivision of the surface. Other edge loops can be added again horizontally and vertically to add details.
17. Then we switch to **Face Mode** and select six faces on the side of the bust in order to round them off with the LoopTools **Circle** feature (press **W** and select **LoopTool | Circle**) (refer to **11** in the following screenshot). We can choose the influence of this tool at the bottom of the left panel (**T**). A value of 80 percent will best fit here.

18. Then we adjust the angle of these faces with a rotation on the X axis ($R + X$). To quickly select polygons, we advise you to use the Circle Select (press *C* and the LMB) tool, which is used like a brush.
19. We do a small inset (*I*) to sharpen the geometry at the location of the shoulders.
20. We continue to move some vertices (*G*) here and there to gradually round the shape, and we adjust sets of vertices by rotating them (*R*). It is important to always navigate around your object to have the correct silhouette from different points of view. This is the essence of 3D modeling.

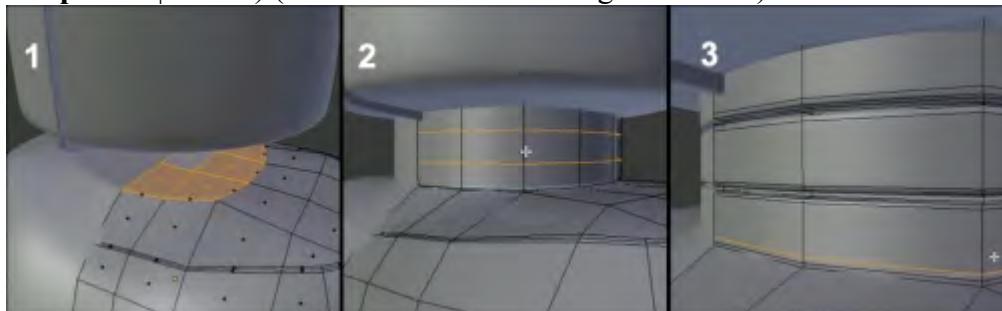


21. If you want more sharp edges, you can add a few edge loops (*Ctrl + R*) around them. Remember to use the smoothness parameter of the LoopCut tool (a value ranging from 0 to 1) in the left panel (*T*).

Modeling the neck

Still working on the bust in the **Edit Mode**, we will again flatten the top part of the bust and more precisely the area.

1. We start with the neck by selecting six polygons and scaling them on the *z* axis (*S + Z + 0* on the numeric keypad).
2. These faces will be arranged in a circle with the LoopTool Circle (by pressing *W* and selecting **LoopTools | Circle**) (refer to 1 in the following screenshot).



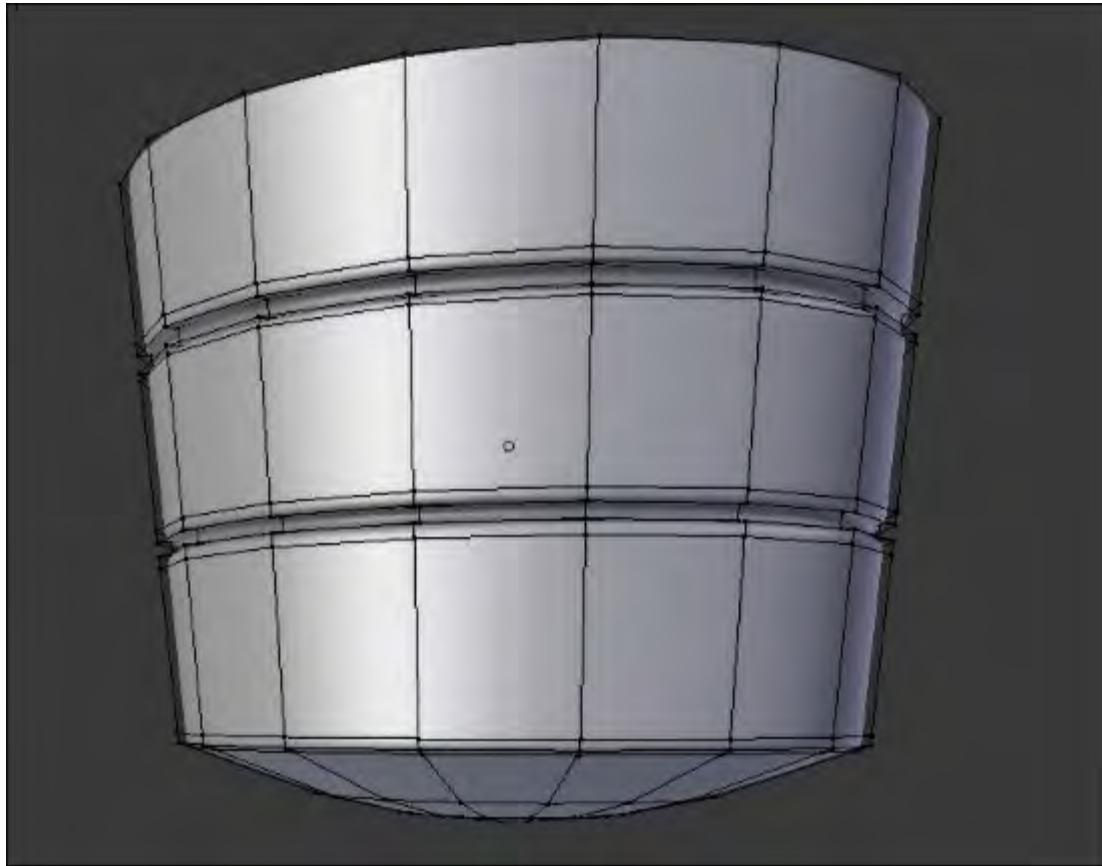
3. Then we make a very light extrusion (*E*) to hold the lower part of the neck.

4. Afterwards, we continue with another extrusion that penetrates in to the head.
5. In the Wireframe Shading mode, we remove the nonvisible face that is inside the head, which is useless.
6. Then we add two loops cuts (*Ctrl + R* and press *MMB*) (refer to **2** in the preceding screenshot) that we will divide into thinner edge loops with a slight bevel (*Ctrl + B*). These newly created face loops will be extruded (*E*) and scaled on the *x* and *y* axes (*S + Shift + Z*) (refer to **3** in the preceding screenshot).
7. As always, we will maintain the shape by adding edge loops with the LoopCut tool (*Ctrl + R*).
8. We end with a small extrusion (*E*) at the bottom to get the neck demarcation.

Modeling the torso

We will now work on the torso, which shares essentially the same modeling techniques as the neck. This will be done as follows:

1. We now snap the cursor on a vertex that lies on the symmetry axis of the lower part of the chest by opening the Snap floating menu with *Shift + S*. We select the fourth option.
2. We add a new cylinder (*Shift + A*) with 16 vertices (the number of vertices could be changed in the left panel by pressing *T*). By choosing this number of vertices, we get a cylinder that can be mirrored at its center and that has enough faces. We place the cylinder under the chest and we remove the top face that is not visible.
3. Next, we select the top edge loop and we change its scale on the *x* axis (*S + X*) and the bottom one on the *y* axis (*S + Y*).
4. We add two edge loops (*Ctrl + R*), add a bevel (*Ctrl + B*) to each of them, and finally, extrude (*E*) them inside (refer to the neck section).
5. We will round the lower part of the torso with a series of extrusions.



6. To get a smooth surface, we again need a **Subdivision Surface** modifier with the **Smooth Face Shading** option (Left panel: *T*).
7. We sharpen the edges with some edge loops (*Ctrl + R*).
8. Then, we clear our topology by solving the N-Gon with the same technique that we've used for the head.

Modeling the buttons

If you have followed the techniques used previously, the buttons are quite easy to make. The following steps are used to create the buttons:

1. We add a cylinder (with 16 vertices again), which we adjust in size and rotation with the Scaling (*S*) and Rotating tool (*R*). You can have a Free Rotation in all axes by pressing the *R* key twice. Don't forget this tool; it is very useful for aligning objects from a certain point of view!
2. We define the shape with a bevel at the top and an inward and outward extrusion.
3. We sharpen the shape with the LoopCut tool (*Ctrl + R*) and the Bevel tool (*Ctrl + B*).
4. The basic shape of one button can be achieved with only these tools (Extrusion, Bevel, and LoopCut).
5. In the **Object Mode**, we duplicate this with the Duplicate Linked tool (*Alt + D*).

Note

The Duplicate Object (Shift + D) and the Duplicate Linked (Alt + D) tools



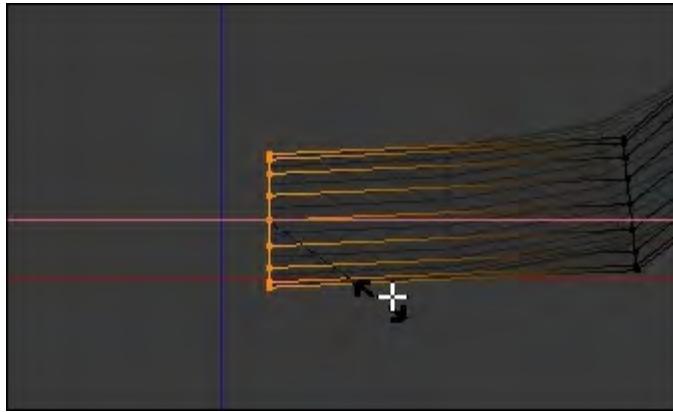
These both duplicate objects or components. The Duplicate Linked tool creates an instance of the object while in the **Object Mode**. The mesh data are connected. It means that, in the **Edit Mode**, any change of geometry will be reflected on the linked objects. However, the transformations done in the **Object Mode** are not reflected. If you want to break the link between two linked objects, press *U* (in the **Object Mode**) – **Make Single User** – **Object & Data**.

6. We will add a mirror modifier on each button. In the **Mirror Object** option, we select the torso. It will serve as the origin for the symmetry.

Modeling the fork

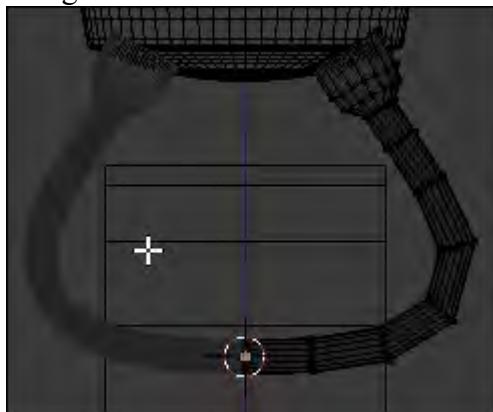
Now that we have finished the body, we will model the fork that covers the wheel, with the following steps:

1. In order to do this, we place the cursor to the right in the front orthographic view and add a cylinder (*Shift + A*).
2. In the **Active tool** options, we change the **Cap Fill Type** to **Nothing**. Our cylinder will have holes at the bottom and at the top.
3. Then we scale it in the **Object Mode** and rotate it with the *R* key, always in the front view for greater precision.
4. We can now add a **Subdivision Surface** modifier and apply **Smooth Shading**.
5. As always, we will maintain the shape with edge loops in the **Edit Mode**.
6. We select the outer edge loop and, pressing *Ctrl* and LMB, we extrude the fork in a twisted arch manner. This is the same tool that was used for the thunderbolt creation.
7. The last edge loop should be flattened on the *x* axis. To do this, we select it and press the *S + X* and *0* numpad key shortcut to constrain our scaling on the *x* axis and give it a value of 0.



Flattening the last (inner) edge loop.

8. We will mirror the half fork to the other side using a mirror modifier. But if we do it right now, we will have a problem with pivot point placement. We have to move the pivot of our object to the same location as our body. To do this, we select the body and using the *Shift + S* command, we select the fourth option, **Cursor to Selected** (Note that, in any floating menu, you can choose the option that you want by typing the corresponding key on your numpad).
9. Now that the cursor is placed at the pivot point of the body, we will map the origin (also called pivot) of our fork by selecting it and going to the **Object** menu in the 3D view header and selecting **Transform | Origin to 3D Cursor**. You can also get a pop-up menu with the same options as that of the **Transform** menu with the *Ctrl + Alt + Shift + C* shortcut (one of the longest in Blender's history). Our origin is at the same location as that of the body.
10. Now, we apply the rotation by pressing *Ctrl + A* and selecting **Rotation**. Applying the rotation is important here because we changed it in the **Object Mode**.
11. We can now safely add a **Mirror** modifier.
12. At this point, we will add a temporary cylinder that will represent the wheel. This will help us to correctly place the fork.
13. If you want to adjust the thickness of the fork tube, use the *Alt + S* shortcut to push the polygons along the normals.

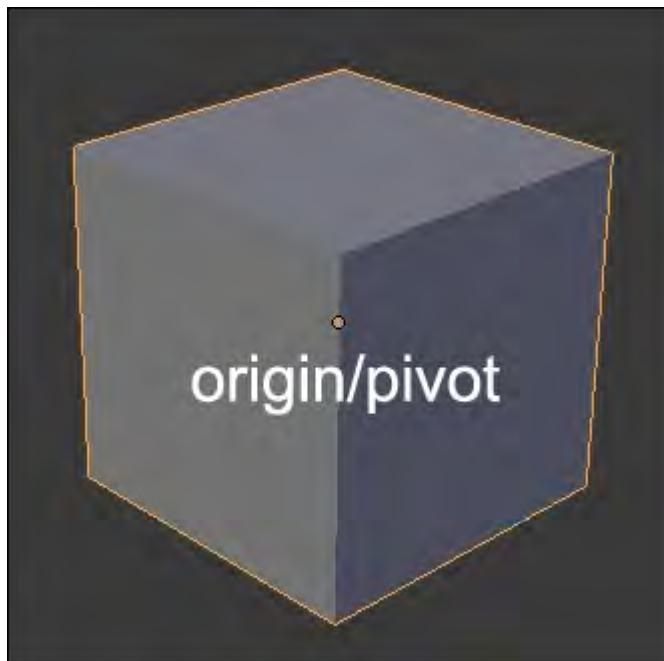


*The fork in the **Edit Mode**, with its mirror modifier and the temporary wheel.*

Note

About the origin/pivot

The origin, also called the pivot, is represented with a small origin circle in Blender. It determines the center of the mass of an object. Any rotation or scale modifications will take the origin into account by default. You can change the way these transformations work by using the **Pivot Point** drop-down menu in the 3D View header (next to the **Shading** drop-down menu).



We will revisit the fork later. It's now time to create protections that cover it.

Modeling protections for the fork

In order to model this piece, we go inside the view by pressing the **3** numpad key and perform the following set of operations:

1. We add a plane, and in the **Edit Mode**, do an inset.
2. After this, we add a loop cut in the middle with **Ctrl + R** and scale it on the **z** axis by pressing **S + Z**.
3. With the two outer vertices of the top selected, we do a scale constrained on the **x** axis. This will give us a pointed shape.
4. After this, we do a bevel of the center edge loop and add loop cuts horizontally and vertically.
5. We will then round the lower part of the shape. While the two middle edge loops are selected, we go inside the orthographic view and, with **Proportional Editing (O)** using a sphere curve, we move the vertices back to round the shape.

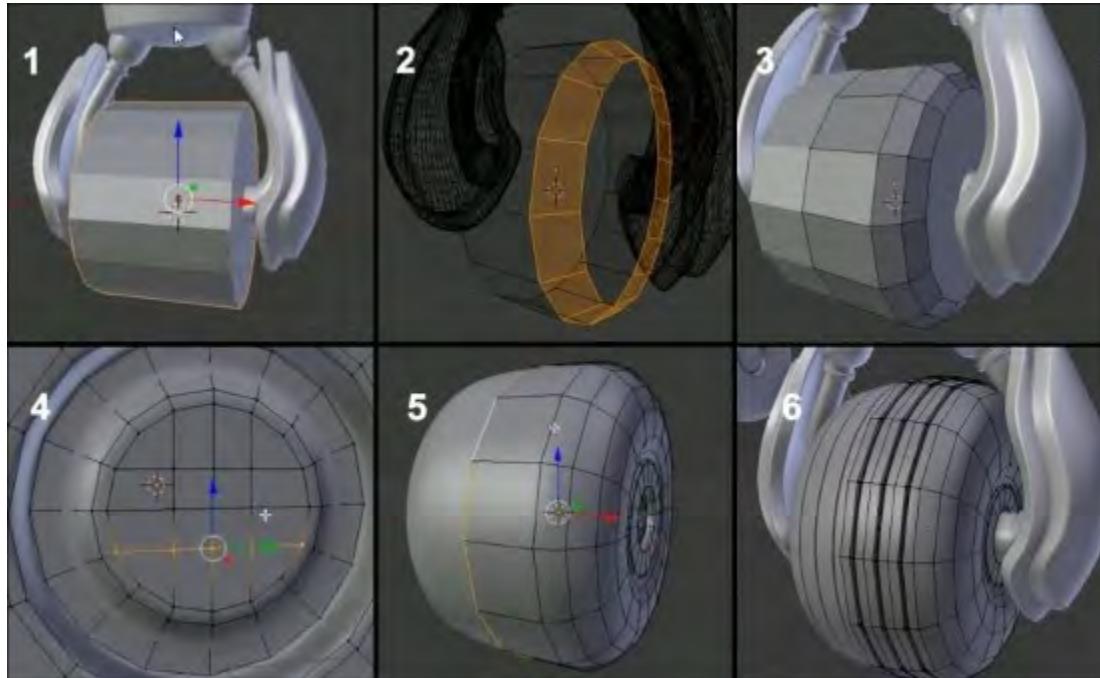
6. We can now place the piece near the fork in the **Object Mode** and, with the wireframe shading activated (Z) in the **Edit Mode**, we can adapt its silhouette by following the fork shape.
7. It's now time to use **Smooth** shading and the **Subdivision Surface** modifier in the **Object Mode**. We will add a new modifier that will add thickness to the object as if we were extruding it entirely. This modifier is called the **Solidify** modifier. You can tweak its **Thickness** slider to change the amount of thickness that you want. This modifier will be placed under the **Subdivision Surface** in order to be applied to it. If you now go into the **Edit Mode**, you will see that it has added a new geometry.
8. You can maintain the newly added thickness with loop cuts.
9. We added some details on the side using the Inset tool and by extruding the created faces.
10. We will now combine protections with the fork. To do this, we first select the protection and then the fork, and by pressing *Ctrl + J* we will join them in one mesh. As a result, the protection is mirrored because it is inside an object that has a mirror modifier. Note that, if you reverse the order of selection, you will join the fork in the protection. That's not what we want.
11. Going back to the fork, we can add decorations to it by adding two edge loops near the top.
12. We can extrude the face loop between these edge loops and scale them according to the normals with *E* and *Alt + S*.
13. Of course, we can sharpen the edges with the **LoopCut** tool.



The process of modeling the protections and the final result with the fork.

Modeling the main wheel

We will start modeling the wheel with the temporary cylinder that we have placed in the fork section. There are many methods to do this. We will do this here with the same tools that we introduced to you before.

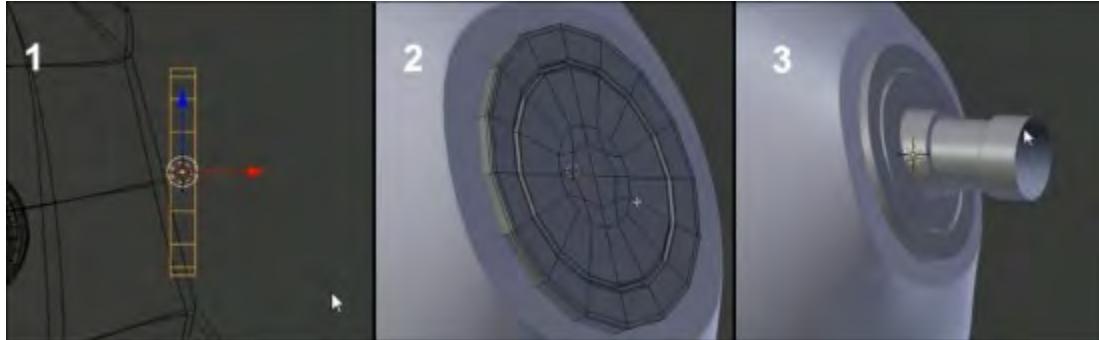


1. We will resize the wheel by enlarging it on the y and z axes (press $S + Shift + X$). When you press *Shift* and click on any axis during a transformation (rotation, scale, or grab), it will remove the constraint on that axis.
2. We then place our cylinder at the center of the forks.
3. Before you enter the **Edit Mode**, consider applying the rotation and the scale (press $Ctrl + A$ and select **Scale and Rotate**) to avoid unpleasant surprises.
4. In the **Edit Mode**, we add an edge loop at the center ($Ctrl + R$) and remove the faces on the left-hand side (press X and select **Delete faces**).
5. Then we can add a mirror modifier with the clipping option activated.
6. In the **Edges Mode**, we add several loop cuts using a **Bevel** (press $Ctrl + B$ and scroll the mouse wheel up) on the outer edge of the wheel (refer to 2 in the following screenshot), then a succession of insets and extrusions to form the side of the wheel. To work more easily on this, we will enter the **Local Mode** by pressing the slash key ($/$). This is like hiding all the other objects. If you want to leave the **Local Mode**, press the slash key again.
7. A **Subdivision Surface** can be added.
8. The N-Gon will also be transformed in quads on the side with the Vertex Connect Path (J) (refer to 4 in the preceding screenshot).
9. We will round the wheel by adding an edge loop to the center of it from the front view.
10. After this, we will add some grooves. For this, we will add five edge loops vertically on the front of the wheel (refer to 6 in the preceding screenshot). We will set the smoothness option to

- 1 in the last active tool panel in order to place the grooves without destroying the curve profile of the wheel.
11. Then we add a **Bevel** (*Ctrl + B*) and push its resulting faces by doing an **Extrude** (*E*) with a scale based on the normals (*Alt + S*).
 12. Then we accentuate every stripe with the **LoopCut** tool (*Ctrl + R*).

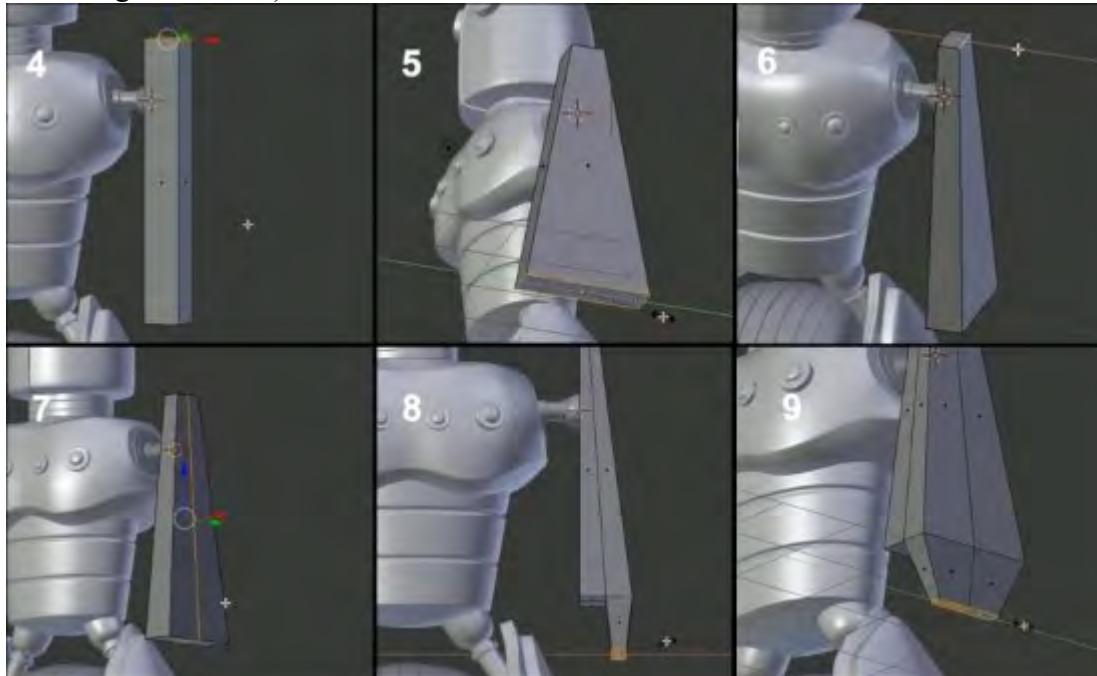
Modeling the arm

The arm is composed of four objects: the shoulder, the articulation ball, the arm, and the wheel. We will review the previous techniques by always focusing on the topology. Let's begin with the shoulder.

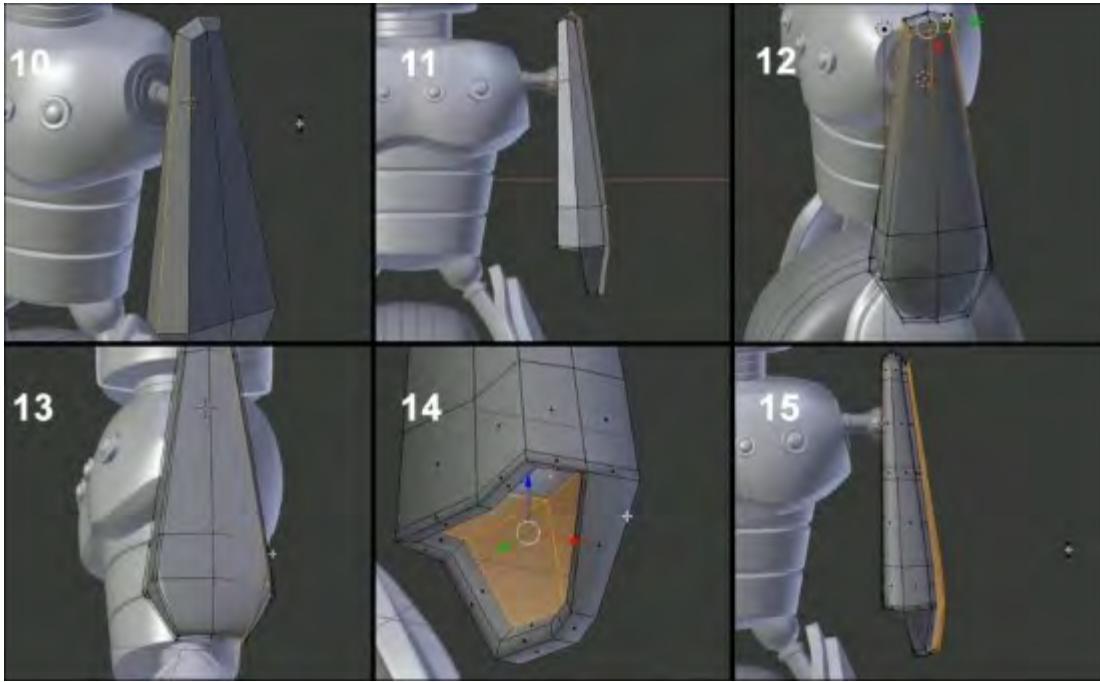


1. We will start with the arm articulation. For this, we simply add a cylinder that will be placed at the right shoulder location (refer to 1 in the preceding screenshot). It must be correctly oriented with a rotation (*R*) and flattened in the **Object Mode** with the Scale tool (*S*). When doing this, remember to constrain on the correct axis of transformation. To do these manipulations, it is better to be in orthographic view. Whenever we transform an object in the **Object Mode**, we don't forget to apply these transformations (*Ctrl + A*).
2. Then, we do a series of extrusions (*E*) in the **Local Mode** (*/*) and remove the face that enters the chest and thus will not be visible. Avoiding polygons that are not displayed is a good practice. Not doing this might lead to wastage of your computer resources and this is especially true for complex scenes with many polygons.
3. We can add details by digging a face loop (extrude and scale on the normals with *Alt + S*), which can be created with two edge loops (refer to 2 in the preceding screenshot).
4. We add a **Subdivision Subsurface** modifier and we apply **Smooth Shading**.
5. Then we extrude the tip that will hold the articulation ball. This extrusion will be flattened on the *x* axis (*S + X + 0* numpad key) (refer to 3 in the preceding screenshot).
6. We add a sphere by placing the cursor in the middle of the last edge loop right at the tip of the articulation (press *Shift + S* and select **Cursor to Selected**).
7. The cursor is in the right place within the **Object Mode**, so we add a UV sphere (*Shift + A*). We lower the number of segments to **16** and the number of rings to **8** in the last active tool panel.
8. We add a **Subdivision Surface** and apply a **Smooth Shading**.
9. We can delete the two vertices that are located on either side of the sphere as they will be hidden by other objects (press *X* and select **Delete vertices**).
10. We place the cursor at the center of the sphere and add a cube.

11. We need to resize the cube and then, in the **Edit Mode**, we move the top and bottom faces on the z axis ($G + Z$) to set the height of the arm (refer to **4** in the following screenshot).
12. With the bottom face selected, we make a scale on the Y axis (refer to **5** in the following screenshot).
13. We select the external edge of the top face and slightly move it on the x axis (refer to **6** in the following screenshot).



14. We add an edge loop to the center (refer to **7** in the preceding screenshot) in the side view and round the shape by slightly moving it upwards.
15. Then, we add a vertical edge loop in the front view in order to add the needed geometry for the protection of the wheel with an extrusion (refer to **8** in the preceding screenshot).
16. We now select the two right faces from the bottom view ($Ctrl + 7$).
17. We scale the faces slightly.
18. We will round the profile of the arm by selecting the middle edge loop and by pushing it along the normals ($Alt + S$).
19. It's time to add our lovely **Subdivision Subsurface** modifier and remove the flat shading. As always, we will maintain the sharp angles with a couple of edge loops.
20. We will then select the four inner faces of the hand and do an inset. This will create a face loop delimiting the inner part of the hand. These inner faces will then be extruded inward in order to create the hole that will keep the wheel.
21. We will then do an inset of the external faces of the arm.
22. These faces will then be extruded to create a small thickness.
23. It's now time to add the cylinder primitive of the hand.
24. We place it at the right location and change its size. As always, we will apply the transformation ($Ctrl + A$).



25. It will be easier to model the wheel with a mirror modifier. So we will cut the cylinder into two equal parts (*Ctrl + R*), and we will remove the left side of it to add the Mirror modifier with the clipping option checked on.
26. With multiples insets and extrusions, we then construct the wheel while in the Local view (/) (refer to **18** in the following screenshot).



27. We will again use the subsurface modifier with the **Smooth Shading** option.
28. After we have shaped the silhouette of the wheel, we will add asymmetric details on its left-hand side by applying the mirror modifier. Add a hole here with your best friends: the Inset and Extrude tools.
29. We will then fix the hand to encapsulate the wheel in it with precision. For this, we will use **Proportional Editing**.
30. Maybe you've seen that there are bad tensions in the right-angle form of the hand and the arm due to the **Subdivision Surface** modifier. These kind of artifacts warns you of a bad topology.

So we will, of course, find a way to correct this. To do this, we use the Knife tool (*K*) and we cut by following the hand outline (refer to **16** in the preceding screenshot).

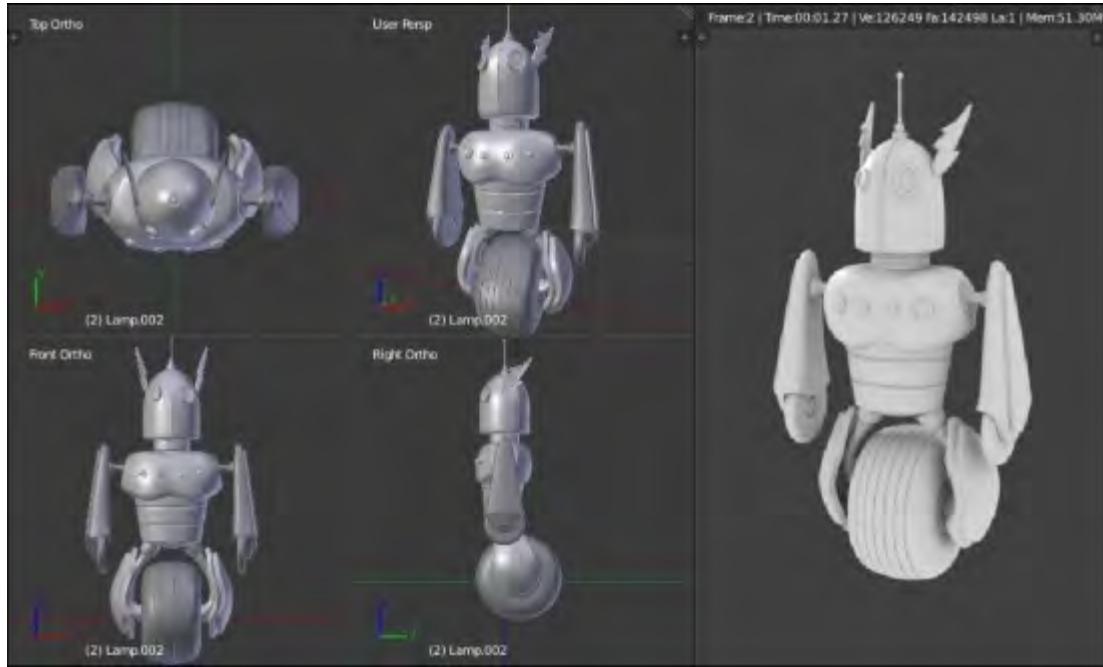
31. We can use the Merge tool (press *Alt + M* and select **At center**) in order to merge the two vertices that are part of the newly created triangle.
32. We will also close the newly created N-Gon with the Vertex Connect Path tool (*J*).
33. The **Bevel** tool will be useful to maintain the inner border of the hand.
34. We will now mirror the arm, its clip, and its wheel in the **Object Mode** with the mirror modifier using the chest as the mirror object.
35. You can always push your modeling further by adjusting the transformation of each part and by adding details with the tools that we have introduced you to (such as the LoopCut tool, Extrude, and Scale along the normals).

All the parts of the robot are done now! Congratulation!

Using Blender Internal to render our Robot Toy

We will now select the camera of our scene, and in a new 3D editor, we will see through it. To do so, we'll perform the following set of steps:

1. We will split the 3D view, and in the newly created editor (which should be 3D view), we will press the *0* numpad key.
2. We can move or rotate the camera like a normal object. Notice that the camera is shaped like a triangle. This represents the field of the view of your camera. If you want to place your camera while navigating around the robot, you can press the *Ctrl + 0* shortcut.
3. Now that our camera is correctly placed (that is, focusing on our robot), we can try to render the scene. We will do a very basic render by activating **Ambient Occlusion** in the **Additive Mode**. This option is located under the World icon button in the Properties editor. You just need to check the corresponding check box.
4. In order to do a render, we will press the *F12* key or go to the Camera icon button in the Properties editor and press the big **Render** button. Blender will automatically switch your current 3D View to a **UV/Image Editor** that will show you the calculated image. You can, of course, switch it back to a 3D view as we showed you in the first chapter.



You've now completed the Robot Toy project!

Summary

In this chapter, you learned to use the main modeling tools in Blender. You will keep learning about the other ways of modeling in the next chapters. Polygons can be seen as virtual clay. There are some rules to follow, but as soon as you know them, you will be free to model everything you wish

Chapter 2. Alien Character – Base Mesh Creation and Sculpting

In this chapter, you will discover a new way of modeling 3D objects with the powerful sculpting tools of Blender.

We will start with an overview of the sculpting process including brush settings and how to optimize the viewport. We will then create a base mesh with an amazing tool that Blender offers called the Skin modifier, which follows the concept art of an alien character.

Afterwards, we will sculpt the character using the tools that we had previously introduced and learn more about their usage in the different cases that are required for our character.

As sculpting is an artistic process, you will also learn about proportions and anatomy.

Let's jump to another planet! This chapter will cover the following topics:

- Understanding the sculpting process
- Optimizing the viewport
- Learning about and using brushes
- Creating a base mesh with the Skin modifier
- Using Dyntopo
- Understanding the basics of anatomy and proportions

You will start the sculpting of the following alien character (shown on the right) using a sketch as a reference (shown on the left). This is done with **Krita** (an open source tool for 2D art).



Understanding the sculpting process

Before starting to sculpt our alien, we will take some time to understand what this means and what the advantages are of using this modeling method. We will then give an overview of the basic tools that Blender has to offer.

An introduction to sculpting

Before the introduction of sculpting in the 3D world, there was only the polygonal modeling method (the method that we've used in the second chapter) that takes more time when creating organic shapes. The goal of sculpting is to have more freedom while modeling. The process looks a little bit like real sculpting art, but in this case we sculpt over a 3D mesh (our digital clay). When sculpting in Blender, we use brushes as tools that act on the mesh. There are many brushes that have different behaviors such as digging, moving, or pinching.

Choosing sculpting over poly modeling

Sculpting allows us to think more about the shape of the object and less about its technical part, such as its topology. So, the goal of this method is to really concentrate on the design part of the object. We won't see the vertices, edges, or polygons. The technique is more efficient when the goal is to reach an organic object. When you model with the tools that we have previously shown to you (the poly modeling method) you need to keep the topology in mind while researching the shape, and it is even more complicated when you have finer details. So what if we want to have a good topology with a sculpture? We have to do a retopology, but you'll see this in the next chapter

Using a pen tablet

When we modeled the Robot Toy in the previous chapter, we used a mouse. While we are sculpting, it's pretty hard to use a mouse because it is not precise according to the process. This is why we use a pen tablet that gives the sensibility needed to get the right shape. It takes some time to get used to this, but with practice you will have more control over your sculpture.

In order to navigate with the pen tablet in the 3D viewport, go to the **User Preferences** panel (*Ctrl + Alt + U*) and check the **Emulate 3 Button mouse** option. We will now be able to use the *Alt* key to navigate.

It's also a good thing to check the **Emulate Numpad** option in order to be able to switch views with the keys that are above the QWERTY keys.



A pen tablet with its stylus

The sculpt mode

In order to access all the tools needed for sculpting we need to go into the **Sculpt Mode**. The **Sculpt Mode** won't let us access the components of our mesh as in the **Edit Mode**, and we also won't be able to apply transformations on our objects as in the **Object Mode**. To switch to the **Sculpt Mode**, we select it in the drop-down menu located in the header of the viewport. As you can see, it is in the same place as the **Edit Mode** and the **Object Mode**.

Optimizing the viewport

Sculpting usually takes more resources than poly modeling because the number of polygons will quickly increase each time you want to add details. This is why we need to activate some settings that will boost our viewport. This is done as follows:

1. The first setting that we will check is located under the **System** tab in the **User Preferences** window (*Ctrl + Alt + U*) and it is called **VBOs**. It is used by **OpenGL** (the rendering API used by Blender) to better organize the data displayed on the screen.
2. In the **Options** tab, under the **Options** subpanel in the left panel of the viewport (in the **Sculpt Mode**), we will activate the **Fast Navigate** option.
3. We will also ensure that the **Double Sided** option is turned off. To do this, we can use a nice little add-on called **Sculpt Tool**. After the add-on is installed, on the **Sculpt** tab of the left panel of the viewport we now have the **Double Sided Off** option. Note that you can always access any option by pressing the Space key in the viewport and by typing the name of the tool that you want.
4. Later, when we sculpt our objects, we don't want to have something else other than our objects in the viewport. So we will deactivate the grid, the gizmos, and any other viewport information that we don't need.
5. In order to do this, we will go to the right panel of the viewport. We can open this by pressing the *N* key, and under the **Display** subpanel we will check the **Only Render** option.
6. By checking this option, we will simply deactivate all the options that are below the **Only Render** option, such as **Outline Selected** that consumes a lot of resources of the viewport. Remember this option as we will toggle it on or off depending on our needs.

Anatomy of a brush

As previously mentioned, we will use a lot of brushes that behave differently in order to sculpt our alien. In this section, we will take our hands over the settings that are shared between all the brushes with the **Sculpt/Draw** brush as an example. Let's perform the following set of steps:

1. In order to do our experimentation, we will use a Cube primitive. In the **Object Mode**, we place our cursor at the center of the scene (*Shift + C*) and we add a Cube (*Shift + A*). Note that you can use the one placed by default in any new scene if you want.
2. The Cube has a low polygonal resolution so we will have to subdivide it by going in to the **Edit Mode**, select all its components, and use the **Subdivide Smooth** option under the **Specials** menu (the *W* key). We will repeat this action six times in order to have a good density of polygons.
3. In the **Sculpt Mode**, we will then select the **Standard** brush (if not already selected) in the left panel of the 3D viewport by clicking on the brush icon button under the **Tools** tab (refer to **1** in the following screenshot).
4. We can now draw on the subdivided cube. As you can see, it pushes the geometry. This is because our brush has the **Add** option activated by default. If we want to go deeper, we will need to switch to the **Subtract** mode. The **Subtract** option simply reverses the behavior of the brush. Both the options are placed under the **Brush** subpanel in the **Tools** tab (refer to **2** and **3** in the following screenshot). It's not very convenient to click on buttons in order to do such a simple manipulation, so we encourage you to use the *Ctrl* key while sculpting on your mesh to switch between these modes.
5. As you may have seen, we are not really precise because of the size of our brush.
6. In order to change the size, we will use the corresponding slider under the brush icon called **Radius** (refer to **4** in the following screenshot). We can (and recommend this to you) use the *F* key as a shortcut.
7. Now that we have more control over the size of our brush, we will change its **Strength** under the **Radius** slider (refer to **5** in the following screenshot). We can use the *Shift + F* key as a shortcut. As you can see, on the right-hand side of both sliders (Strength and Radius), there is a little icon that allow us to use the pen tablet sensitivity in order to dynamically change these options. We will only use this for the Strength option, so when we lightly press on the pen tablet, we will have less strength than if we were pressing it harder.



Brush options

8. Another interesting thing that we can set for our brushes is **Texture** (also called an **alpha**). An alpha is usually a black and white image that is useful while adding details such as skin pores or patterns to an object. When an alpha is added to a brush, the black pixels will remove the brush behavior during sculpting. To import an alpha, we will first need to go in the **Textures** subpanel of the Properties panel (on the far left of the interface, by default) and click on the third icon (a checker pattern) (refer to **1** in the following screenshot).
9. We can now add a new texture, and under the **Image** subpanel, we can open an image. We can now go under the **Textures** subpanel of the **Tools** tab in order to select our newly imported texture (refer to **2** in the following screenshot).
10. If we want, we can also change the repetition of our alpha by changing the **X**, **Y**, and **Z** size sliders (refer to **3** in the following screenshot).



Adding a Texture (alpha) to our brush

11. The last setting that we will test is the **Curve** profile of our brush. The curve of the brush is located under the **Curve** subpanel in the **Tools** tab. Changing the curve profile allows us to change the behavior of the brush. For instance, with our current brush, **Sculpt/Draw**, if we click on the last icon (the flat curve) under the curve, we can see that the brush is harder while sculpting over our cube.
12. To understand this setting better, imagine that this is half of the profile of a real brush (see the following screenshot). We can select each point that composes the curve and move it to change the curve profile. We can also add a new point on the curve by clicking anywhere on it. When a point is selected, we can remove it by clicking on the **X** button. This curve is called a Bezier curve, so we can also change the smoothness of a point using the Tool icon and choosing the handle type that we want.



A modified curve of a brush.

Dyntopo versus the Multires modifier

In order to test our brush settings, we subdivided our cube by hand but it's not practical while sculpting an object because we didn't have enough control over the subdivision. In order to have control over our mesh, Blender gives us two main methods, the **Multires** (a.k.a. Multiresolution) modifier and **Dyntopo**.

First touch with the Multires modifier

The Multires modifier is added to the modifier stack of an object and allows us to maintain subdivision levels of sculpture. For instance, we can sculpt at a low level (with a low resolution), and the details will be transferred to the higher levels and vice versa. We will test it right now! This is done as follows:

1. We will first create a new Blender file (by navigating to **File | New**) and then with the default cube selected, we will go to the Properties panel in order to add a Multires modifier.

- We will subdivide our cube six times with the **Subdivide** button (refer to **1** in the following screenshot). If we were using a **Subdivision surface** modifier, our cube will be rounder. To test this out, we can go into the sculpt mode and start sculpting our cube with the Draw brush.
- We can now move between the different subdivision levels with the **Sculpt** slider of the **Multires** modifier (refer to **2** in the following screenshot). As you can see, we don't lose our sculpted information while changing levels, we are just changing the amount of details of the object. Of course, when you are at a lower level, you won't have as much detail as at the higher levels. The goal of all of this is to give you the possibility of changing the main shape of your sculpture at a low resolution without overwhelming yourself with all the details that you have sculpted at the higher levels. So don't try to add details too early in order to get the shape right and progressively increase the subdivisions.



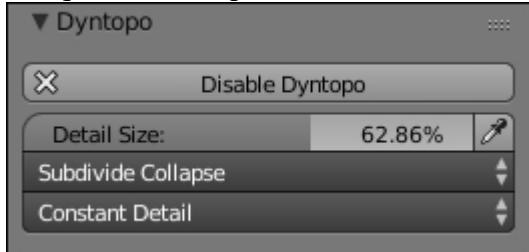
The Multires modifier with an example of three different levels of subdivisions

First touch with Dyntopo

The Dyntopo method will generate details according to the amount that we choose. The geometry will be subdivided when we sculpt an object and will be located where we have placed our mouse pointer. We will be using this method for our alien soon, so let's test this to get used to it:

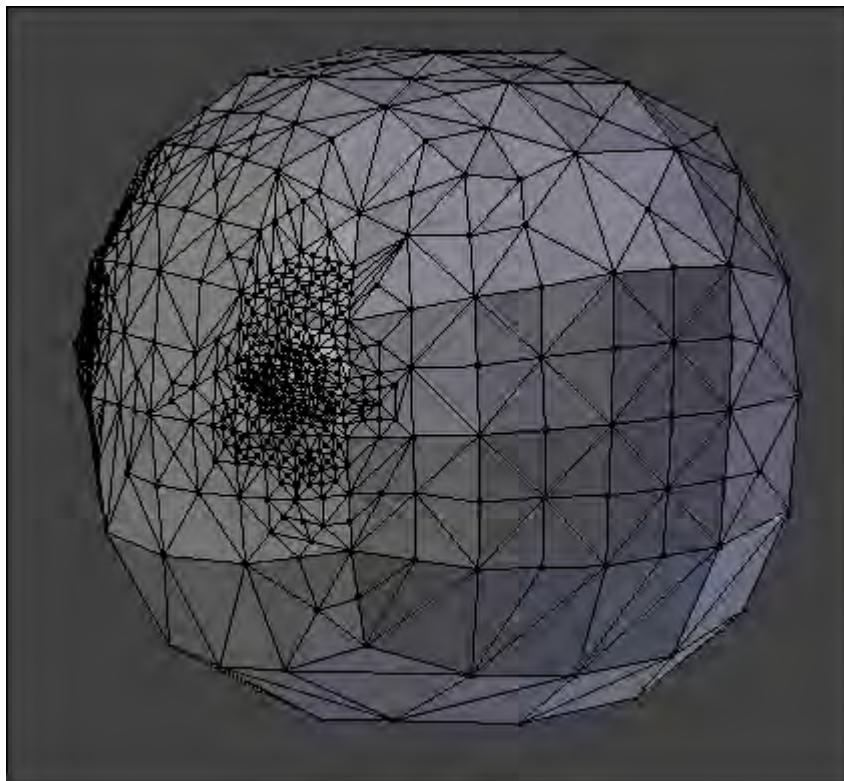
- We will first create a new Blender file (by navigating to **File | New**).
- The cube is a little bit low in resolution, so we will subdivide it twice with the **Subdivide Smooth** option (the **W** key).
- In the **Tools** tab, under the **Dyntopo** subpanel, we can activate Dyntopo by clicking on the **Enable Dyntopo** button. Our cube will now be converted to triangles (this is not a problem because remember, while sculpting an object, we don't care about its topology, we care about its shape). If you want to look at the wireframe of the object, use the **Z** key or simply go into the **Edit Mode**.
- By default, we are in **Relative Details** as you can see in the second drop-down menu. This option means that the amount of detail will be proportional to the distance of your working camera view. If we sculpt near the object, the amount of detail will be much more important than if we sculpt far from the object.

5. This method is nice, but there is another method that allows us to control the amount of detail without caring about our distance from the object. This is the constant details option (we will use this one for the alien). We can change from **Relative details** to **Constant Details** in the Sculpt details drop-down menu.



The Dyntopo settings

6. As you can see, we now have the detail size slider expressed by a percentage that allows us to change the amount of detail that our brush will generate on our mesh. With a small percentage, we will have finer details and vice versa.



A Dyntopo mesh with different levels of sculpted details.

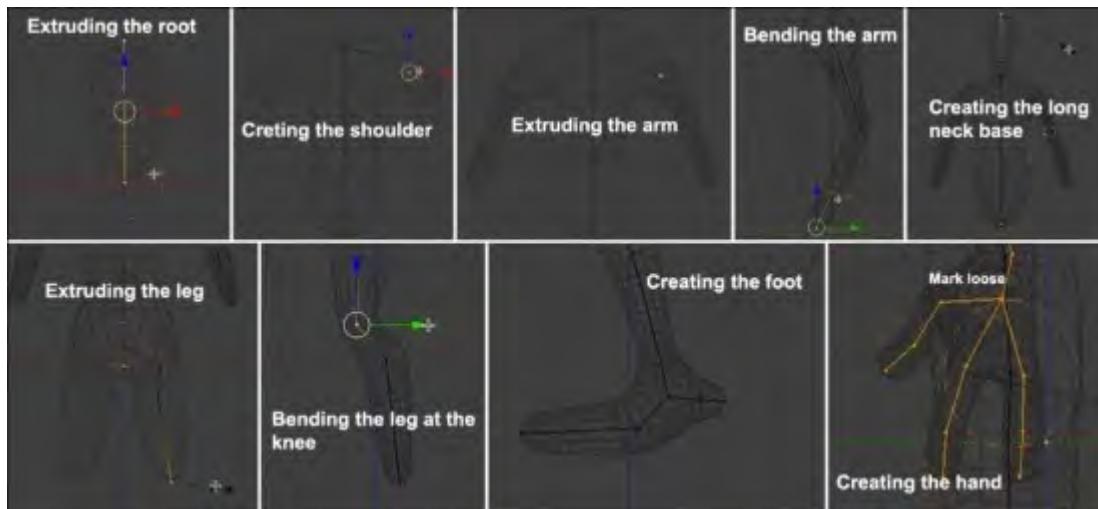
Creating a base mesh with the Skin modifier

Before we sculpt our alien, we need to have a base mesh that has roughly its proportions. If you want, you can use the methods that you've learned in the previous chapter in order to model it, but here we are going to use a cool modifier that Blender has to offer: the **Skin** modifier. Its goal is to create a geometry around each vertex. We can simply extrude some vertices as if we were doing a real wire armature, and the Skin modifier will add volume around it. For each vertex, we have control over the volume size.

Let's start our base mesh:

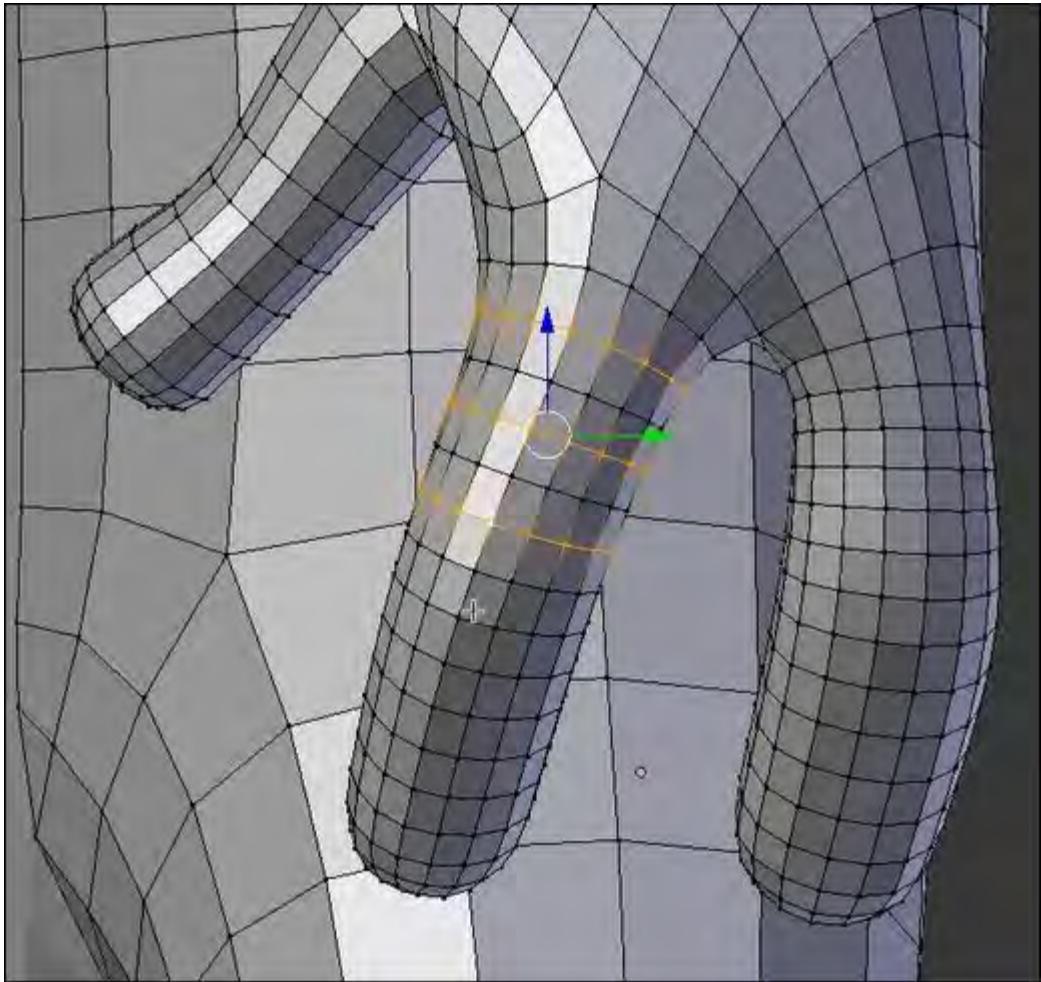
1. We will start by entering in to the **Edit Mode** of our default cube. Then we will select all the vertices (**A**) and merge them to a sole vertex at the center (press **Alt + M** and click on the center). We now have our root vertex that will be the pelvis of our alien.
2. It's now time to add a **Skin** modifier to our object in the modifier stack. As you can see, our vertex is controlling a new geometry around it. The geometry is low, so we will add a **Subdivision surface** modifier on top of the Skin modifier in order to have a smoother look. As you may have seen, the vertex has a red circle around it. This means that it is the root of our armature.
3. We can now extrude our vertex (**E**) on the **Z** axis in order to start the torso of our alien.
4. Now we will add a **Mirror** modifier with the **Clipping option** turned on. This modifier needs to be placed before the Skin and **Subdivision Surface** modifier (use the up and down arrows to move it to the first place). Ensure that the vertices that are on the symmetry axis are merged.
5. We can now select the top vertex (the base of the neck) and extrude it by pressing **Ctrl** and **LMB** to the right in order to create the shoulder. Be careful with the position of your vertex as the topology generated could be bad. Always try to move your vertices a little bit in order to see whether you can't have a better topology.
6. We will now change the size of the volume that has been generated around the shoulder vertex. To do this, we will use the **Ctrl + A** shortcut and we move our mouse to adjust it.
7. We will then extrude the arm. In order to give a more dynamic shape, we will bend it at the elbow. We then adjust its size. At this point, it is very important to match the proportions of the concept. Proportions means the length and size of the different members with respect to each other. If you need to constrain the size or change the volume on a certain axis, you can press **Ctrl + A + X, Y, or Z**.
8. Let's extrude the long neck of our alien.
9. It's now time to add the legs of our alien. We will do this by extruding the pelvis vertex at a 45 degree angle (press **Ctrl** and **LMB**). Then we extrude the leg and adjust its profile by changing the size of the different vertices (**Ctrl + A**). As we did for the arm, we will bend the leg a little bit at the knee location. Note that the pelvis vertex should be always marked as the root of the armature. If this is not the case, select it and in the **Skin modifier**, press the **Mark Root** button.
10. The foot will be then extruded and resized. We can create the heel by simply extruding the ankle vertex to the back. The ankle vertex needs to be marked as **Loose** in order to have a nice transition with the front and the heel of the foot. To do this, we select it and use the **Mark Loose** button in the Skin modifier.
11. It's now time to create the hand by extruding the wrist vertex. From the new vertex, we will extrude three fingers that will be rescaled appropriately. Note that the thumb is at a 45 degree angle from the other fingers. To add a more dynamic feeling to the hand, we will slightly bend the fingers inwards.

12. We will then extrude the base of the neck. From the newly created vertex, we can extrude the head vertex that will then be the base for the chin and the cranium.



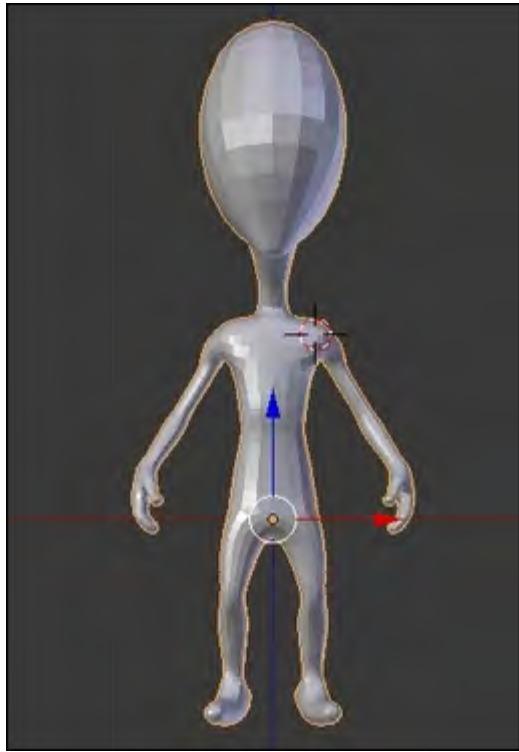
*The steps of the base mesh creation with the **Skin** modifier.*

13. We can now apply all our modifiers from the top to the bottom of the stack.
 14. If we enter in the **Edit Mode**, we can see that some parts are very dense. This is why we are going to remove some edge loops from certain parts such as the fingers. However, rather than doing this for both sides of the model, we are going to split the mesh in two and add a mirror modifier. To do this, we first need to ensure that there is a symmetry axis in the middle of our mesh. If this is not the case, you can use the knife tool (*K*) to create it. Then we can delete half of our model and add a mirror modifier as we did in the previous chapter. We can now select some edge loops where there is a lot of condensed geometry by pressing *Shift + Alt* and the RMB, and by pressing *X* we can delete them (delete the **edge loops**, not the vertices or faces).



Removing some edge loops of the dense parts.

15. The base mesh is now ready to be sculpted.



The final base mesh

Visual preparation

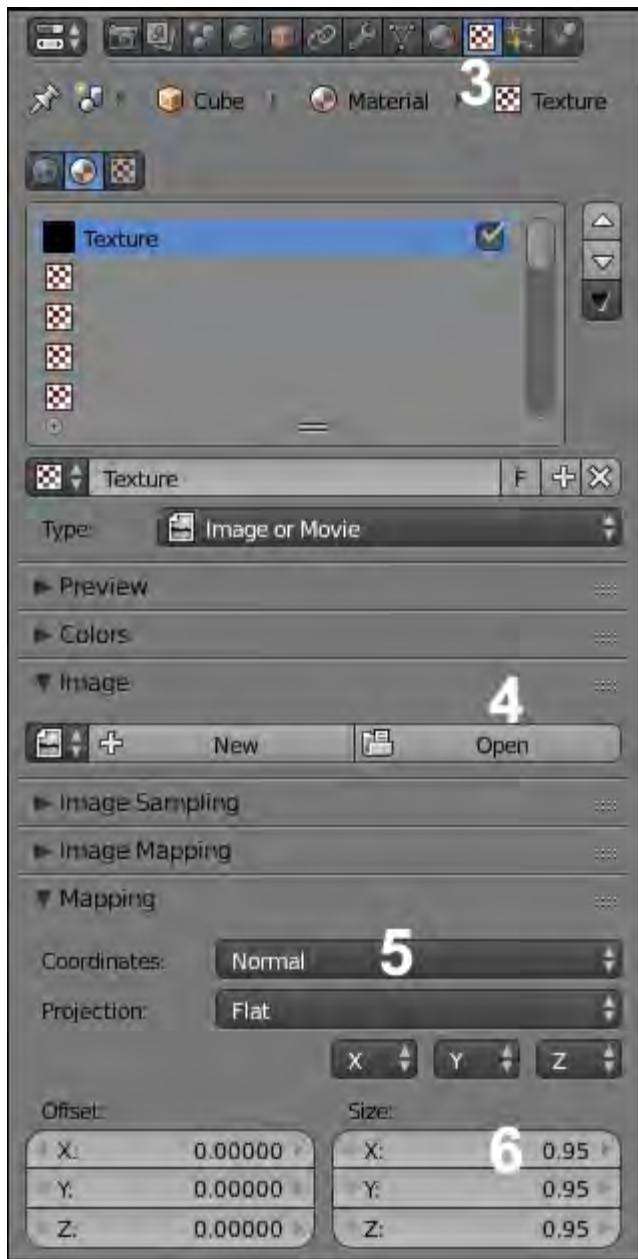
While sculpting, it's nice to use Matcap. It is simply an image that will be projected on your mesh in the viewport and that looks like a material. For instance, you can use a Matcap that reminds you of clay. Let's begin with our sculpting:

1. To set up a Matcap for our mesh, we will have to set up the default material of our mesh in the Properties panel under the **Material** tab (refer to 1 in the following screenshot). Note that if you can't see a material, you can press the **New** button.
2. Now we will check the **Shadeless** option under the **Shading** subpanel (refer to 1 in the following screenshot).



Creation of a new material with the Shadeless option.

3. As we have said before, a Matcap is an image, so we will import our image as the texture of our material. To do this, we go to the **Texture** tab (refer to 3 in the following screenshot) and we add a new texture by clicking on the **New** button. In the image subpanel, we will click on the **Open** button (refer to 4 in the following screenshot) and we choose our Matcap image.
4. A Matcap is mapped to a mesh according to its normals, so in the **Mapping** subpanel change the **Coordinates** from **UV** to **Normals** (refer to 5 in the following screenshot). We will also adjust the size of our projection by decreasing the **X**, **Y**, and **Z** size sliders to 0.95 (refer to 6 in the following screenshot).



Setting the Matcap image texture for our material

5. In order to see our Matcap in the viewport, we will replace the **Multitexture** display mode with **GLSL** in the right panel of the 3D view (*N*) under the **Shading** subpanel.
6. Last but not least, we will go into the **Shading** mode under **Texture Viewport** using the corresponding drop-down menu in the 3D view header. You can also use the *Alt + Z* shortcut to quickly switch to this mode. Our Matcap is now perfectly set up!



*Setting the **GLSL** display mode in the right panel of the viewport (N).*

An introduction to artistic anatomy

Before continuing with the alien creation, some basic knowledge of anatomy can be very useful. It is a basic discipline for a character artist. Don't worry, we will clarify the concerned parts of the body for each step with illustrations.

Of course it is an alien, so we can accept the fact that we don't always have to respect human anatomy for specific parts. He has a huge head, only two fingers and a thumb, and very different feet. His humanoid appearance imposes some anatomical likelihood. This is especially useful if you plan to animate it later on. Improving your knowledge of this topic will help you to understand the movements and postures better.

In the early 16th century, Leonardo Da Vinci was one of the first artists who tried to understand the human anatomy. While religious obscurantism prohibited the examination of corpses, he dared to defy this. By dissection and observation, he did many illustrations detailing the positioning of muscles, joints, nerves, and organs.

It won't be necessary in our case to know the scientific names by heart. It's rather important to know and understand their general forms. Overall, it's more about the comprehension of human body mechanisms.

Note

Many good books treat this subject. For more information, you can have a look at these websites:

- <http://md3dinc.com/>
- <https://www.anatomy4sculptors.com/>
- <http://www.3d.sk/>

Sculpting the body

We are continuing the modeling of our alien using Dyntopo as we had previously mentioned. This will allow the creation of the antennae very easily while they are not yet present in the base mesh.

The following is the preparation of our environment before sculpting:

1. We will set the optimizing options that were previously explained.
2. We can adjust the lens parameter in the right panel of the 3D viewport (*N*). A high value lessens the focal deformations.
3. We must check the mirror options in the left panel of the 3D viewport (*T*) to Symmetry/Lock by choosing the axis of symmetry. It is very important in order to save time.
4. We will then activate the **Dyntopo** option in the left panel of the viewport. A detail size of around 25 percent is enough to start. It depends on the size of your model.

For a better understanding, we are going to start sculpting by adding details by iteration.

The head

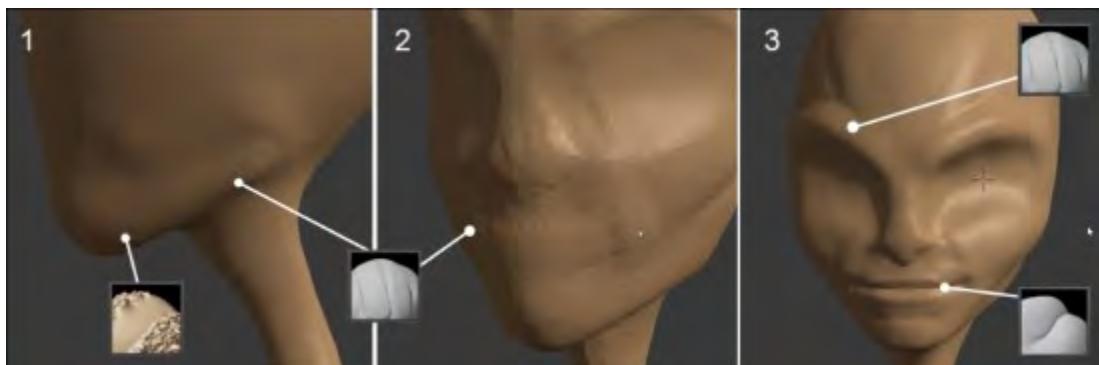
We start defining the jaw and chin with the **Clay Strips** brush (refer to 1 in the following screenshot). It is unnecessary to have too many details for the moment. While sculpting, always remember to define the main shapes (the volumes) and then gradually move towards the details.

We then accentuate the delimitation between the jaw and neck without exaggeration. A strength of **0.5** is enough.

Note

The Clay Strips brush

The **Clay Strips** brush is a very useful brush to define muscles, dig or add polygons in a straight direction with pretty sharp outlines. It is the equivalent of the Clay Buildup in Zbrush.



1. In order to dig into the polygons, we press *Ctrl* while sculpting. It allows us to switch to the **Subtract** mode on the fly.
2. Then we slightly smooth the added geometry with the **Smooth** brush for a better blend of the created shape. You need to remember to smooth the shape very often in order to avoid having something too grainy.

Note

The Smooth brush

This is often a very useful brush. It allows you to soften and smooth your shapes. This brush is so useful that there is a special shortcut that you absolutely need to know. Hold the Shift key while sculpting in order to use it. It works while using any brush. When you use it over a big density of polygons, it will be harder to smooth your shape, so it will be mainly useful before working on the details.



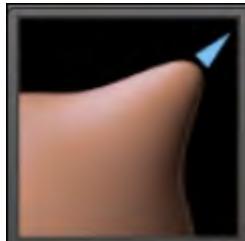
3. Now that the jaw is sketched, we will go to the side view. Then we have to adjust the silhouette of the neck, as well as the skull with the **Grab** brush. Our alien has a very curved neck.
4. We will adjust the shape of our model little by little by turning it around. It's very important to observe your model from different points of view.

Do not hesitate a moment to look at the reference sketch and position the alien in a very similar pose. You can also create a new 3D view editor to keep an eye on the view of your choice (in our case, we've used the orthographic side view).

Note

The Grab brush

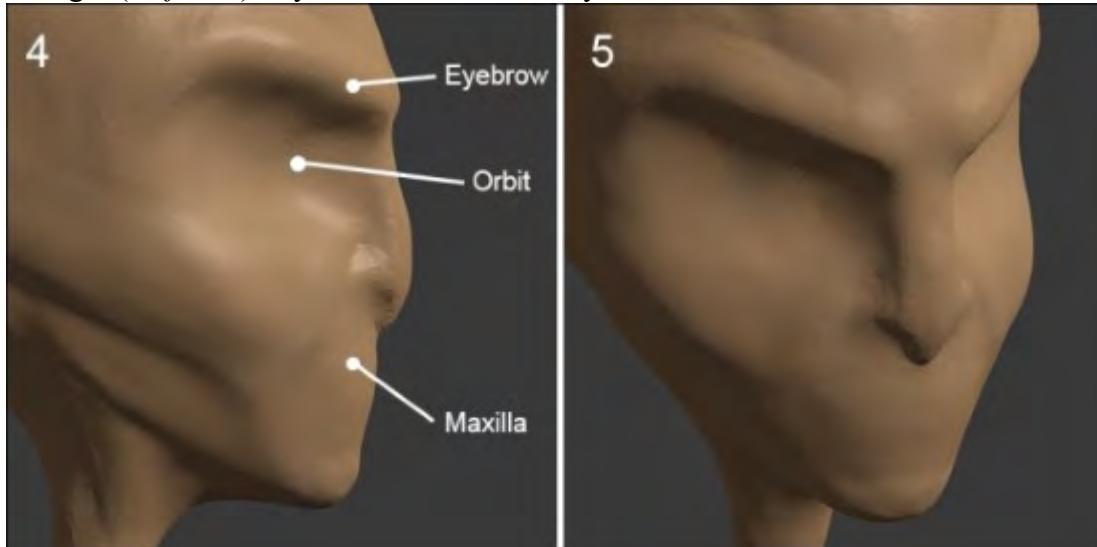
This is often a very useful brush when you start a new model. It works a little bit in the same way that the **Proportional Editing** tool does. If you have any difficulties using it, you can go to the **Curve** options in the left panel of the viewport and modify the curve to decrease its profile.



In our case, we are going to use a very soft curve for the **Smooth** brush.



- Again, we will take the **Clay Strips** brush to keep going on the face of the alien in order to dig the orbits and accentuate the eyebrows by adding matter (refer to **2** and **3** in the previous screenshot). Be careful to not add too much volume at this place. Do not hesitate to decrease the strength (*Shift + F*) of your brush, if necessary.



- We will also start to sketch the nose and the mouth (refer to **3** in the previous screenshot). Before detailing the lips, we need to start adding some new volume. We form a rounded edge created by the maxilla and the jaw. Remember that the front of a set of teeth has almost a semi-cylindrical shape. It's easier to create the mouth rounded volume from the bottom view.
- Once this is done, we can start sketching the opening of the mouth with the **Crease** brush, which allows us to draw a mined line. Consider the fact that the geometry is dynamic, so don't hesitate to add some resolution and pinch your shapes in order to make them more accentuated.

Note

The Crease brush

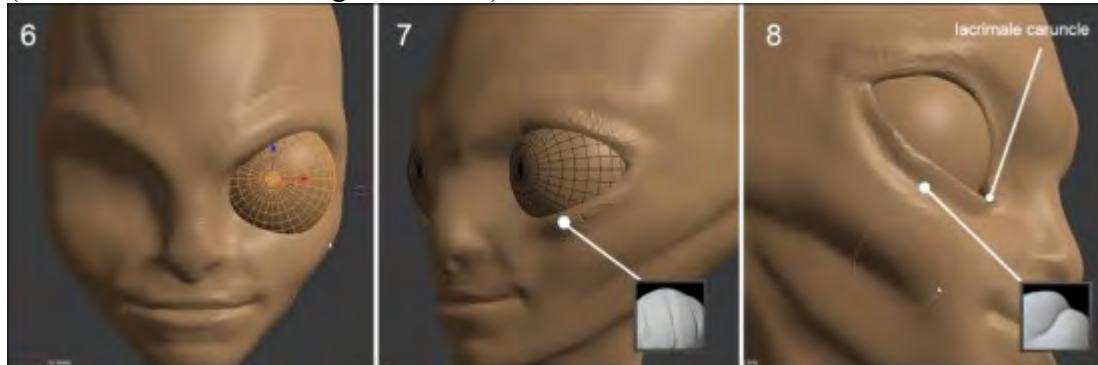
This brush will allow you to draw lines by digging or adding some volume to your shapes while being pinched. It is perfect to accentuate muscles and make them well visible. It is the equivalent of the Dam Standard in Zbrush.



4. We will again use the **Clay Strips** brush in order to give some volume to the lips.

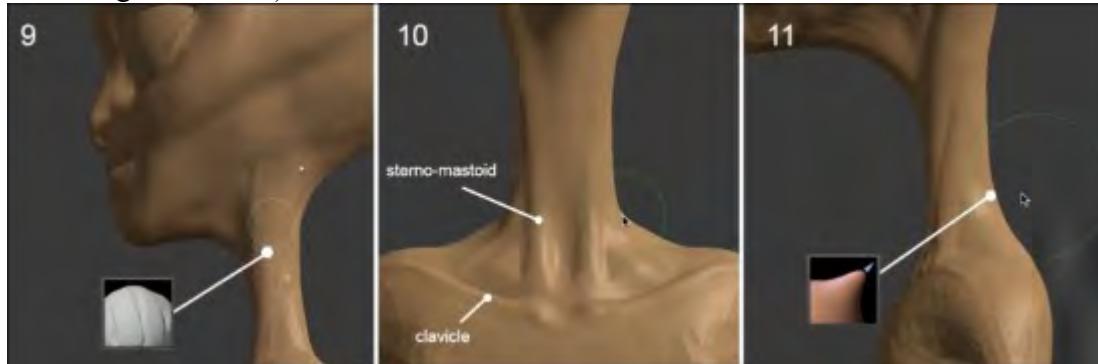
Remember to leave a little gap between the lower lip and the top lip. The top lip is a little bit more forward than the lower lip from a profile view.

5. In the face view (refer to 5 in the previous screenshot), we will go back to the **Object Mode (Tab)** and place the 3D cursor where we want the eye at the middle of the orbit. It doesn't matter if the orbit is not completely dug.
6. We will add **UVSphere** (*Shift + A*). We will resize this with the **Scale Tool (S)**, and we will position it (*G*) in the front view (*I*) just as in the side view (*3*). For a good placement of the eye, looking at the wireframe can be helpful. You only need to go to the Object Data tab and then to the **Group** in the Properties editor, and check the **Wire and Draw all Edges** option.
7. Then, we can sculpt around the eyelids with the **Clay Strips** brush by being careful to accentuate the outside and inside corner (the lacrimal caruncle) (refer to 7 in the following screenshot).
8. In order to accentuate the eyes, let's pinch the upper and lower eyelids with the Crease brush (refer to 8 in the following screenshot).



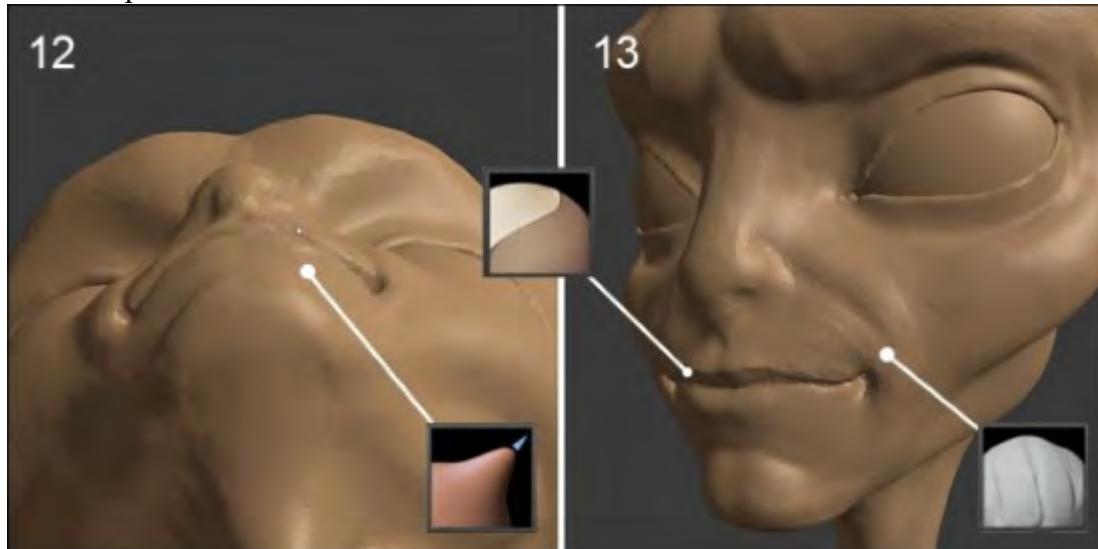
Finding a good position of the eye is not an easy task. You can move the facial structure with the **Grab** brush if you have problems. The top eyelid must be slightly forward. Conceptually, the eyes are inordinately big, so be careful that they don't touch each other.

9. Now that the face begins to take shape, we will continue with the neck. We adjust the shape a little bit more with the **Grab** brush, then we start sculpting the muscles and bones of the neck. We carve the clavicles with the **Clay Strips** brush.
10. Then we will move on the sterno-mastoid muscles. It is a muscle group that starts from the mastoid near the ear and attaches to the sternum and the clavicle. We keep working with the **Clay Strips** brush (refer to **9** and **10** in the following screenshot).
11. In the side view, we can polish the silhouette of the neck with the **Grab** brush (refer to **11** in the following screenshot).



Now let's refine the face:

1. We will come back to the mouth by adding some volume to accentuate the circular muscles around the mouth. Be careful to adjust the level details of Dyntopo to around 10 percent (refer to **13** in the following screenshot). As you may have seen, when you are sculpting an object, you don't directly get the shape that you want, so you always need to go back and forth over the different parts.



2. This brings us to accentuating the wrinkles that make the junction with the cheeks.
3. Then we will go and pinch the upper lips with the **Pinch/Magnify** brush. The lower lip doesn't need to be as pinched like the upper one. Again, you can increase the level detail of Dyntopo to around 7%.

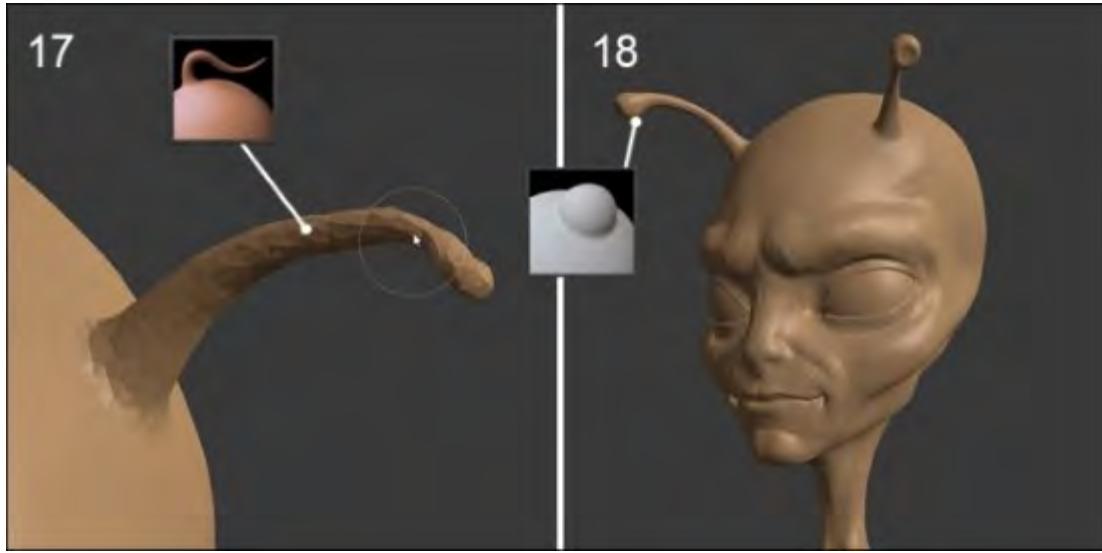
Note

The Pinch/Magnify brush

This brush allow us to pinch the polygons outwards or inwards (in the **Subtract** mode). It is perfect in our case to detail the lips, the wrinkles, or accentuate the contour of muscles. It is often used to get a cartoon style or for a hard surface modeling where you need to sculpt angular surfaces.



4. We will take a moment to turn the head, including the top view (**16**); then we adjust the round shape of the skull with the **Grab** brush.
5. Now, it's time to add the antennae (refer to **17** in the following image). For this, we are going to use the **Snake Hook** brush with a Dyntopo level detail of around 14%. The difficulty will be to find a good point of view of the head because we can't move the view while extracting the geometry with the **Snake Hook** brush. We must be positioned on the side in order to be able to extract the matter from a little area on the top of the forehead and stretch it outwards in a good direction. Do not hesitate to make several tries if this does not suit you. You can always adjust the size of the brush and the level of detail.



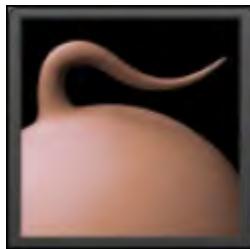
Note

Undo while being in the Sculpt Mode

Unfortunately, the Undo function of Blender is not very optimized for the moment in the **Sculpt Mode**. It can be very slow, so do not use it too often. In many cases, you can probably quickly fix your mistakes without Undo.

The Snake Hook brush

This is very useful to sculpt horns or tentacles. This brush is more interesting with Dyntopo. The problem with a mesh that uses a Multires modifier is that topology problems quickly appear with a lack of geometry. As long as the topology is dynamic, we can easily create an arm, a leg, or anything else. We can extend this as long as we wish the shape of our model to be.



6. Once the antennae are sculpted, we will add some polygons with the **Clay Strips** brush, then we will smooth them with the **Smooth** brush. We will magnify the extremity with the **Inflate/Deflate** brush (refer to 18 in the preceding screenshot).
7. We will end this by digging forward a little bit with the **Clay Strips** brush in order to break the rounded shape.

Note

The Inflate/Deflate brush

This brush will allow you to inflate volumes by pushing the polygons in the normal's direction, or the inverse in the opposite direction in the **Subtract** mode. It can be very useful for meshes with closely spaced surfaces that are difficult to sculpt. This gives a very fast volume that makes it much more comfortable for sculpting.

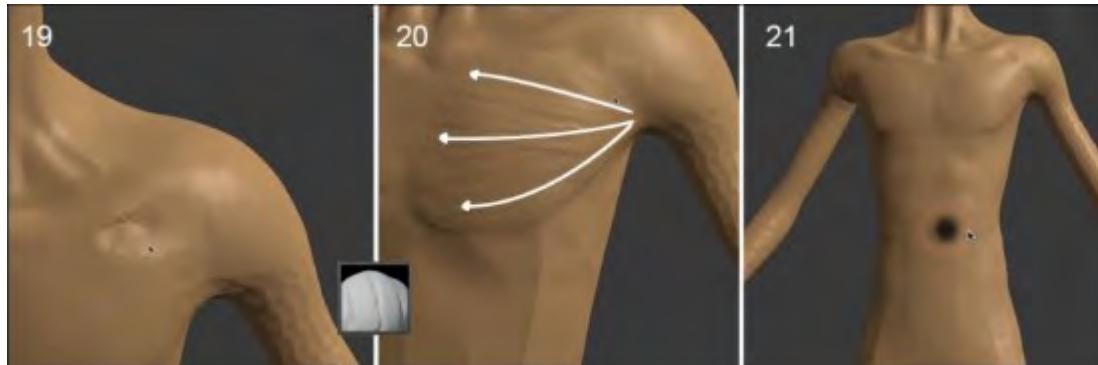


So our little alien has now its telepathy organs. We are going to sculpt the torso.

The torso

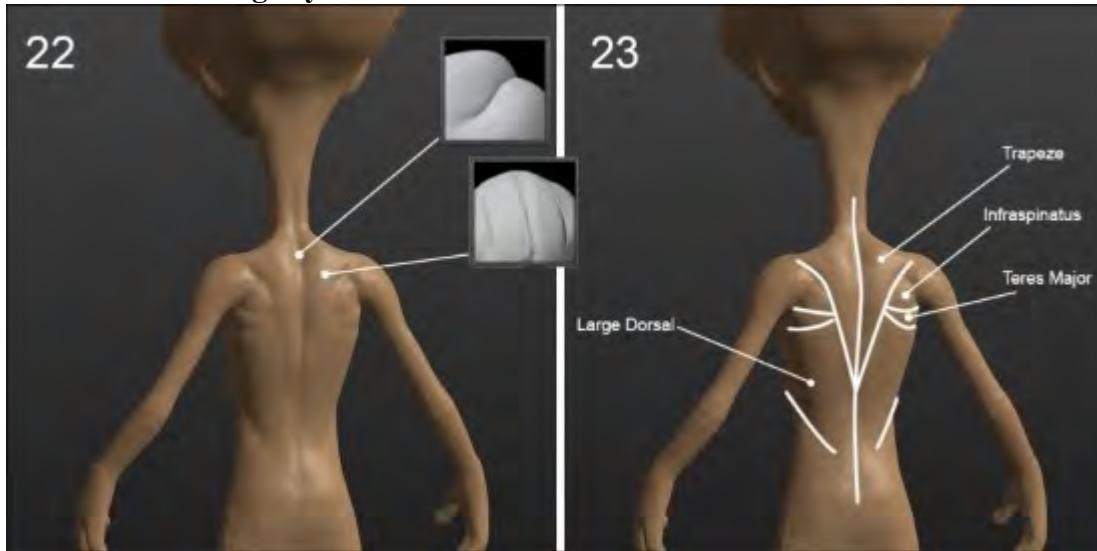
We will start the torso by sketching the pectoralis major muscle:

1. We will start by smoothing the surface and adding enough details.
2. With the **Clay Strips** brush, we will dig the dividing lines with the clavicle and the shoulder. (refer to **19** in the following screenshot).
3. Then gradually, we will add some volume accentuating the muscle fibers. The brush strokes start from the bottom of the shoulder at the clip with the biceps and go to the center of the chest (refer to **20** in the following screenshot). Get used to guiding your brush movements in the direction of the muscle.

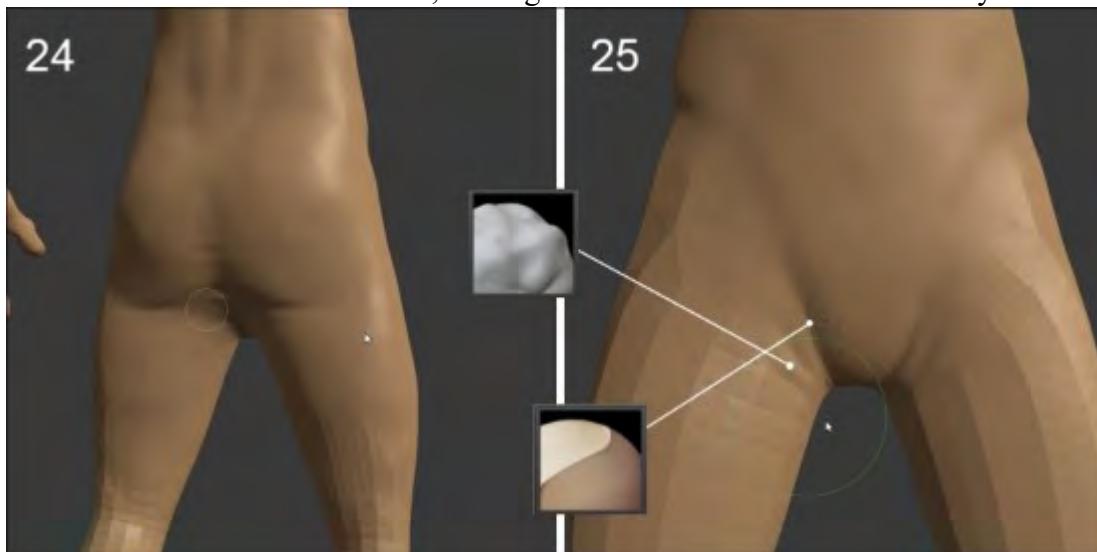


4. Once the pectoral is sculpted, we will slightly accentuate the bottom of the chest and the abs with very light touches of the **Clay Strip** brush. This is to suggest forms rather than showing them (refer to **21** in the preceding screenshot).

- We will then work on the back part of the alien. It is a complex part of the body. So we will start to draw the muscles (refer to **22** and **23** in the following screenshots) to gain visibility with the **Crease** brush. We can soften and smooth the muscle shapes. Then we will accentuate the spine with the **Pinch/Magnify** brush.



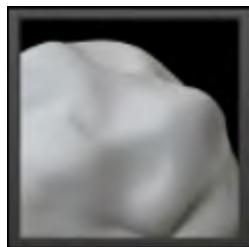
- Now we will sculpt the buttocks. Avoid putting too much volume here. Turn the model around and observe the side view a moment, if necessary adjust the silhouette. Remember to draw a pinch line to accentuate the bottom of the buttocks with the **Pinch/Magnify** brush. This forms a fold between the buttock and the thigh (refer to **24** in the following screenshot).
- We will add a few folds to show that it is a combination.
- We will use the **Pinch** brush to accentuate the lower abdomen and pelvic bones (refer to **25** in the following screenshot). Unless you desire a different sexual orientation for our alien, feel free to add some volume to his crotch, it brings a little more realism. Don't be shy.



Note

The Clay brush

This brush allows you to add planar relief with a few soft edges. It is quite close to the **Clay Strip** brush that you already know but with a less sharp effect. It adds volume by raising with a low intensity. It is very good to refine organic shapes with precision.



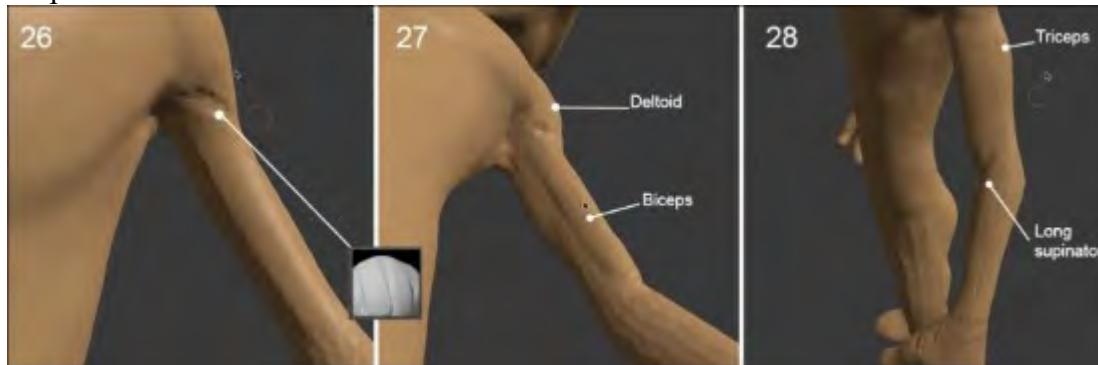
The arms

Now, let's start the arms. We can see in the drawing that they are pretty fine and not very muscular. His hands have two fingers and a thumb.

We won't need a lot of muscle details. We will be just interested in the major forms.

Let's begin the process:

1. We will start by digging the part between the shoulder and the biceps with the **Clay** brush (refer to **26** in the following screenshot). This accentuates the shoulders.
2. We will smooth a little, and then add some volume to the biceps (refer to **27** in the following screenshot). Slightly, we mark the outline of the muscles with the **Crease** brush and adjust the shape with the **Grab** brush.



3. Then we will work on the triceps that is located at the back of the arm. It is a muscle connected to the deltoid and covers the entire rear portion of the upper arm. We give it some volume by drawing the muscle fibers with some touches of the **Clay Strips** brush.

4. The forearm is a complex area of human anatomy. It is usually quite difficult to sculpt. It consists of several muscles that twist to ensure the mobility of the hand and the fingers. For our alien, we simplified it by lessening the muscle visibility. With the **Clay Strips** brush, we will draw the long supinator that emerges because it forms the junction with the end of the biceps (refer to **28** in the preceding screenshot) near the elbow. We will then add some volume to the elbow.
5. We will start from the elbow, and we mark a slight stroke of the **Clay Strips** brush in the direction of the wrist.
6. The wrist is reinforced by slightly accentuating the bones on the sides of the upper part.
7. We go and dig and slightly flatten the lower part of the wrist. (Refer to **30** in the following screenshot.)

The hand is also a fairly complex part of the body. We will not detail the anatomy here. We will try to focus on the main forms that compose it. Observe your own hands for better understanding of the forms.



Now, we will begin forming gristle on the upper part of the hand with the **Clay Strips** brush.

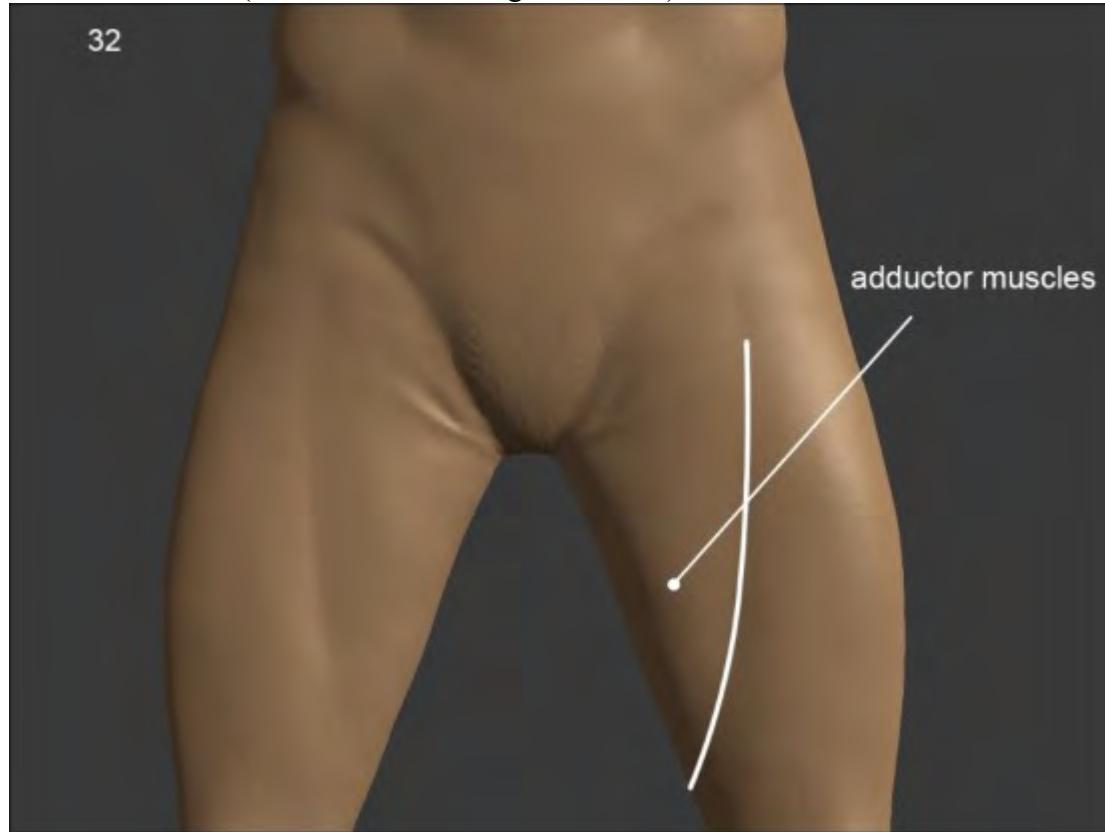
1. We have placed a bit of volume to the different phalanges in order to accentuate them. (Refer to **29** in the preceding screenshot.)
2. There is some skin between the two fingers and between the index finger and thumb that we will dig. This skin allows the flexibility and elasticity of finger movements.
3. We will take the **Crease** brush and mark the lower part of the phalanges where the folds of fingers will be.
4. We will keep working with the **Crease** brush and draw the lines of the hands. The three main lines are enough to give the appearance of a palm (refer to **30** in the preceding screenshot).

The legs

We continue our sculpture with the legs. We can see that he has quite muscular thighs. The feet have a dynamic style that reflects the legs of a rabbit.

1. As with other parts of the body previously created, we will adjust the silhouette of the legs with the **Grab** brush before detailing the shapes.

2. We will slightly dig a line from the hip to the inside of the thigh that allows us to accentuate the adductor muscles (refer to the following screenshot).



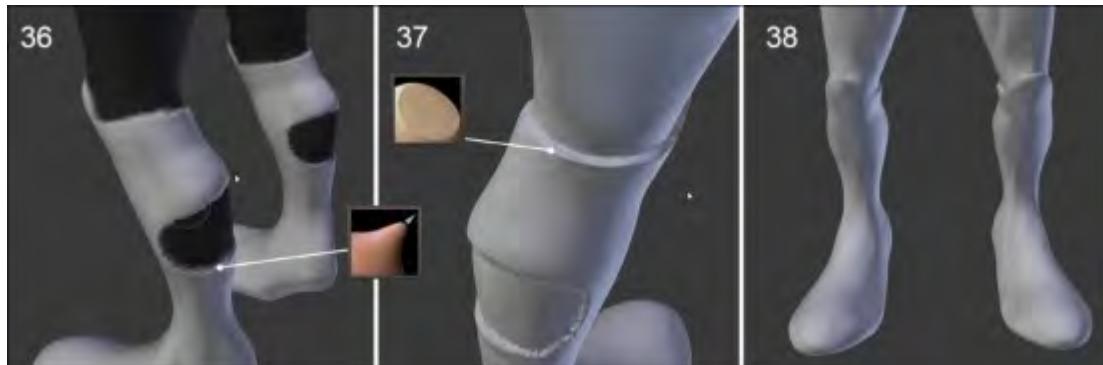
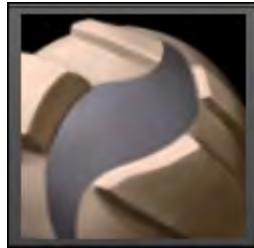
3. Now it is time to use the **Mask** brush that will be only shown without our Matcap activated, that's why you can't see the Matcap in the following screenshots. The boots are up to the knees and have a window over the calves (refer screenshot 34). It is necessary to have enough polygons in order to get the mask contour sharp.
4. In order to highlight the edges of boots, we will reverse the mask with the shortcut *Ctrl + I* (refer to 35).



Note

The Mask brush

This is a quite special brush. It allows us to mask an area of the mesh. It means that this area stays unchanged when any other brush is used as long as it is masked. Thus, we can create shapes that would be impossible to do otherwise. The uses are many. It is very useful for extruding surfaces.



5. With the **Grab** brush, we will pull the polygons at the edge of the masking. Then we slightly raise them (refer to screenshot 36).
6. We increase the level of detail to enhance the edges of the boots. We mark the separation with the **Flatten/Contrast** brush. We need to zoom enough and adjust the brush size (*F*) accordingly.

Note

The Flatten/Contrast brush

In the **Flatten** mode, this brush allows us to smooth over a surface while digging slightly, or otherwise in the **Contrast** mode, it greatly increases the height of the relief. These two very different functions make it an even more interesting brush.



By using the same technique, we can sculpt the collar in the neck area.

The belt

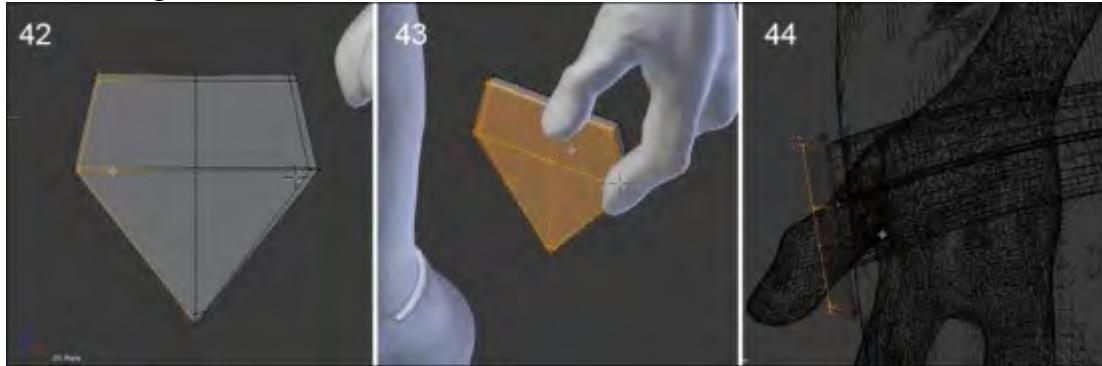
The belt needs to be treated separately because it's not about sculpting. We are going to use more the traditional tools that we saw in the previous chapter. But as you'll see, it is very interesting to mix the different techniques that you've learned in this chapter with polygonal modeling tools. That's why we are going to use the **Grab** brush in order to wrap the belt around his waist.



We will start the modeling of the belt with a primitive circle. After this, we will then place our cursor at the center of the character in the **Object Mode** from the top view in order to add a new circle with 32 vertices.

1. In the **Edit Mode (Tab)**, and with the **Wireframe** option of **Viewport Shading** on, we will adjust the size of the circle by scaling on the Y axis ($S + Y$). We then rotate it a little bit in order to match with the shape of the alien (refer to **39** in the preceding screenshot).
2. We will then extrude the circle to form the height and the thickness of the belt. As we said before, we can now use the **Grab** brush (in the **Sculpt Mode**) in order to stick the belt to the waist. In our case, it's as if we were using the **Proportional Editing** tool (refer to **40** of the preceding screenshot).
3. Then we will go back to the **Edit Mode** in order to add more resolution with the **Loop Cut** tool ($Ctrl + R$). We will also place a loop cut in the middle of the belt on which we will add a little **Bevel** ($Ctrl + B$).

4. In the middle of the bevel, we will add a new edge loop that we will scale along the normals (*Alt + S*) (refer to **41** in the preceding screenshot).
5. We can now switch back to the **Sculpt Mode**, and with the X symmetry option off, we can move the right-hand side down a little with the **Grab** brush.



Now that we've finished the belt, it's now time to add the belt buckle as follows:

1. In the **Object Mode**, we will add a new plane.
2. Then we will go in the **Edit Mode (Tab)** and add a horizontal and a vertical edge loop (*Ctrl + R*).
3. We will then resize these edge loops so that they form a diamond shape (refer to **42** in the preceding screenshot).
4. It's now time to add a **Subdivision Surface** modifier.
5. We will then add some edge loops on both sides in order to maintain the diamond shape.
6. In order to add thickness to the buckle, we will do some extrusions of the whole geometry (*A* and *E*). As always, we will maintain the shape with the Loop Cut tool (*Ctrl + R*).
7. We will also scale the front polygons of the buckle.
8. Finally, we can place our belt buckle at the right place in the **Object Mode**.

There you go! Our little alien is ready for crazy galactic adventures!



A render of the final alien sculpt with Blender Internal Renderer

Summary

In this chapter, you've learned a new modeling technique that is best suited for your organic models. By mixing this method with polygonal modeling techniques you will be able to create awesome characters in a very short time! If you have a powerful computer, you can go into further details such as skin pores and wrinkles using alphas, for instance. For now, the alien character doesn't have a good topology, so we will learn how to create a new topology over the model and extract the details of our sculpting in the next chapter.

Chapter 3. Alien Character – Creating a Proper Topology and Transferring the Sculpt Details

This chapter will be more technical than the previous one. We will see how to create a production-ready character with a nice topology, starting with the sculpture of our little alien. Of course, we can't go through all every possible techniques to reach our goal but you will have a solid understanding of what a good organic topology is. You will also learn how to retrieve the details of a sculpture with a normal map. Furthermore, you will learn how to enhance the look of the alien with an ambient occlusion. These maps could be a good starting point to create a more complex and rich texture later. As you learn more and more tools, you will be able to have more possibilities to express your imagination. What you really need to grasp in this chapter is the logical way of doing a good topology, because each object needs one topology according to your needs. So, let's dive in!

This chapter will cover the following topics:

- Understanding the retopology process
- Using the UV unwrapping tools
- Baking normal maps and Ambient Occlusion
- Displaying the baked maps in the 3D viewport
- Making a good topology

We will now create a retopology of our sculpture by using some of the tools that you've already encountered during the robot toy modeling, and some new tools. But, wait a minute, why are we remodeling our alien if we have already sculpted it?

Why make a retopology?

The main goal of doing a retopology is to have a clean version of the sculpture with a good topology. It means that the mesh geometry needs to follow the shapes of the sculpture by defining proper edge loops. A good topology is also a must-have when you want to animate an object, and it's even more important when you are dealing with organic shapes. The muscles need to correctly bend, so this is why we are following them with edge loops. But, of course, we can't delimit each of the muscles, so we are thinking more about their overall form, treating them as groups, such as the pectorals muscles.

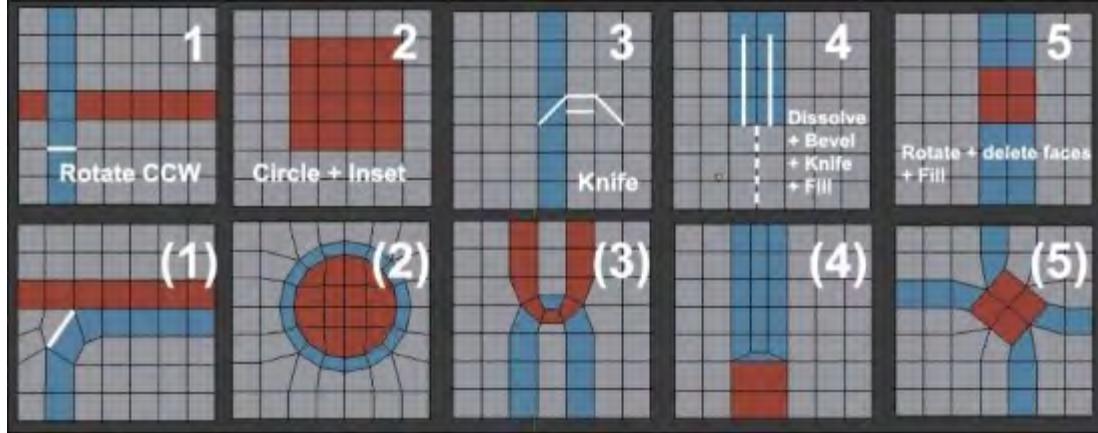
Another goal of a retopology is to have a less dense object. I don't know whether you have already made the mistake of entering into the **Edit Mode** of the alien sculpture, but if this is the case, you have seen a tremendous amount of polygons organized in a fancy way. Technically, the process will be as easy as adding new geometry that snaps to the sculpture. However, it could quickly turn out to be a puzzle if you don't know what you are doing.

Possibilities of arranging polygons

As we have mentioned before, we will be using pretty much all of the poly modeling tools that we've learned previously, such as the grab, rotate, or face creation tool. But what really matters while doing a retopology is the arrangement of the polygons through the loops that defines the shape. In this section,

we will give you some useful techniques in order to help you rework the flow of your topology. Before reading further, we advise you to train yourself on a subdivided plane:

1. Create a new plane (*Shift + A*).
2. Select it in the **Edit Mode** and under the **Specials** menu (*W*), select the **Subdivision** option.
3. Redo step 2 thrice.



Five topology cases you may encounter

We will now go through the five cases presented in the preceding screenshot.

In many cases, you will need to change the direction of a loop. As you can see in the first case of the preceding screenshot, we have two colored face loops. We will rearrange them so that the blue one doesn't cross the red one. Remember that in order to select a face loop, we will use the *Alt + RMB* shortcut.

1. To resolve the first case, we will select the bottom edge of the blue face situated below the cross intersection.

We will then select the **Rotate Edge CCW** (Counter Clock Wise) in the **Edge** menu (*Ctrl + E*).

Now, we can rearrange the polygons to get a nice round corner.

Sometimes, you may want to have a circular shape in your topology. This occurs mainly in hard surface modeling.

2. In order to resolve the second case, we will select the piece of geometry that will be of circular shape and use the Mesh Loop add-on (by pressing *W* and selecting **LoopTools | Circle**) to form a circle. You can also move your vertices one by one if you don't want a perfect circle.

Now, we will maintain the geometry by doing an inset of the selected faces. As you can see, our circle perfectly incorporates the flow of our geometry.

Another situation that you may encounter is when two face loops forming an arc are stuck together. This technique can be useful when defining muscles or articulations. We will later use this for the knee of the alien.

3. For the third case, we will start with the Knife tool by cutting the three edges that form an arc.

As you may have seen, we now have two triangles that we need to resolve in quads. So, we will add a cut in the middle of both of them with the Knife tool or the Loop Cut tool.

You will often need to reduce the number of polygons at some location. For instance, there could be lot of condensed polygons behind the head, so in order to have less of them for the back, we will make a U shape.

4. So, our goal for the fourth case is to make a U shape with blues faces that leaves us with only one face under it:

1. To do this, we will first dissolve all the vertical vertices that are in the middle of, and below, the blue faces (the dashed line in the screenshot).
 2. Now we can do a bevel of the edges that separates the two vertical lines of the face.
 3. We can now use the *J* key to join the two vertices that form the base of the U shape (the one marked in the screenshot).
5. We will now show you one last technique, but you need to keep in mind that there are many other methods that we can't show you here because it would require a whole book! Sometimes you need to rotate some polygons in the geometry, but you don't want stretches after rotating them.

If you look at the last case of the preceding screenshot, the red square of the polygons has been simply rotated with the *R* key.

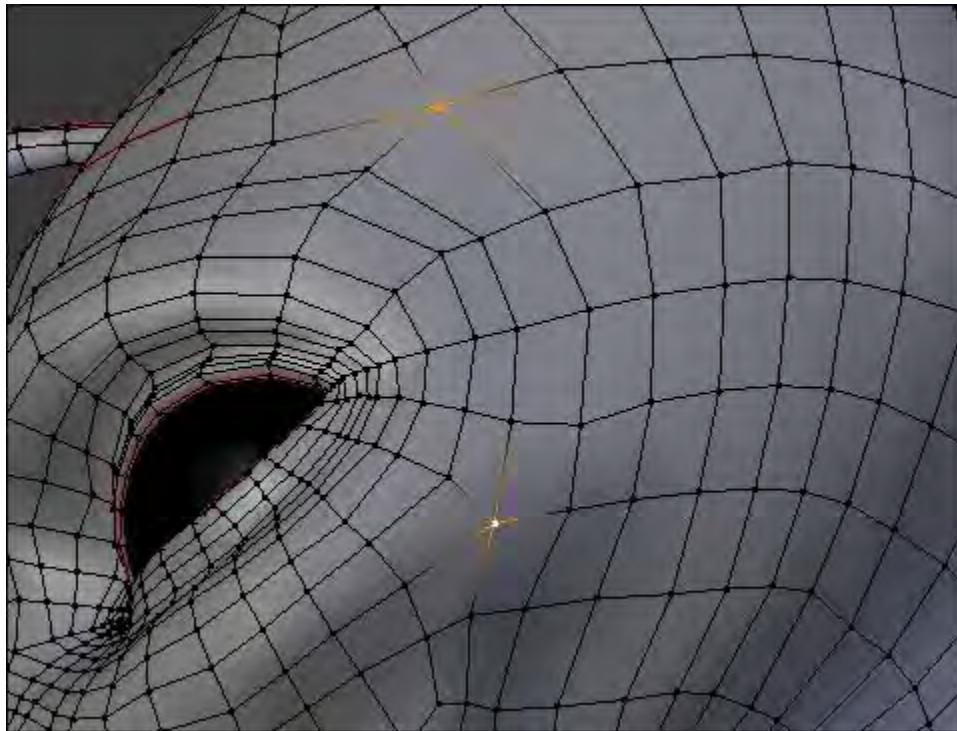
At this point, the geometry around it should be stretched. The best way to solve this is to simply remove the parts that are causing you problems and recreate the geometry in a better way. In our case, we've done this by bridging the square outline back to the rest of the geometry with the *F* key.

Errors to avoid during the creation of retopology

We will now talk a little bit about the main problems that you should avoid while creating a proper topology. The first thing that we have already expressed before is that we need to try to have as few triangles possible. Triangles are bad because they break the face loops, and they can cause some rendering artifacts with the lighting, and the topology could be harder to maintain.

Some people may think that this is not a problem because triangles are used a lot in the video game industry, but usually, the triangles that we see in a game model result from a tessellation made by the game engine after the model was created. They are also needed in cases where there is a lack of performance, especially in mobile games where the amount of geometry needs to be lowered. But the performance of smart phones are doing better with time, so this is not going to be a problem in the future. Sometimes, triangles can be placed at a position that doesn't bend a lot. For instance, the ear of a human rarely deforms, so it's not a problem to place a triangle here. When you add triangles, do it in such a way that it doesn't bother the silhouette of your object.

Another thing that you want to avoid is poles. A pole is a vertex that connects a minimum of four edges. Usually, poles are useful to redirect face loops, so they are usable but they need to be carefully managed. You can create poles if, and only if, you need to change the flow of your face loops and if you are in a place where the geometry won't bend a lot during animation. For instance, on the human face, we can place poles on the cheekbone because we need to redirect the topology. You will encounter this with the alien's head.



An example of two selected poles

Don't be discouraged if you don't get the topology right the first time, come back later and you'll have a clear mind to try to figure out the problem again. Another thing to keep in mind while doing a retopology is that if you want to have a certain loop, create it without waiting because you'll be quickly overwhelmed by polygons, and you could face problems when you need to remove a lot of geometry in order to connect the loop back. Creating a good topology is like solving a puzzle—it always has a solution!

Density of polygons

While doing a retopology, you need to think about the general flow of your topology. If you've done this right, you'll be able to quickly add or remove edge loops in order to increase or decrease the density of the mesh. The more geometry you have, the more you will be able to be precise in the approximation of the sculpted shape. Of course, you need to take into account the eventual constraints that you will have. For instance, if you are creating a mobile game character, it's best to try to reach the minimum number of polygons that gives you the global silhouette. Choosing the right number of polygons is more of a decision that you'll take on the fly.

Making the retopology of the alien character

Now that we have seen a few basic techniques, we are going to see a practical case by working on our little alien character sculpture.

Preparing the environment

Before starting the retopology of our little character that came from distant worlds, we must prepare the working environment for the method that we will use. There are several possibilities in Blender to do a retopology, including with very good add-ons (Retopology MT and Retopology Tools, for example). We will not use them here; instead we will focus mainly on internal Blender tools.

1. We will start by placing the cursor at the center of our sculpted mesh. For this, we will select the mesh and press *Shift + S* to open the snap menu. Then, we will choose the **Cursor to Selected** option.
2. We will then create a plane in the **Object Mode** (pressing *Shift + A* and selecting **Mesh | Plane**) that we will place in front of the eyes. This new object that has been created is going to be the new mesh for our character. We rename it as **Alien_Retopo**.
3. In the **Edit Mode**, we will bisect the plane, delete the vertices on the left-hand side, and then we will add a **Mirror Modifier**. Don't forget to check the **Clipping** option.
4. We need to make a transparent shader that will allow us to work comfortably in order to visualize the mesh. In the **Material** menu, we will check the **New** button. It's also good to name this; in our case, it is **M_Retopo**. This way, we can look through our polygons and check the **Transparency** option with an **Alpha** value of **0.208**.
5. We will then go in the **GLSL** mode in the **Shading** menu of the right panel of the 3D viewport (**N**). This allows us to visualize the new material that has been created.
6. We will also check the **Backface Culling** option just below the **Shading** menu to avoid being too bothered by the rear faces.
7. We switch to the **Texture shading** mode in the corresponding drop-down menu located in the 3D view header.
8. After this, we need to add a **Hemi** light (by pressing *Shift + A* and selecting **Lamp | Hemi**) that is to be placed just above the 3D model in the direction of the ground. We will set its energy value to **1**.
9. We will activate **Xray** in the Properties editor in the **Object** tab under the **Display** subpanel.
10. For our comfort, we need to change the size of the vertices. For this, we will go to **File | User Preferences | Theme | 3DView | Vertex Size**. We will enter a value of **6**.
11. In the **User Preferences** menu, we will check that the **F2 add-on** is activated.
12. In the 3D viewport header bar, we will activate **Snap** (it looks like a little magnet) in **Face Mode** (just on the right-hand side of the snap icon).
13. We still have to activate the buttons located a little more on the right-hand side called as **Project Individual Elements** on the surface of other objects and **Snap onto itself**. Let the mouse hover over a button for a moment so that you can see its tooltip appear.

Now we can start to build the new topology.

The head

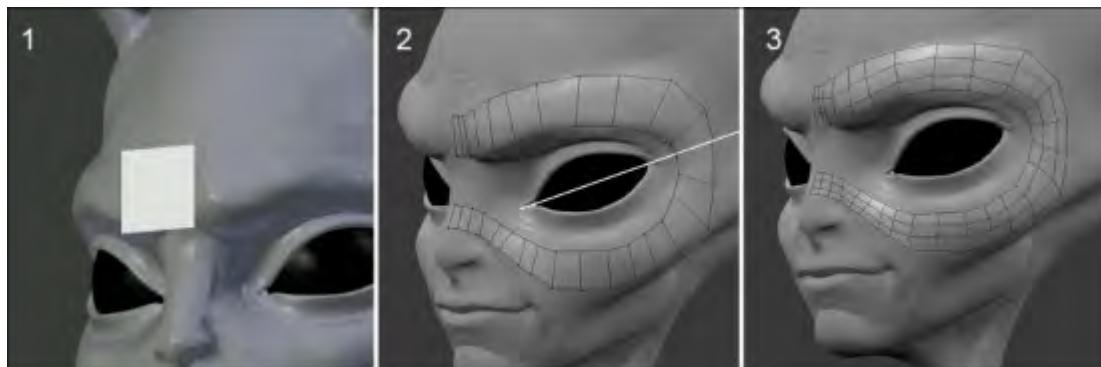
Let's start our retopology with the head of our alien character:

1. In the **Edit Mode**, we will snap our first polygon with the Grab tool (**G**).
2. We will then begin to form the first face loop around the eyes. We will select the edge on the right-hand side of the plane and pressing **Ctrl** and a left-click, we will create new polygons in sequence on the surface of the sculpted mesh. There is no need to be very accurate while tracing the first face loop. We will be able to reposition the vertices and adjust the number of polygons thereafter. So, we will make the first face loop, which passes through the eyebrow, the cheek, and joins the symmetry axis right at the nose. At the corner of the eye, we need to be careful while placing an edge that extends the diagonal direction of the eye (refer step **2** in the following screenshot). This face loop is called the **mask**.

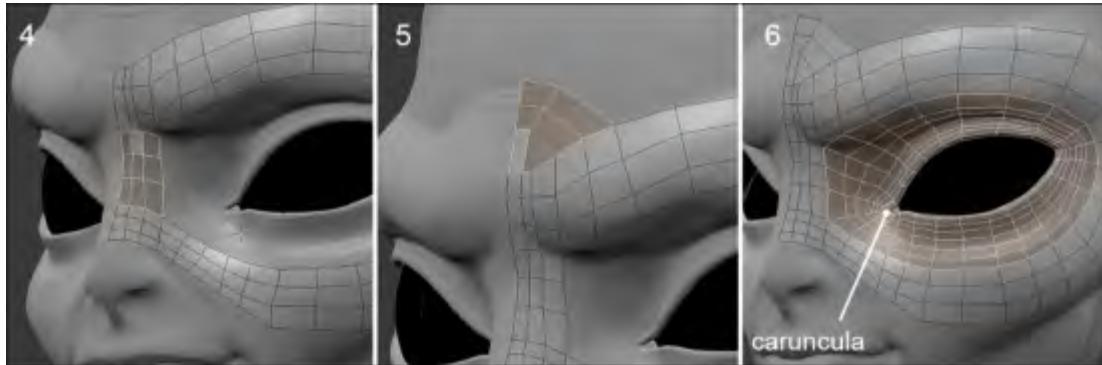
Note

Pair number of polygons

Be careful to always have a pair number of polygons whenever you form a face loop. This will allow you to connect the polygons more easily. If in some cases you only have an odd number, you can try to delete an edge loop and replace the polygons around (with **G**) it.



3. We will then add two edge loops (**Ctrl + R**). Try to equalize as much as possible the size and the distribution of the polygons. We will reposition the vertices to have a rounded shape (refer to step **3** in the preceding screenshot). For our selected components snap on to the sculpture, we have to move them with the Grab tool (**G**). Remember to do this often, but only with the components that directly face your point of view.
4. We will add an edge loop to the nose in order to have enough polygons to better define the outline shape. We will have a 28-face face loop.
5. We will join the nose with the new polygons (refer to step **4** in the following screenshot).
6. To save time, you can start making a shape with a few big polygons to cover a large area that you want to work on. You can then define the number of cuts that you need and snap the topology, rather than make many little polygons one after the other.



Try to follow the shape of the sculpted mesh as much as possible.

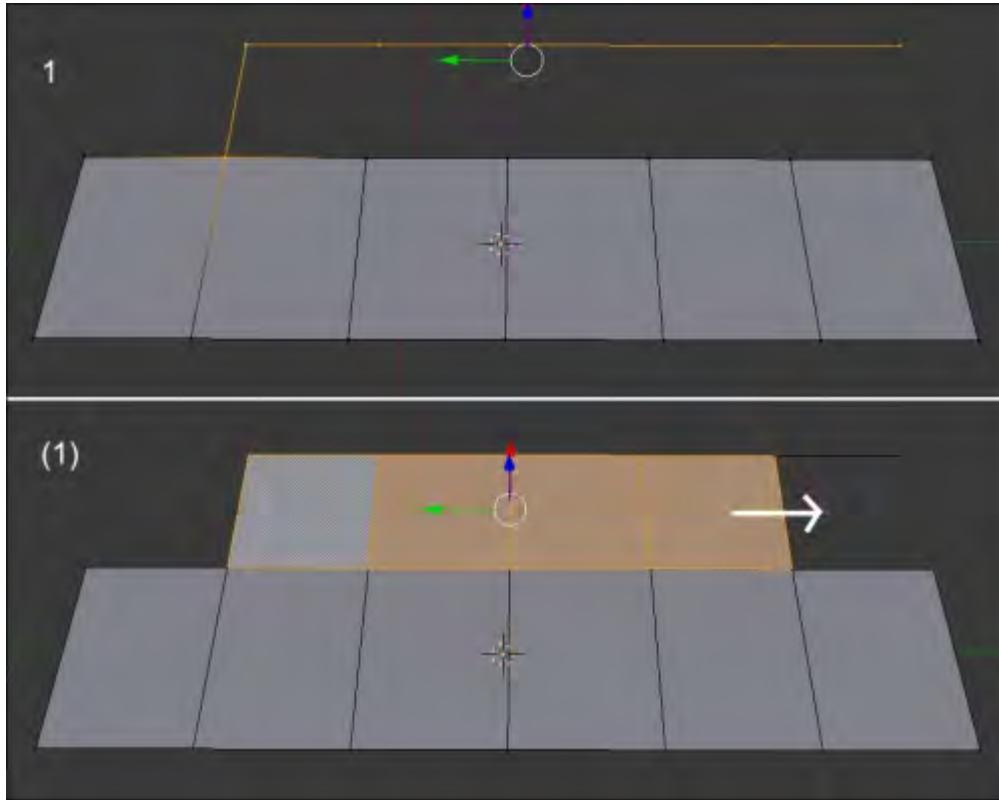
Note

F2 add-on

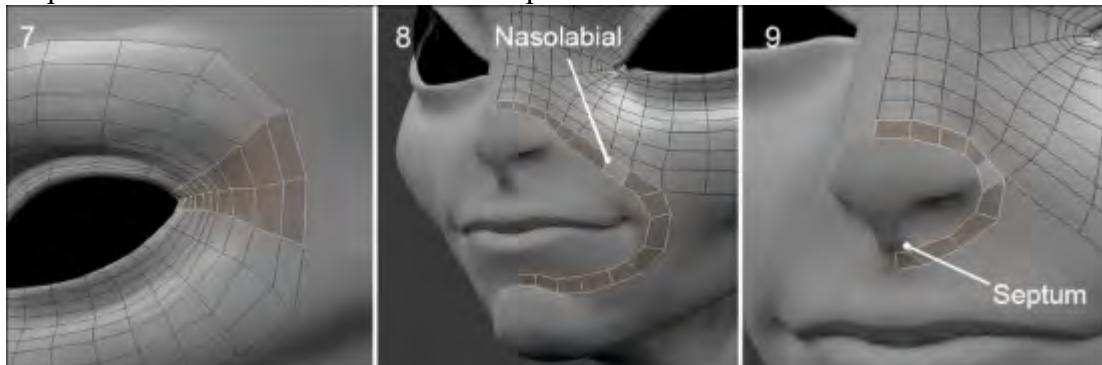
We mentioned this while preparing the environment. *F2* is an essential add-on in Blender that allows us to quickly create faces.

Between the two rows of the edges, the *F2* add-on can generate the missing faces. You can also generate a face by selecting one vertex on the intersection of two faces by pressing *F*. Repeat the process to quickly make a face loop, and remember to weld them (select the polygons, and then press *W* and select **Remove doubles**).

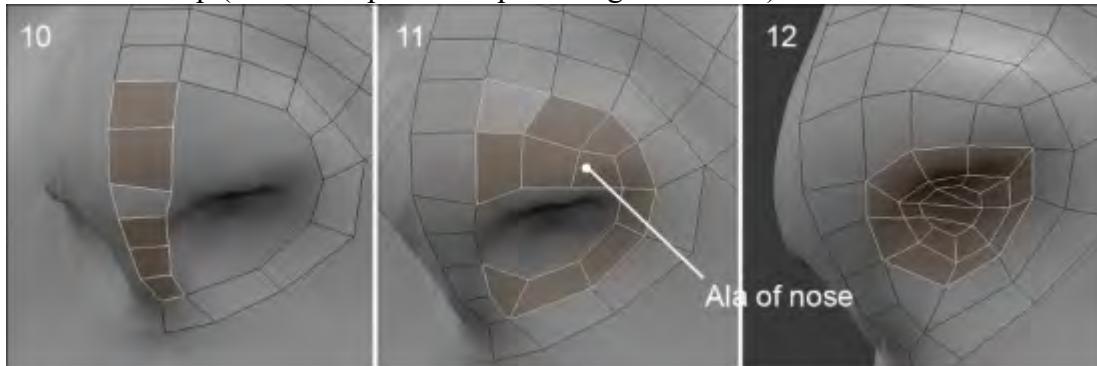
You can also generate the face loops in the edge mode. You need to select a starting edge (the one to the left and perpendicular to the *x* axis according to the following screenshot) by pressing *F*. The sense of creation of a face can be determined by the placement of the mouse if the starting edge is at the center of the face loop that is to be created.



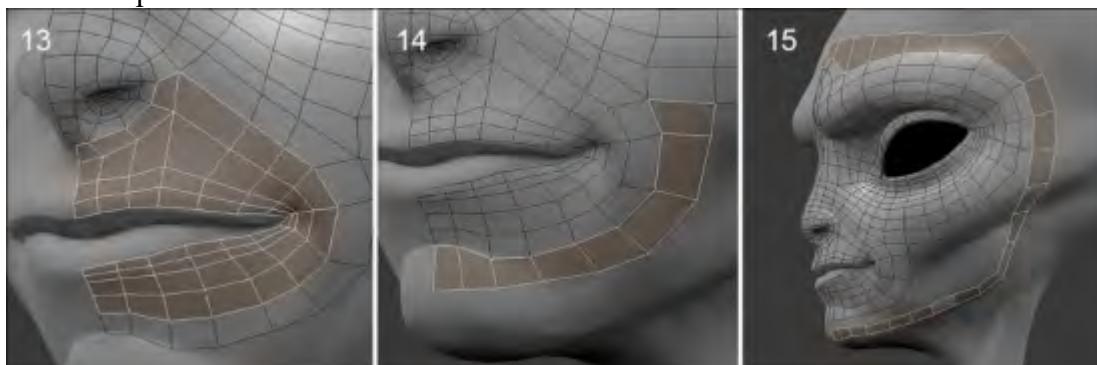
7. We can see that we have a pole. We will avoid making it too visible, so we will place it under the eyebrow.
8. We complete the topology of the upper nose with a polygon reduction on the forehead (refer to step 5 in the preceding screenshot).
9. Now, we will select the edge loop that bypasses the eye, and we do an extrusion by changing the scale (*E* and *S*). We will then place the vertices on the outline of the eye (*G*). We will have a 24-face face loop (refer to step 6 of the preceding screenshot).
10. We will add two edge loops (*Ctrl + R*) around the eye that we will reposition correctly (*G*). A first edge loop takes the shape of the eye, marking the fold with the eyebrow. A second edge loop makes the link between the two shapes.



11. At the left outer corner of the eye, we will tighten the vertices. We will take care to have a conical shape that spreads outwards (refer to step 7 in the preceding screenshot).
12. Now, we create a face loop from the nose to the chin. It is called the nasolabial loop. We will add the polygons using *F2* (by selecting a corner vertex and pressing *F*). Just like the eye, we extrude an edge and form a face loop from it that follows the circular shape of the mouth. We will have a 16-face face-loop (refer to step 8 of the preceding screenshot).
13. We will add a face loop that comes around the nose. This allows us to define the nose. Remember to slightly bring the edges at the bottom to the septum of the nose. We will have a 9-face face loop (refer to step 9 of the preceding screenshot).

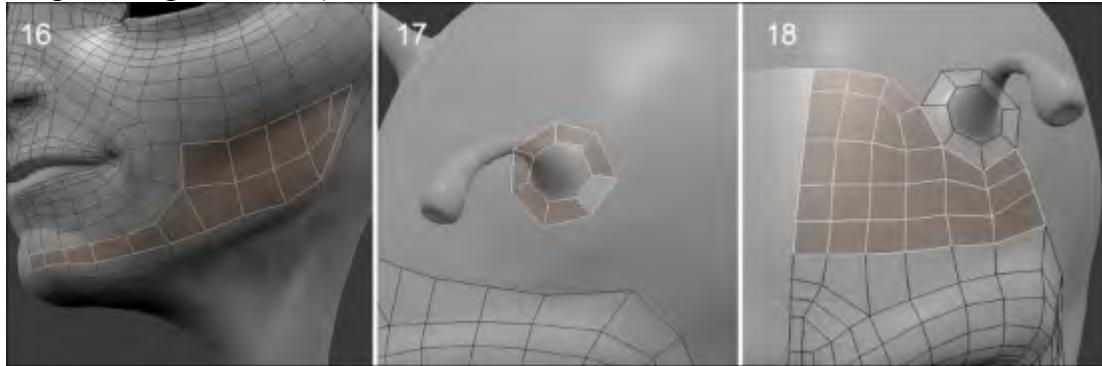


14. We will create the topology that forms the top and bottom of the nose. We will have a 6-face face loop (refer to step 10 of the preceding screenshot).
15. Once the nose shape is outlined, we will begin the ala of the nose. We will extrude a face loop that starts from the top of the nose down to the bottom of the nose, avoiding the nostril. This arrangement allows us to better create the shape of the nostril loop (refer to step 11 of the preceding screenshot).
16. We will extrude the edge loop of the nostril with a scale transformation (*E* and *S*). We will then close the hole by connecting the edges (refer to step 12 of the preceding screenshot). If you don't have enough edges to make four-sided polygons, you can use a triangle. It must be as hidden as possible.

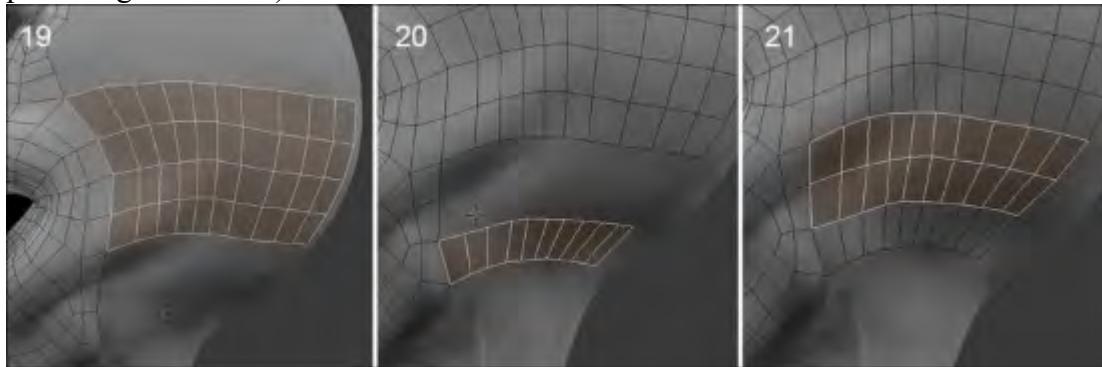


17. We will start from the corner of the nose to make a face loop that goes under the mouth.
18. We will then close this part by following the shape of the lips. Take care to slightly pinch the corner of the mouth. The topology must be as symmetrical as possible between the top and

- bottom of the mouth in order to connect them later. We will have a 14-face mirrored face loop (refer to step **13** of the preceding screenshot).
19. We will continue around the mouth by adding a polygon strip above the chin (see **14** and **15** in the preceding screenshot).



20. Now, we will make the face loop of the jaw, which ascends to make the contour of the face. We have a 22-face mirrored face loop (refer to step **15** of the preceding screenshot). We need to align the faces correctly for an easy connection and close the area of the jaw (refer to step **16** of the preceding screenshot).
21. We will make the contour of the antennas with an extruded vertex on the top of the forehead in a circular manner (refer to step **17** of the preceding screenshot).
22. We will then continue to form the polygons of the forehead, in line with what was done, by making a reduction in the number of faces, avoiding the antennas (refer to step **18** of the preceding screenshot).

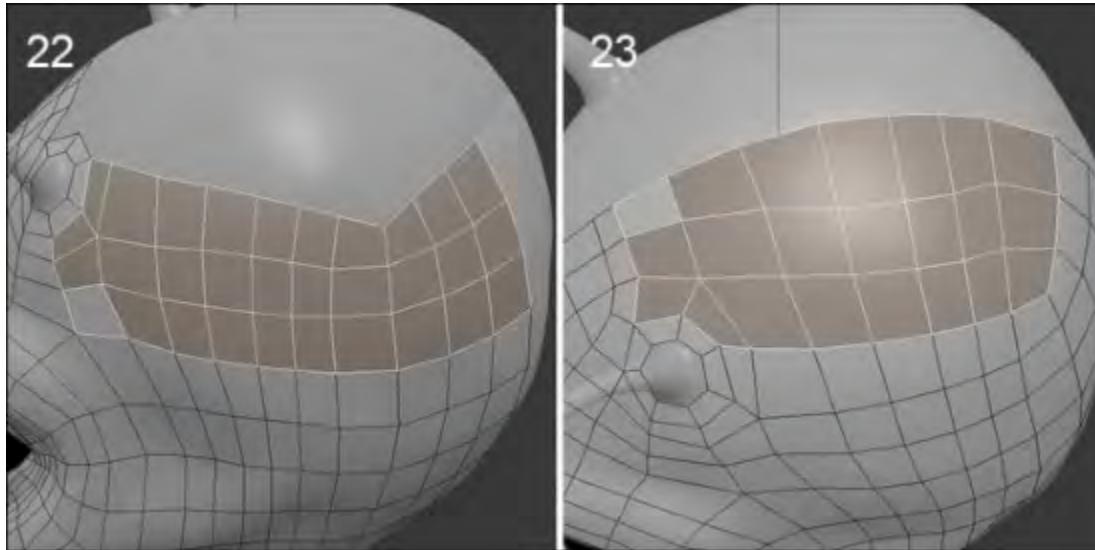


23. We will select three edges on the temples and create a polygon strip that follows the back of the head. We need enough faces to have a nice rounded.
24. We will then select the edge located at the jaw, and we will continue to make the topology of the back of the head (refer to step **20** of the preceding screenshot).
25. Now that the face strips are facing each other, we can fill the missing topology with the Fill and Loop Cut tools. (Refer to step **21** of the preceding screenshot.)
26. We will continue to make the faces on the back of the head in the same way. We will form an angle to close the top of the head later more easily. (Refer to step **22** of the following screenshot.)

Note

The Smooth option

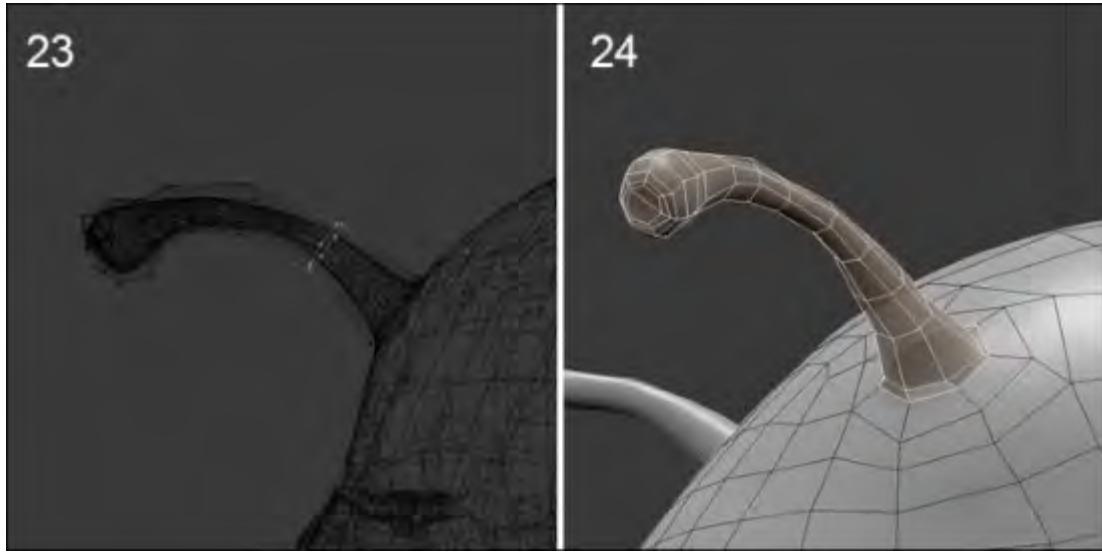
In order to have a topology as homogeneous as possible, we can use the **Smooth** option of the **Specials** menu (**W**). This allows us to relax the polygons. We first need to select the faces we want to modify (**C**). Be careful as you have to snap them after the process (**G**), so select only the visible faces in the view.



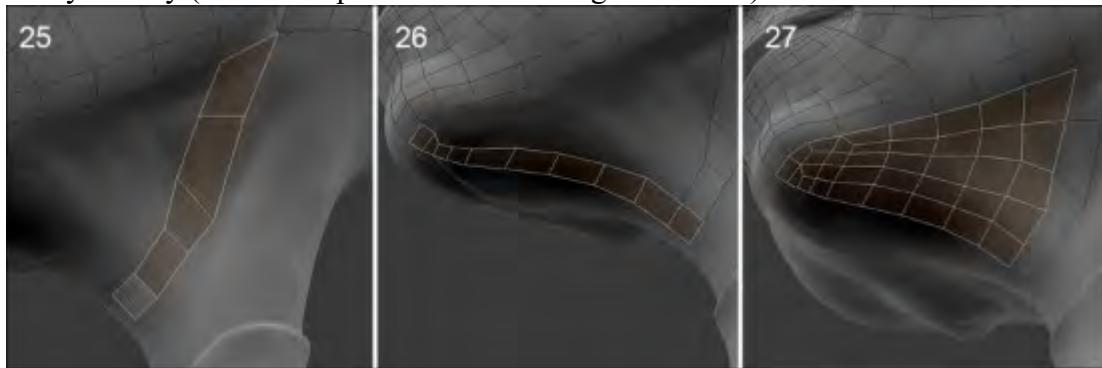
We now need to close the topology of the top of the head. We must pay attention to respect the number of edge loops created previously. You can apply the smooth option to quickly relax some misaligned faces.

Now we are going to make the antennas that require another technique:

1. We will duplicate the edge loop situated at the base of the antenna already created to make a new object (by pressing **P** and select **Selection**).
2. We need to deactivate the **Snap** option and the **Project Individual Elements** option.
3. We can extrude this new object following the shape of the antenna. The polygons must completely cover the surface of the sculpted mesh. You can set **Wireframe Shading Mode** in order to view your geometry better.
4. Once we get the desired topology, we add **Shrinkwrap** modifier and select the sculpted mesh in the **Target** parameter, called **Body**. This modifier allows us to snap a mesh on another.
5. We must apply the modifier by clicking on **Apply** button, and then we will join it back to our other piece of geometry. For this, we select the mesh of the antenna, then the mesh of the head, and we press **Ctrl + J**.
6. As we want to snap our components on the sculpture again, we will restore the **Snap** and **Project Individual Elements** options.
7. We can add a few edge loops to the antenna in order to get a good enough shape.
8. We will connect the faces in the back of the head in continuity with the topology already done.



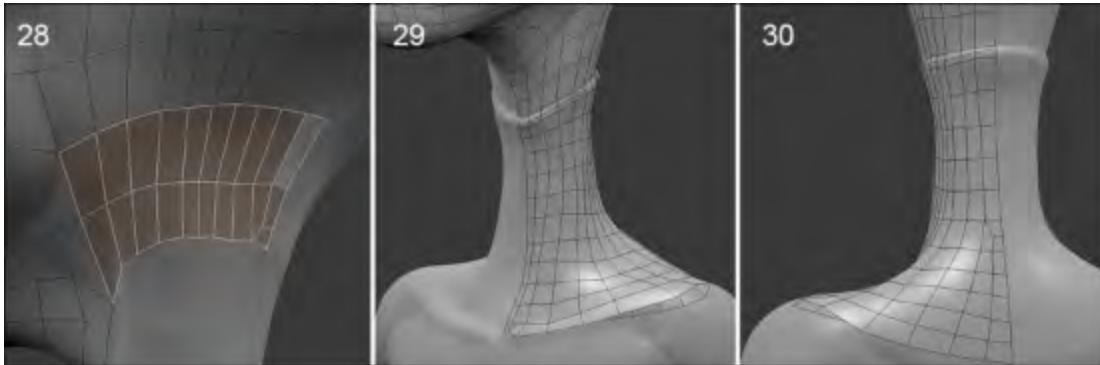
9. From the angle of the jaw, we will create a face loop that ends at the symmetry axis on same level of the glottis (refer to step **25** of the following image).
10. We will then connect the face loop to the chin by adding enough edges. We need to connect the entire underside of the jaw. We have a 10-face face strip (refer to step **26** of the following screenshot).
11. This area forms a triangular shape. We need to make loops to reduce the number of polygons around the chin. We can easily reduce the polygon flow by redirecting the loops on the axis of the symmetry (refer to step **27** of the following screenshot).



12. To better visualize your polygons, you can hide the ones that you do not want to see. You can use the *H* shortcut. Press *Alt + H* to make them reappear.

The neck and the torso

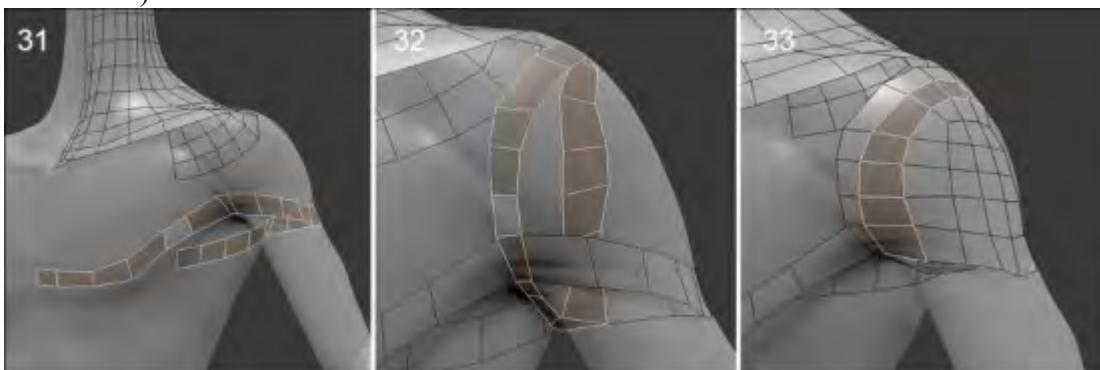
We will continue this retopology with the neck.



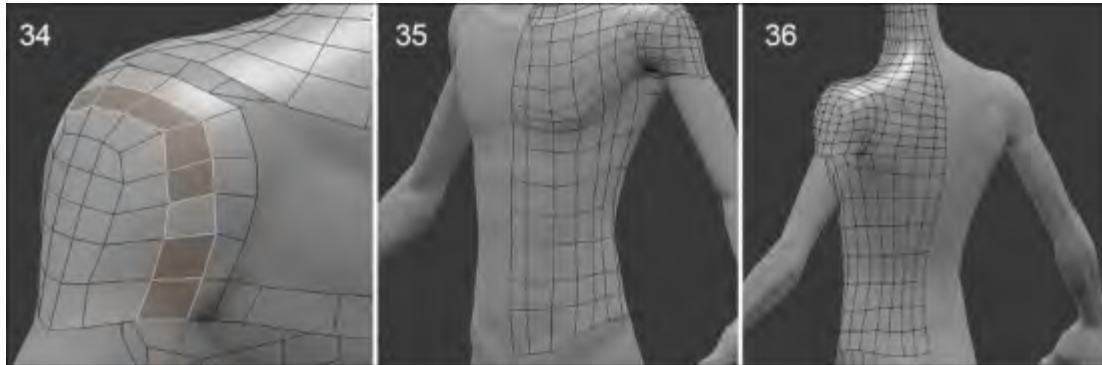
1. We will make a small polygon reduction at the back of the neck in order to lighten the density a little bit. We only need to progressively extrude the polygons along the neck. We will then align the polygons on the collar. We slightly pinch the edges to restore the volume of the collar outline.

The challenge is to properly harmonize the polygon flow along the front and the rear.

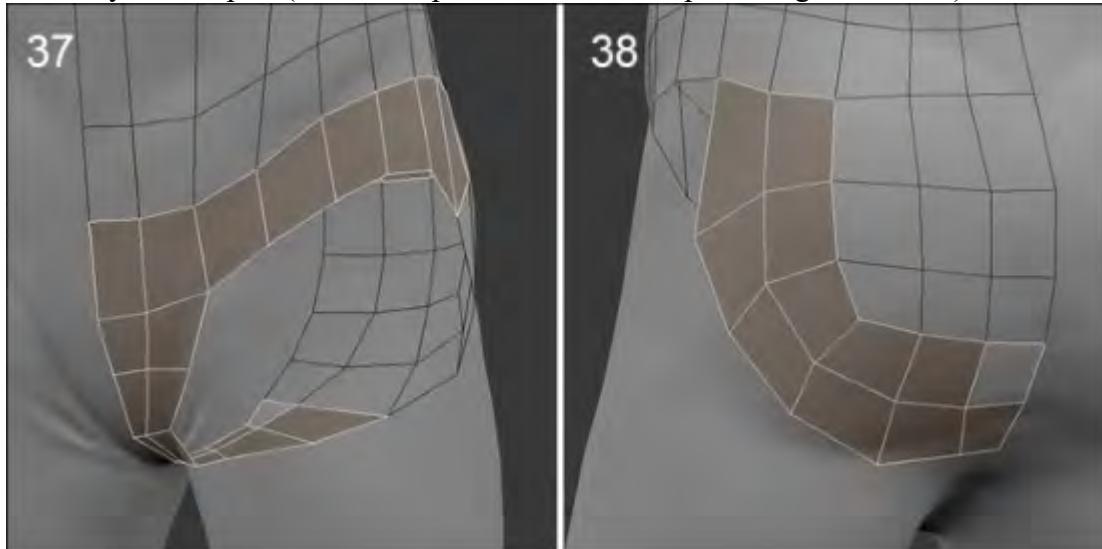
2. We will continue to extrude to the clavicles. Using loops, we will add a polygon row in the middle of the nape, and we will reduce one on the sternum (refer step **28** of the preceding screenshot).



3. There is an important face loop to place in case we want to animate the arms later. It follows the lower pectorals and shoulders muscles. We will have a 20-face mirrored face loop (refer to step 31 of the preceding screenshot).
4. We will make another important face loop that goes vertically around the shoulder and passes under the arm. We will have a 16-face mirrored face loop (refer to step 32 of the preceding screenshot).



5. Therefore, we will connect the polygons to form the chest and the upper back. On the shoulder, a new face loop links the polygons that intersect vertically and horizontally (refer to step **33** of the preceding screenshot).
6. We will extend the topology straight to the pelvis in a cylindrical manner. There is no particular difficulty on this part (refer to steps **35** and **36** of the preceding screenshot).



7. We end the hip with a strip of polygon that joins the axis of symmetry and we will create the crotch by connecting the buttocks polygons. Don't add too many edge loops, just what is needed. We will have a 6-face strip to make the connection (refer to step **37** of the preceding screenshot).
8. Now, we will make a face loop that follows the shape of the buttocks. This face loop allows us to have a good deformation when animating the legs. To connect the vertical polygons that come from the back, we will make a second face loop right in this area (refer to step **38** of the preceding screenshot).

The arms and the hands

It is time to make the topology of the arms. Considering they are quite thin and not very muscular, we are going to use the same technique that we used for the antennas.



1. This time we will add a 10-face cylinder (*Shift + A*) positioned around the arm. Be careful to match the number of faces on the shoulder (refer to step **39** of the preceding screenshot).
2. To facilitate the visualization of two meshes, we will parent our semitransparent material by first selecting our retopology, and then selecting our new cylinder by pressing *Ctrl + L* and selecting **Materials**.
3. We will add a **Shrinkwrap** modifier with the sculpted mesh as a target.
4. We need enough polygons to get a sufficiently smooth shape. We will bring, little more loops on the elbow in case we need to animate the arm later so that it will bend appropriately.
5. As for the antennas, we will apply the **Shrinkwrap** modifier in order to freeze the new shape. We will then join the arm to the rest of the body (*Ctrl + J*).
6. Now that we have only one mesh, let's connect the arm to the shoulder by selecting the two opening loops and by bridging them together (by pressing *W* and selecting **Bridge Edge Loop**). Since we have the same number of vertices on the two sides, Bridge must work fine. We will then need to adjust the topology by adding some edge loops here and there. Don't forget to snap back the vertices on the sculpture (refer to step **40** in the preceding screenshot).

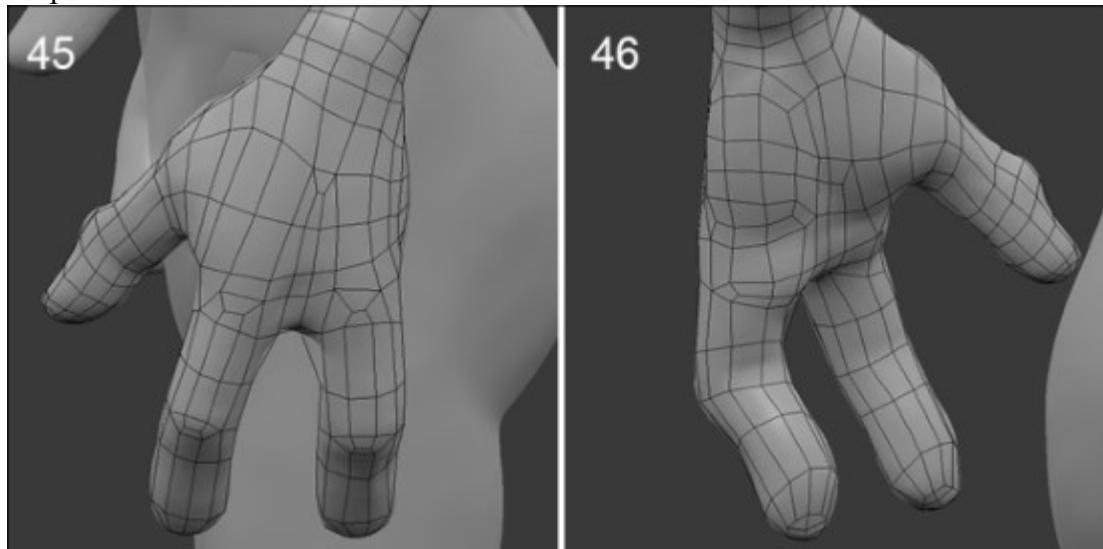
Let's start the hand. It is a particularly delicate part of the anatomy to do:

1. We will start by doing the fingers using an extruded circle again and the **Shrinkwrap** modifier for each finger. This is the same technique used for the antennas and the arm. This time, we need to extrude a 10-edge circle for each finger. This number is important because we need to have enough polygons for the hand. Pay attention to the orientation of the thumb (refer to step **41** in the preceding screenshot).
2. Once this is done, each finger is closed by joining six four-sided polygons. Our character has only two phalanges by the finger, so be sure to place three cuts around each of them if you want to properly animate the fingers later.
3. Then, we will make some of the important face loops of the palm. There is a 10-face face loop that passes around the thumb (the thenar muscles). There is a 14-face face-loop on the opposite

side of the hand near the little finger (the hypotenar eminence) (refer to step 42 in the following screenshot).

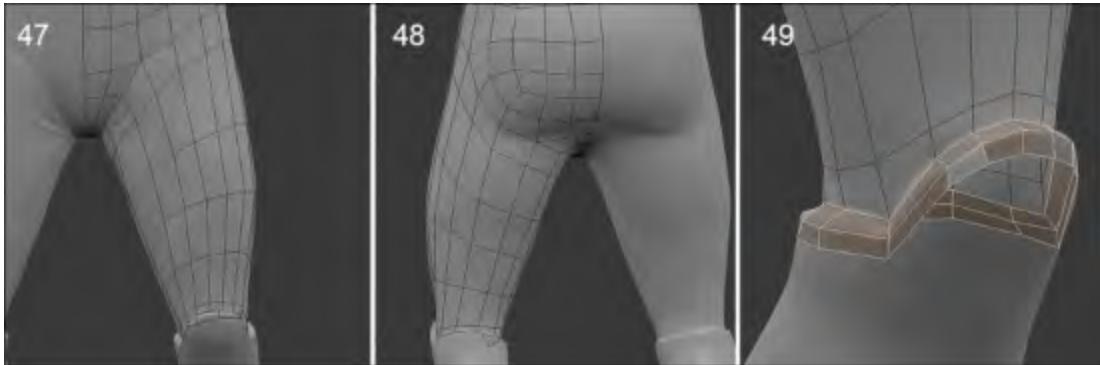


4. We will create a face strip in order to accentuate the basis of the fingers. It is also important to leave a space between each finger (refer to step 43 in the preceding screenshot).
5. Now, all the face loops have to be connected. We have the needed density to only use four-sided polygons without too much difficulty. Train yourself to fill holes with the number of polygons that you have. In this case, we can't change the number of polygons that wraps around each finger (ten in our case) (refer to step 44 in the preceding screenshot).
6. We will add a few more horizontal edge loops around the phalanges in order to have a smooth shape.

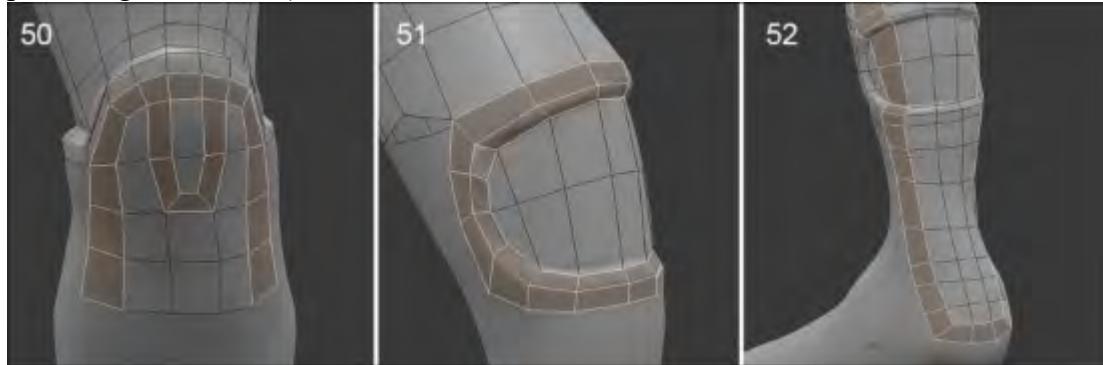


The legs

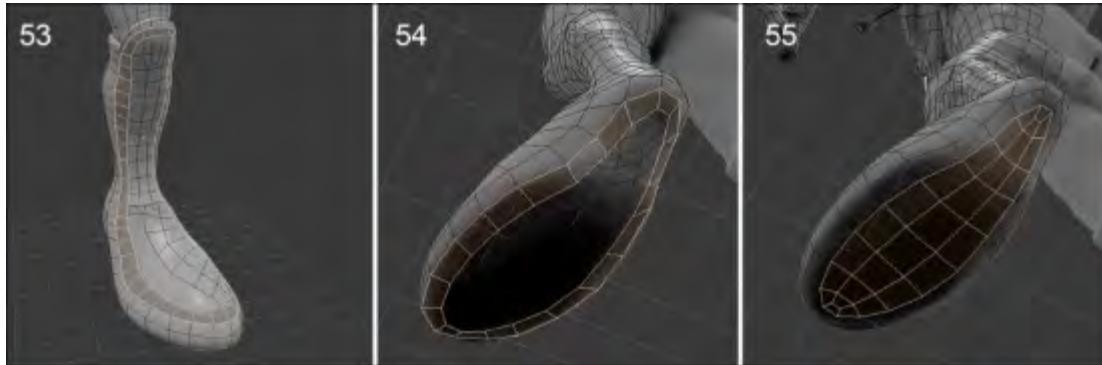
We are going to finish this retopology with the legs. Let's begin with the thigh:



1. We will select the edge that makes the outline of the leg and extrude it to the knee (*E*). As before, we will add a few edge loops (*Ctrl + R*), and then adjust and snap the topology on the surface of the sculpted mesh. We could have used the technique used for the antennas and arms with a cylinder but the thigh is wide enough and less complex (refer to steps **47** and **48** in the preceding screenshots).



2. We will add an edge loop a little tighter near the knee.
3. We can see that the boots are just above the knees, so we will make two edge loops that stick to the relief (refer to step **49** in the preceding screenshot).
4. On the front of the knee, there is a loop that will reduce the number of polygons on the shin. It goes up and down along the thigh (refer to step **50** in the preceding screenshot).
5. There is another face loop to be created that will follow the rounded shape of the boot at the knee. It goes up toward the thigh (refer to step **51** in the preceding screenshot).
6. We must create the missing polygons on the top of the boot in line with what was done until the opening of the calf.
7. This opening of the boot requires two face loops that allow us to maintain the hole. We must pay attention that it is easily connectable between the top and bottom. This looks like the circular topology case that we had analyzed before. We have two 18-face mirrored face loops.
8. We will continue the retopology by creating a loop that goes under the heel and connects with the upper leg (refer to step **52** in the preceding screenshot).



9. Let's go back to the front of the thigh and continue to form a strip of polygon that goes around the foot and forms a long face loop. It goes through the knee. This allows us to control the polygon flow by adapting it to the shape. We just need to keep the same number of polygons on both sides of that face loop in order to connect them properly (refer to step **53** in the preceding screenshot).
10. A final important loop remains to be made. This is located under the foot. It follows the outline of the plantar arch with a 20-face face loop aligned so that we can easily connect it to the rest of the foot (refer to step **54** of the preceding screenshot).
11. We still have to connect the inside of this underfoot face loop. To move from one row to three rows of polygons along the length of the foot, we must create a face loop by cutting the faces at the ends of the feet (refer to step **55** of the preceding screenshot).

The retopology of this little character is now over. We complete a mesh that offers many possibilities, such as creating textures or doing animations. All of this couldn't be envisaged with a very high polygon density and sculpted 3D model. Having a few polygons will facilitate the process. The important thing now is to get the small details that we had sculpted back on our mesh that we can describe as a **low poly mesh**. For this, we must discover a process called UV's unwrapping.



A presentation of each important face-loops of the alien

Unwrapping UVs

Now that we have a clean topology, we can learn more about the UV unwrapping process that we introduced in the first chapter. We have to do this in order to project the sculpted details on our clean mesh. Before starting, let's see what UVs are.

Understanding UVs

The goal of the UV unwrapping process is to flatten the 3D mesh in a 2D space in order to project textures (2D images) on it. To understand the process better, imagine that we are going to remove the skin of our alien in order to flatten it (I know that the metaphor is a little gory, but stay with us, there won't be any blood). If we have to do this, we would need to detach the skin by cutting along the imaginary seams and then flattening it down.

Another way to better understand UV unwrapping is to think about how clothes are made and take the steps in the reverse order. For instance, at the beginning, a shirt is a flat piece of tissue where all the seams are marked down. Then, the different cut pieces are attached together in order to form the volume of the shirt, like the sleeves. So, if we were unfolding a cloth along its seams, we will tell Blender where the seams are placed on our 3D mesh. After this, the ones that have been marked and the model that has been flattened down will correspond to each vertex, edge, or face between the 3D model and the flattened version of it; this is actually a 2D representation of the geometry.

In general, we call the coordinate axes of a 2D space the *x* and *y* axes, but in this case they are named as U and V. Hence, the name **UV unwrapping process**. Another really important thing to note is that we often want to avoid any overlap of geometry in the UV space. This is because when we use UVs of an object in order to project 2D textures on it, we seldom want to have the same texture information twice on the 3D model. We will also need to optimize the UVs so that they don't generate strange distortions.

Lastly, similar to the shirt example, our UVs will be usually separated in different parts called islands. For instance, we will disconnect the head of our alien from its hands or other limbs. These islands need to have a proportional scale to the geometry data they represent (that is, the head is bigger than the hands, so it needs to be bigger in the UV space).

The placement of the seams

It's now time to unwrap the UVs of our alien character. So, as we stated before, we first need to tell Blender where the seams are going to be:

1. We will first select all the edges that start from the lower back of the neck to the middle of the forehead (the ones that are on the symmetry axis). We can also select four perpendicular connected edges together at the end of the previous group of edges selected. It will look like an inverted T shape from the front view.
2. Now, in order to mark our selection as seams, we will go to the **Edges** menu (*Ctrl + E*) and select the **Mark seam** option.
3. Always try to think as if you were using a cutter, so the mesh will be disconnected in the UV space along the marked edges. Also, note that the seams are in red.
4. Now, we will select the top collar edge loop (press *Alt* and the RMB) where it meets the previously marked edge and mark it as a new seam again. This will completely separate the head and neck portion into a new island.
5. In order to have less deformation under the chin, we will select edges starting from the top of the collar (where we placed the seam) to the chin along the axis of symmetry. We will add five perpendicular edges to our selections and mark our selection as new seams. We've done this in the same way as the forehead.

6. In order to separate the antenna in its own island, we will select the circle loop of its base and mark it as a seam. We will also have to mark a vertical line of edges that start from the previously marked seam and end at the top of the antenna.
7. Then, we will end this seam by following the circular cap of the top of the antenna without closing it completely, leaving an edge unmarked. When you are marking the seams, try to think of an approximation of the shape that you know in the real life. For instance, in this case, the antenna looks like a candy wrapper.
8. To end with the head, we will select an edge loop around the eye and mark it as a seam. If you want to remove a seam, you can simply select a seam, and in the **Edges** menu (*Ctrl + E*), you can select the **Clear Seam** option. We've now completed the UVs of the head, but we will finish the body before looking at them unwrapped.



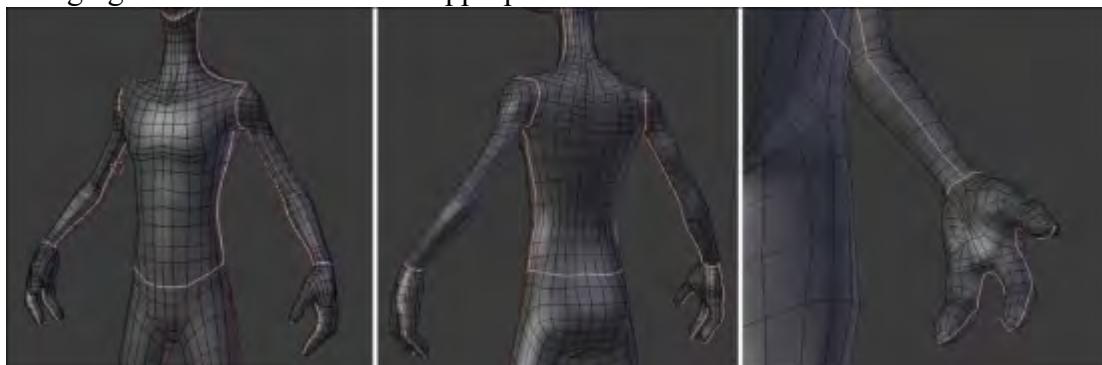
The head seams

9. Now, we will select a vertical edge loop around the shoulder and mark it as a seam. This is going to represent the separation between the arm UVs and the torso UVs.
10. In order to clearly see that the arm UVs are disconnected from the rest of the body, you can go in the **Face** mode, hover your mouse on the arm and press the *L* key (selection of the linked parts).
11. We will then separate the torso UVs from the legs by marking an edge loop situated under the belt.
12. It's best to consider the torso UVs as two separate islands, the front and the back. So, we will split this by marking the edges that connect the shoulder's vertical seam to the seam of the collar. These edges follow the angle created by the neck and the shoulder.
13. After this, we will mark a line of edges that start under the arm and go right to the belt seam. As you can see, using the linked parts method presented before, the torso is now in two parts that are delimited by the collar, belt, and shoulder seams (of course, we always have the mirror modifier turned on, so we can only see half of the torso's linked parts)
14. We will then mark an edge loop that follows the top boot outline.
15. As we did with the torso, we will split the legs UVs into two islands. To do this, we simply mark the edges that start from the belt seam (following the same direction of the seam on the side of the torso) and end at the boot seam.
16. For now, the legs are still in one part, so we will add a new seam starting from the boot seam and ending below the pelvis bone. Now, our legs UVs are split into two islands.

17. It's now time to mark the seams of the boots. To do this, we will go to the bottom view (*Ctrl + 7* numpad key) and mark a loop that follows the footprint of the boot. We will also mark the loop that follows the hole of the boot on the calf muscles.
18. We will then mark the vertical edges of the back that connect the footprint seam to the hole seam and the ones that connect the hole seam to the top boot outline.
19. Lastly, we will mark the seams of the hand and the arm. We will mark the wrist edge loop in order to disconnect its UVs from the arm. Now, we will mark a continuous line of edges that start from the tip of the thumb and follow the silhouette of the hand by connecting to the wrist seam. As you can see, we didn't split the top of the thumb so that the interior palm and the back of the hand share the same UV Island. Then, we will mark a continuous line of edges from the wrist to the shoulder in order to do the UVs of the arm.

Congratulations! You've finished marking all the seams, and we are now ready to unwrap the alien. In order to do this, we must first apply our **Mirror** modifier. As you can see, all the seams are now mirrored to the other side; what a time saver!

1. Now it's time to unwrap the alien. Let's see how this is done. We will first select all its geometry by pressing *A*.
2. Now, we will press the *U* key to get the **UV Mapping** menu and select the **Unwrap** option. Unwrap is now done!
3. To see the UVs, we will open a new **UV/Image Editor** by splitting our 3D view into two and changing the new window to the appropriate editor.



The upper body seams

The seams follow the volume of each part of the character.



The lower body seams

As you can see, in the **Edit Mode**, all our 3D geometry is being flattened down in the UV space. We are now ready to reorganize each island. Note that if you still have problems while understanding the relation between the 3D space and the UV space (don't be disappointed if this is the case, UVs are hard to understand in the beginning), you can use the linked method presented before to see how each part is represented in the UV space.

The placement and adjustment of the islands

We are now going to adjust each island in the UV space. As you can see, all our islands are bounded to a square. Everything that is on the outside won't be used, so all the islands need to be tightly packed in it. Moreover, we'll have to check whether there are any important deformations on our 3D mesh. If this is the case, it means that each texture will be stretched whether we want it or not.

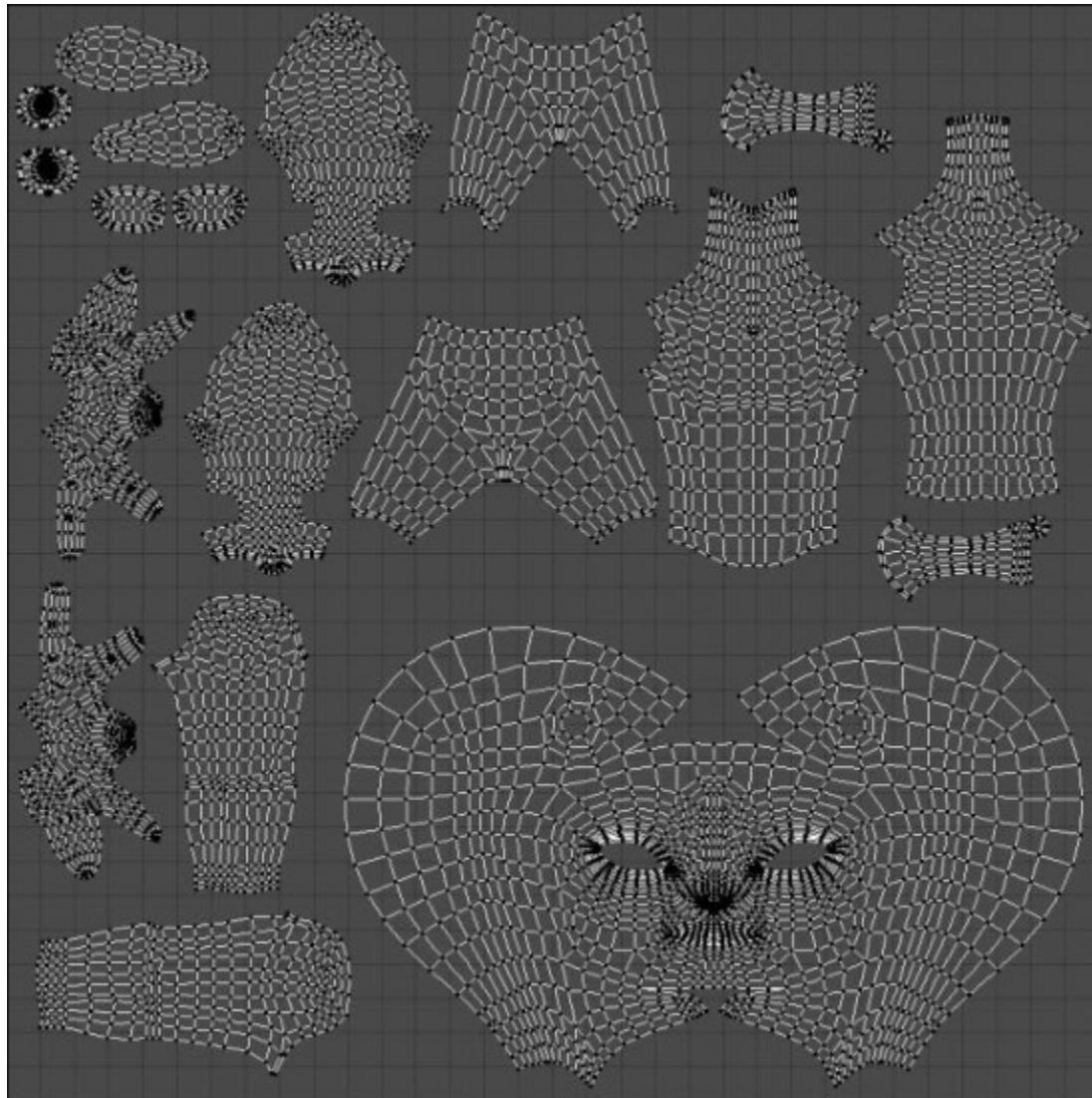
1. In order to adjust the different island placements, in the **UV/Image Editor**, we will first need to enter the **Island UV selecting** mode.

This allows us to select every island and place it with the grab (*G*) tool, scale (*S*) or rotate it (*R*) in the **UV/Image Editor**.

2. When we try to place the islands correctly on the UV space, we need to be sure that they don't go out of the square bounds (represented by a grid).
3. You can also use the **Pack Islands** tool located under the **UVs** menu, in order to have this automatically done. But it's always best to do it by hand.
4. Another thing to remember is to have the island proportional in scale to what they represent on the mesh. This was automatically done when you first unwrapped your mesh, but if you scale the islands by hand, be aware of this. However, you can sometimes counter this rule when you need more texture details on a specific location. For instance, the head is one of the most important part of the alien, so we can scale its island a little bit more.
5. Once you've packed all your islands, you can add a checker texture to check whether there are any important stretches with your UVs. We will add a predefined texture by first entering into the **Edit Mode** of the alien.
6. We will click on the **+ New** button in the header of the **UV/Image Editor**, and select a **UV Grid** image under the **Generated Type** drop-down menu. We can then validate it with **OK**.

7. In order to see the test grid on our alien in the viewport, we will need to enter the **Texture shading** mode located under the **Viewport shading** drop-down menu in the 3D viewport header. We can also toggle it on or off by pressing **Alt + Z**.

Now that we see our test grid on our alien, how can we interpret it? If all the squares are still approximate squares, then it means we don't have any important deformation. Their orientations don't matter too much here.



The final UV Island placement

The baking of textures

Now that we have UVs on our little alien, we can take a look at how we can transfer all the details from our sculpture to the new **retopologized** mesh. The details in the sculpture are simply there because of

the amount of geometry it is composed of, but in the retopology, we have kept the amount of details to about 5000 polygons, in order to have a manageable mesh.

The baking of a normal map

The best solution that we have in order to transfer the details is to bake a normal map that will act with lights to give the impression of detail.

What is a normal map?

At its lower state, a normal map is an image or a texture that will be projected on the mesh through the UVs. This' is why it's important to UV unwrap the object. Note that if the UVs are stretching at some place, the details that are in the normal map will then be stretched too. As mentioned before, a normal map will do its magic with the lighting. It is composed of red, green, and blue pixels that respectively represent the X, Y, and Z orientations of the normal of a face. So, we will need to create a normal map that will contain all the normal information of the high poly sculpture. The higher the definition of the normal map texture will be, the most precise the details will be.

Making of the bake

In order to create a normal map from our sculpture, we will need to use the bake tools of Blender.

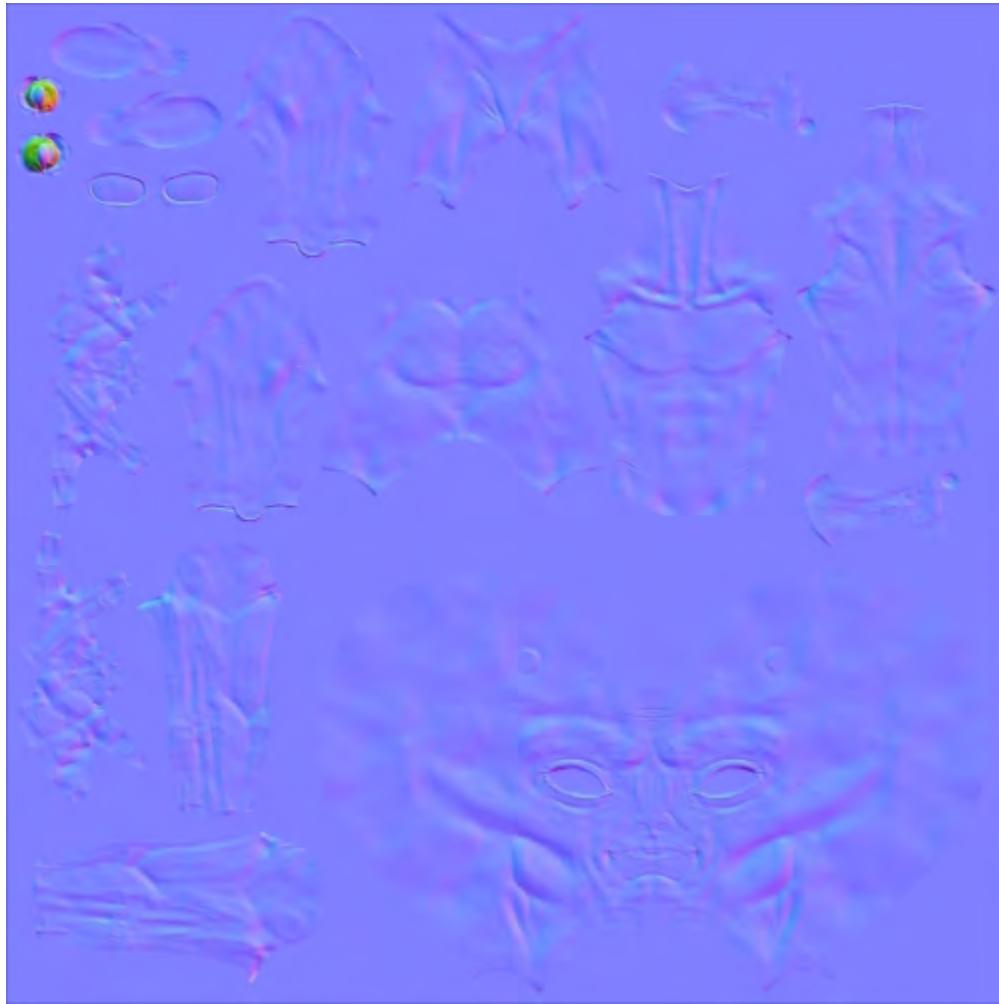
1. We only need to see the sculpture and the retopology in the 3D view. To hide the rest, we will select them both and press *Shift + H*.
2. We can now deselect all our objects by pressing *A*.
3. In order to do the bake, we will have to first select the alien sculpture and then the retopology (this becomes the active object).
4. Now, in the Properties editor, under the **Render** section, we will expand the **Bake** subpanel.
5. The first option to choose is what type of map (or texture) we want to bake. So, under the **Bake Mode** drop-down menu, we will select **Normal Map**.
6. The next thing we'll have to check is the **Selected to Active** option that tells Blender to bake from the sculpture to the active object (our retopology).
7. We will then need to add a blank texture to bake our normal map on. So, we enter the **Edit Mode** of the retopology and click on the **+ New** button (or the **+ icon** on the right-hand side of the texture list), and instead of selecting a UV grid, we choose a **Blank** texture with a width and height of 4096 pixels.
8. Before baking our map, we need to click on the **Smooth Shading** button; otherwise, we will see the polygons on our bake.
9. Last but not least, we will come back to the **Bake** panel and click on the **Bake** button. Don't forget to save your map (**Image | Save As Image** or simply press *F3*) or it will be lost!

At this point, you should see that the normal map has started to appear. If you get an error message, it may be because you didn't add the texture on the low poly while you were in **Edit Mode**, or that you've selected the retopology before the sculpture. If you want to have your normal map packed into the **.blend** file, you can go to the **File** menu and select the **Pack All into blend file** in the **External Data** subpanel.

Note

About the size of the textures

Usually, textures aren't rectangle. They are set with the power of two of the width and the height. The common sizes are 256 x 256, 1024 x 1024, 2048 x 2048, and 4096 x 4096.



The baked normal map of our alien

Displaying the normal map in the viewport

Now that we have a nice normal map baked, we will show to you how to display it in the viewport:

1. We first need to be in the **GLSL** mode (press **N** and select **Shading Subpanel | Material mode** dropdown).
2. We also need to add a new material on our low poly. To do this, we can click on the **New** material button under the **Material** tab of the Properties editor.
3. We will then go to the **Texture** tab of the Properties editor and click on the **New texture** button.

4. Under the **Image** subpanel, we can choose our normal map with the left-hand side drop-down menu.
5. Then, we will check the **Normal Map** check box under the **Image Sampling** subpanel.
6. For now, Blender will only interpret the map as a diffuse map. We want to tell Blender to use the normal information of the map. So, under the **Influence** subpanel, we uncheck **Color slider** and check **Normal slider**.
7. Now, we will need to add a light in the 3D viewport (press *Shift + A* and navigate to **Lamp | Hemi**) and orient it correctly.
8. Lastly, we need to enter the **Texture shading** mode. We can now appreciate the comeback of our sculpture details. If you want, you can move the light around to feel the relief of the normal map.

The baking of an ambient occlusion

Now that we have the normal map set, we will improve our alien with another map called an ambient occlusion.

Understanding the ambient occlusion map

An ambient occlusion is a black and white texture that represents the contact shadows of a mesh. The contact shadows are the shadows produced by the small proximity of objects. In order to have a nice ambient occlusion, we need to increase a sampling parameter that corresponds to the "noisiness" of the shadows. The more samples you have, the smoother the shadows will be. This map will then be multiplied on top of our diffuse material color.

Note

Multiplying colors

In computer graphics, black is represented by a value of 0 and white by a value of 1. So, when we multiply a color with black, its result is 0, and when we multiply a color with white, its result is the color. For instance, we will take two colors, J and K, a pure blue that is represented by $R(J) : 0$, $G(J) : 0$, and $B(J) : 1$ (**RGB** means **Red Green Blue**), and white, $R(K) : 1$, $G(K) : 1$, and $B(K) : 1$. When we multiply both, we will have $R(J)*R(K) = 0 * 1 = 0$, $G(J)*G(K) = 0 * 1 = 0$, and $B(J)*B(K) = 1 * 1 = 1$, so the resultant color is $R : 0$, $G : 0$, and $B : 1$. It is the original blue.

Creation of the bake

We will now follow the same principle as the normal map, but we will change the sampling value:

1. The sampling slider is situated under the world tab of the Properties editor in the **Gather** subpanel. Even if it's grayed out, it will work for the bake. We will set it to **10** in our case. Don't go too high with this as it will increase your baking time.
2. Now, refer to the normal map baking process, but instead of choosing a normal map in step 5, choose an ambient occlusion. Again, don't forget to add the blank texture in **Edit Mode**, and save it after the bake. You can also name your textures in the **UV/Image Editor** in the header with the corresponding text field.



The baked ambient occlusion map of our alien

Displaying the ambient occlusion in the viewport

In order to see the ambient occlusion applied to our mesh, we will have to add a new texture to our material. This is done as follows:

1. First, we will select our alien, and then we will select the material that has the normal map on it in the **Material** tab of the Properties editor.
2. We will, then, go to the **Texture** tab of the Properties editor and add a new texture below the normal map. In order to do this, we select the second slot and click on the big **New** button.
3. We can now choose our ambient occlusion under the **Image** subpanel.
4. Under the **Influence** subpanel, we turn the color slider on but we change the **Blend Mode** from **mix** to **multiply**, as we had previously explained. As you can see, this perfectly works in the viewport when the **Texture** shading mode is turned on. We can clearly see the contact shadows of our mesh around his eyes, for instance.



The alien with a proper topology (shown on the left-hand side) and with its normal map and ambient occlusion (on the right-hand side)

Summary

In this chapter, we saw how to create a proper retopology based on the sculpture made in the previous chapter and retrieve its details with a normal and ambient occlusion map. There are other maps that you may want to create, such as a diffuse, displace, or lighting map. Now let's go to another project, the Haunted House!

Chapter 4. Haunted House – Modeling of the Scene

Welcome to the scary project!

In this chapter, we will model a haunted house that we will texture and render in the future chapters. You will use the modeling techniques that we have already seen in the previous chapters and learn some new techniques using some useful modifiers and time-saving tools. Moreover, you will learn how to correctly organize your scenes by grouping objects and placing elements in layers. Now that you have more experience with Blender, we aren't going to show you all the steps in detail but rather describe the key points of the process. Let's start our scene! In this chapter, the following topics will be covered:

- Modeling on scale
- Blocking the house
- Advance modeling tools
- Modeling with curves
- Organizing the scene

The final haunted house should look like the following screenshot:



Blocking the house

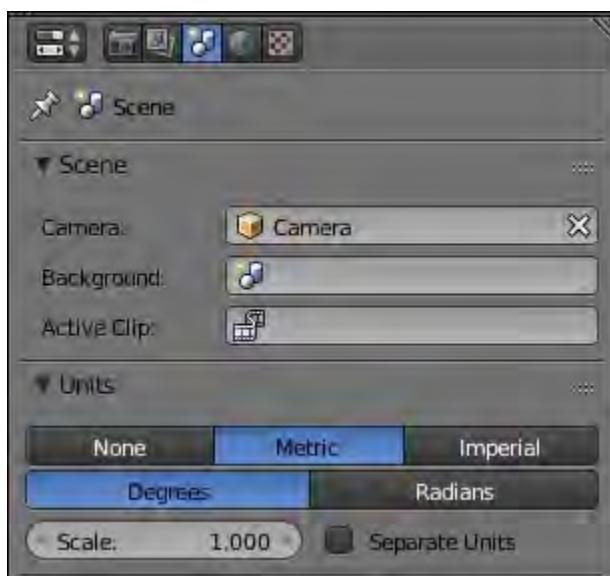
Before going into detail, we will start by testing different shapes in order to create the concept of our house. It is like a 3D sketch.

Working with a scale

In order to create our haunted house and its environment, we need to work with a real world scale. Indeed, when you are working on objects, such as buildings, where the scale matters, it is important to remember to adjust the units of measurement of Blender.

Blender uses, basically, its own unit of measure: the **Blender units** that correspond to a fictitious unit of measure. You aren't going to encounter Blender units in the real world.

There are two other unit systems of measurement in Blender that you can use: the metric system and the imperial system. We prefer the **metric system**. For this, go to the Properties panel on the right-hand side of the user interface under **Outliner** (in the default layout). In the **Scene** tab, you will find the **Units** tab. Choose **Metric** and **Degrees**.



The metric system allows us to work in kilometers (km), meters (m), centimeters (cm), millimeters (mm), and micrometers (μm). Let's choose meters in our case. For this, we set the **Scale** value to **1.000**. A value of **0.1** would make us work in centimeters.

To know the size of your 3D models, in the **Object Mode**, you can look at their size in the **Transform** tab on the right-hand side panel of the 3D viewport (*N*). This information is also given in the **Dimensions** section for the *x*, *y*, and *z* axes.

You can then display the size of the selected edges. In the **Edit Mode**, under the **Transform** tab, go to the **Mesh Display** tab, and check the **Edge Info | Length** option. If you want to measure something, Blender gives you a ruler under the **Grease Pencil** tab in the left 3D view panel (*T*). To use this, simply click on the **Rule/Protractor** button and drag it in the 3D view.

Be careful to always apply your scale and move or rotate the transformation of your objects when you manipulate them in the **Object Mode**. To do this, we open the **Apply** menu (*Ctrl + A*) and select **Rotation and scale**. It is important to avoid involuntary deformations after this.

Blocking the bases of the house

To make this house, we don't start from concept art but from an idea and a few references found on the internet.

It is very important during any creation to spend a little time documenting to confront the different possibilities of shapes and styles. We need to see what has been done previously and be informed enough to be precise in our work.

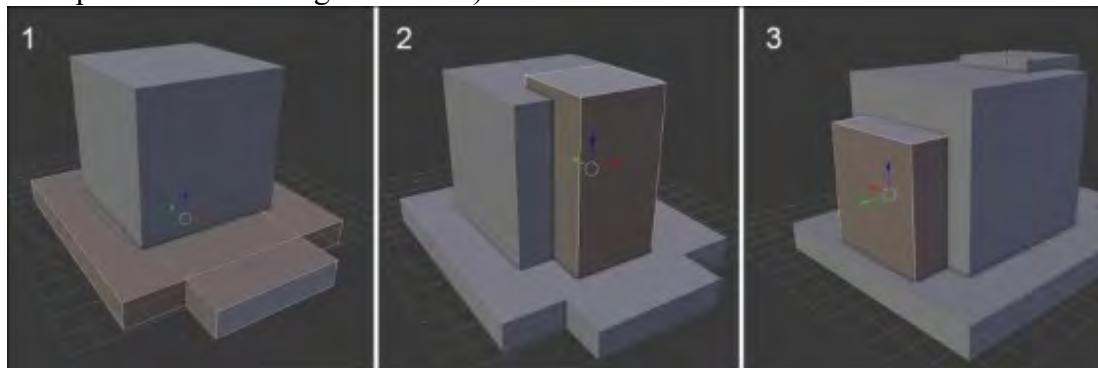
As we are not completely sure of the form as a whole, we will adopt a method that involves testing and quickly developing ideas with simple forms. This method is called **Blocking**. This is done as follows:

1. We will begin the modeling by adding a cube at the center of a new scene (*Shift + A* and select **Mesh | Cube**), which will represent the central part of the house.
2. In the **Object Mode**, we will adjust the size in the **Transform** tab to have something realistic. It is an imposing house, so we will set 7 m on the z axis, 7 m on the x axis, and 8.5 m on the y axis.
3. In the **Object Mode**, let's duplicate our cube (*Shift + D*) in order to make the terrace. So we will scale it to a height of 1.26 m, then we will place it at the base of our haunted house under the main block previously created.
4. The terrace is not completely cubic. We will add two edge loops and an extrusion to the front, which is less wide (on the x axis) than the main block.

It is necessary that this terrace is large enough to be credible, so we will create a passage of at least two meters wide. It is not necessary to be very accurate for the moment, but be aware of your measures, and remember to apply the transformations when you switch back to the **Object Mode** (refer to step 1 in the following screenshot).

5. To improve the general shape of our house a little, we will add a new cube that fits in the front of the central part of the house, centered on the x axis.

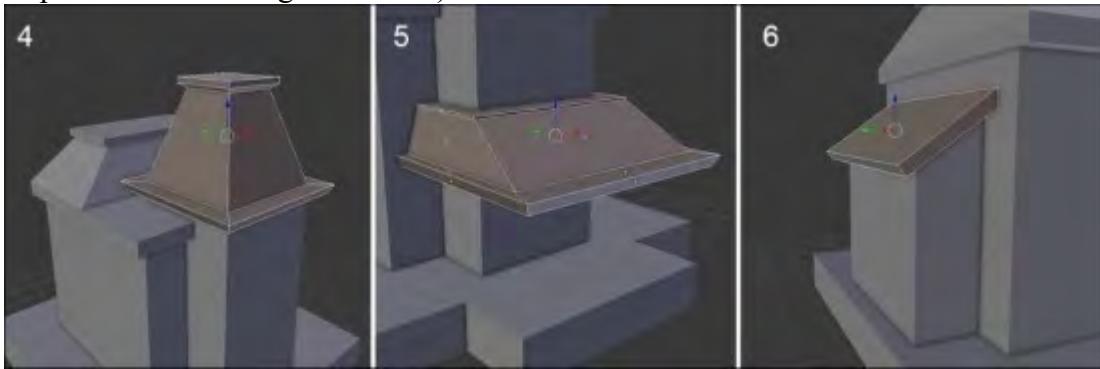
The height of this cube exceeds the height of the other cube by one third. The rest is hidden in the central block. It has a square base, and it is higher than the main block by about 45 cm (refer to step 2 of the following screenshot).



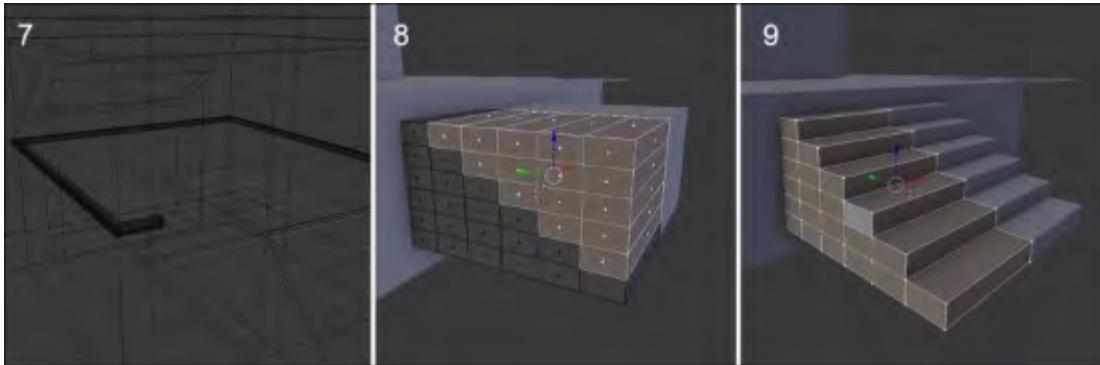
- Likewise, on the rear part, we will duplicate our front block (*Shift + D*) and move it to the other side on the *y* axis. This block is lower than the main block. Its size is 4.7 m on the *x* axis, 1.67 m on the *y* axis, and 5.3 m on the *z* axis.

Now we have our basic volumes. We can now make the roof that is composed of several parts; one for each block. This is done as follows:

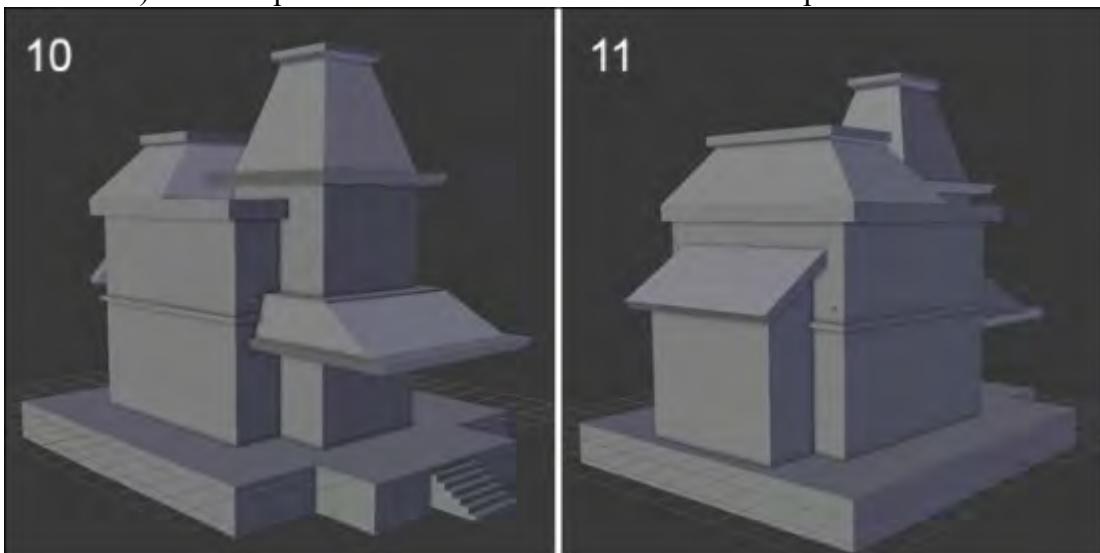
- We will begin with the roof of the front block by adding a new cube (press *Shift + A*, and select **Mesh | Cube**). We need to adjust its size to be larger on the *x* and *y* axis.
- We will move down the top face to flatten it. It is the bottom part of this section of the roof.
- Then we will do an inset (*I*) and an extrude (*E*) to the top. We will adjust the scale of the top, and then we will add two extrusions (*E*) to make the top thicker and finish the shape (refer to step 4 of the following screenshot).



- We will duplicate this part of the roof (*Shift + D*), and we will place it on the rear half of the central block. We will scale it on the *x* axis (press *S + X*) in order to have the same width as the central block. This is also lower than the roof of the front part, so we will also scale it on the *z* axis (*S + Z*) (refer to step 4 of the following screenshot).
- In the same way, we will make another roof that covers the front portion of the terrace. It will be supported by pillars. We will again duplicate our roof that is cut in half, and we will adjust it according to the dimensions of the front of the terrace. We will remove the top part to form a small balcony (refer to step 5 of the preceding screenshot).
- For the roof of the rear block, we will slightly change the style with a simple tilted platform. We will change the rear block to bevel it. We will duplicate the top face (*Shift + D*) and make a new object with it (press *P* and select **Selection**). We will need to make an extrusion on the *z* axis to add a thickness, then we will adjust the size and the position of the wireframe in the **Shading Mode** (refer to step 6 in the preceding screenshot).



7. We will now mark a boundary of two floors with a concrete ledge. For this, we will need to add a new cube (press **Shift + A** and select **Mesh | Cube**) scaled on the *y* and *x* axes (*S + Shift + Z*) to be around the main block. We will give it a height of 15 cm and make it exceed the block by about 20 cm (refer to step 7 in the preceding screenshot). We will do an inset (*I*) on the top and bottom faces, then we will delete the nonvisible faces.
8. Let's form the stairs. We will add another cube, then we will resize it to be 84 cm on the *z* axis and 1.5 m on the *y* axis. We will need to divide it into six equal parts horizontally and vertically. In order to gain time, we will add a **Mirror modifier** (refer to step 8 in the preceding screenshot). We will remove the unwanted faces, and then extrude the contour of the top towards the symmetry axis to create the missing faces (refer to step 9 in the preceding screenshot). We will place our stairs in the middle of the front part of the terrace.



These few simple 3D models done in a short time gives us an idea of what our haunted house will look like with further modeling.

Refining the blocking

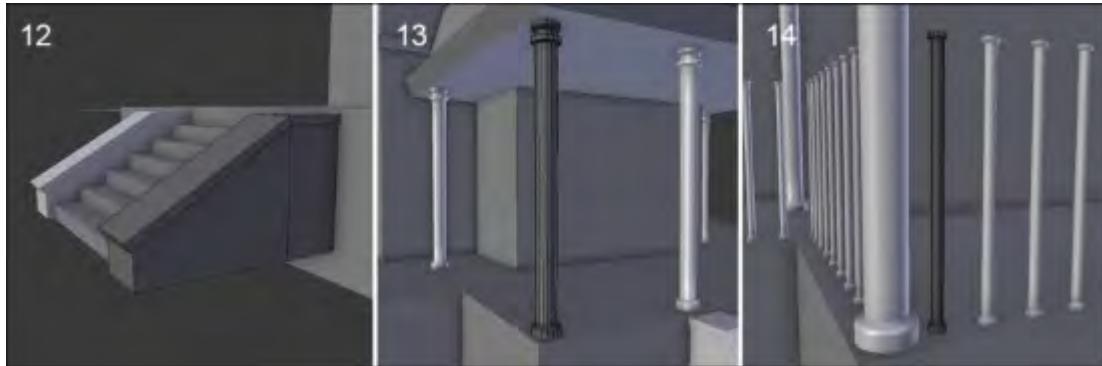
Now that we have the foundations of our model, we will go into the details by adding more defined objects.

Adding instantiated objects

If we analyze the majority of the houses, we can see that they are mainly composed of repetitive shapes such as windows and doors.

So we will use the techniques that allow us to duplicate objects by instance. This means that if we change the geometry of the source object, all the duplicates will change too. As you may have understood, this is really useful in order to save time: for instance, with UVs. Now, perform the following set of steps:

1. Let's start with the low wall around the steps. We will add a cube that we will orient with the slope of the stairs.
2. We will add an edge loop in order to break the slope. Then we will add a thinner piece that recovers the slope. To do this, we will extrude the top faces and scale them appropriately on the same level, and we can redo an extrusion.



3. In order to mirror the other side of the low wall, we will center the pivot point of the stair object at its center (*Ctrl + Alt + Shift + C* and select **Origin to Geometry**). Now we can safely add a mirror modifier with the stairs as the mirror object (refer to step **12** of the preceding screenshot).
4. Next, we will do the columns that will support the roof that covers the front of the terrace. We will need a 16-face cylinder that is 18 cm wide with a height of 2.8 m. In the last tool options in the left 3D view panel, we will choose the **Nothing** option under **Cap Fill Type**. We will then position it on the left-hand side of the stairs, and we will duplicate it as instances with the **Duplicate Linked** (*Alt + D*) function.
5. In the **Object Mode**, we will place our columns at the four corners of the terrace roof. In order to add some details to the columns, we will add some loop cuts (*Ctrl + R*) on the top of the roof and extrude the face loops along the normals (*E + Alt + S*) (refer to step **13** of the preceding screenshot).

Note

Duplicate Linked

This tool allows you to duplicate your 3D model as instances. This means that when you do a modification in the **Edit Mode** on the source object, the transformations are applied to the other duplicated objects in real time. The UVs are also instantiated. However, when you manipulate the object in the **Object Mode**, the changes are not reflected in the other instances.

In order to break the instantiation link, we can use the **Make Single User** menu (**Call Menu** (press *U*) | **Object and Data**).



6. We will now work on the bars that delimit the terrace. We will take a new cylinder, this time thinner with a radius of 5 cm and a height of 1.2 m. We will again remove the caps that are pointless here.
7. In order to duplicate our 3D object, we are going to use the Array modifier. We will use a relative offset of 3.100 on the *x* axis. We will take advantage of the replication of the array in order to improve the shape of the bars a little bit with some loop cuts and extrusions (refer to step 14 of the previous screenshot).
8. Since the bars are along a straight line, we will duplicate them with a normal duplicate (*Shift + D*) to place them on each side of the terrace. We will also need to adjust the number of bars to match the surface of the terrace.

Note

The Array modifier

This modifier allows you to duplicate your 3D models with a customizable offset. You only need to choose the number of repeated objects that you need with the **Count** parameter and the distance of the offset (constant or relative) on any axis. You can also automatically merge your duplicated polygons with the **Merge** option.

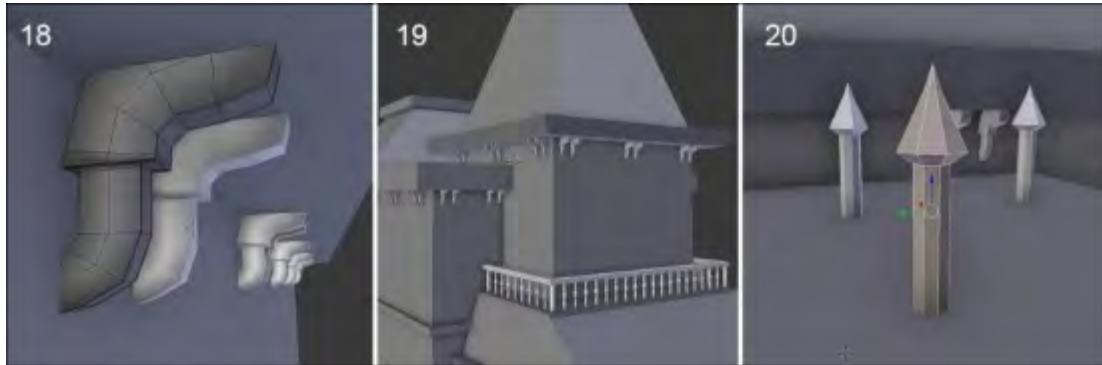


If you want to modify the geometry of a mesh, the array takes the object volume into account, so be careful. A transformation in the **Edit Mode** can change the offset.

9. We will complete this with ramps. The ramp is a simple cube scaled on the x axis to make it longer. We will duplicate the ramp as an instance ($Alt + D$) wherever needed, but remember that you need to duplicate with $Shift + D$ if you want to do some changes in the geometry (refer to steps **15** and **16** in the previous screenshot).

The Duplicate Linked tool is very useful, but it is not very flexible when we only want to do a transform on certain objects.

1. Let's repeat the same technique to make a balcony railing on the top of the roof that covers the front of the terrace. We are going to change the shape of bars a little (refer to step **17** of the preceding screenshot).
2. We will also use an **Array modifier** to make the wall brackets that will support the different parts of the roof. We will use one Array modifier to make a pair and another to duplicate it with a good offset (refer to steps **18** and **19**).



3. The same thing is done for the pikes on the roof that give a threatening look (refer to step **20** in the preceding screenshot).



4. The walls are a bit flat at the moment, so we are going to model a bay window with a particular shape (refer to step **21** in the preceding screenshot). We will start with a new cube (*Shift + A* and select **Mesh | Cube**), and we will resize it on the *x* axis to form the base.
5. We duplicate it to make the top part and add an inset for the frames of the windows. When this is done, we will duplicate it as a new instance (*Alt + D*), and we will place it on both sides of the main block of the house.



It is also time to make a few conventional old-style windows. We will start with the windows on the front of the house (refer to step 22 in the previous screenshot). Let's start again from a cube:

1. We will delete the left side to use a **Mirror** modifier. From the front view, we will make the shape of the window with a few edge loops (*Ctrl + R*) and add an inset. From this model we will can make the frame and extract the shutters (refer to step 23 in the preceding screenshot).
2. We will create another window model for the roof. We will start its base with a flattened (*S + Z*) and beveled (*B*) cube. The central part is an extruded edge with an inset, and there is a little roof with a curved slope (refer to step 24 in the preceding screenshot).
3. Let's add a fireplace (refer to step 25 in the preceding screenshot). This is also fairly simple to model, starting from a cube that is extruded and scaled. It uses the same basic modeling techniques that we have previously covered.

Reworking the blocking objects

Until now, we had quickly added a series of 3D objects to form a fairly rudimentary house and have a general idea of the entire model of the house. Now we will finish the modeling and review the topology of our models in order to make them more presentable.

There is still a final important object to be made: the front door. It is composed of several 3D objects. There is the frame, door, lock, handle, and there is the top frame with a decorative pattern. The frame and the door are quite similar to the windows. The door will be created as follows:

1. We will use a mirror modifier each time. The lock is quite simple, just an extruded cube with a few **insets** (*I*) (refer to step 23 in the preceding screenshot). For the decorative pattern, we will use a circle in order to get a better round shape.

The rays are extruded and moved by hand. Don't forget to add a few edge loops when you add the **Subsurface** modifier (refer to steps 24 and 25 in the preceding screenshot).

2. We will place the address number of the house. For this, we will use the **Text tool** (*Shift + A* and select **Text**). We will use the basic **Bfont** font of Blender, but you can download and use any font instead. We will use the **Extrude: 2.3 cm** and **Depth: 1.2 cm** parameters to make a small bevel. We will make two small nails to hold them.



3. We will now review every asset and remove the nonvisible polygons that are useless in our case. Instantiating many of our objects has greatly facilitated the work.



4. We will also often add a **Subdivision Surface** modifier when it's needed.

But sometimes, this is not the case, as a simple bevel will do the job as well. As always, you need to maintain your geometry with loop cuts or bevels. In both cases, we want to activate the **Smooth Shading** (in the left panel of the 3D viewport).

5. We will also detach some parts of the geometry from other objects. To do this, we will go in the **Edit Mode** and we will select the faces that we want to detach by pressing *P* and selecting **Selection**.

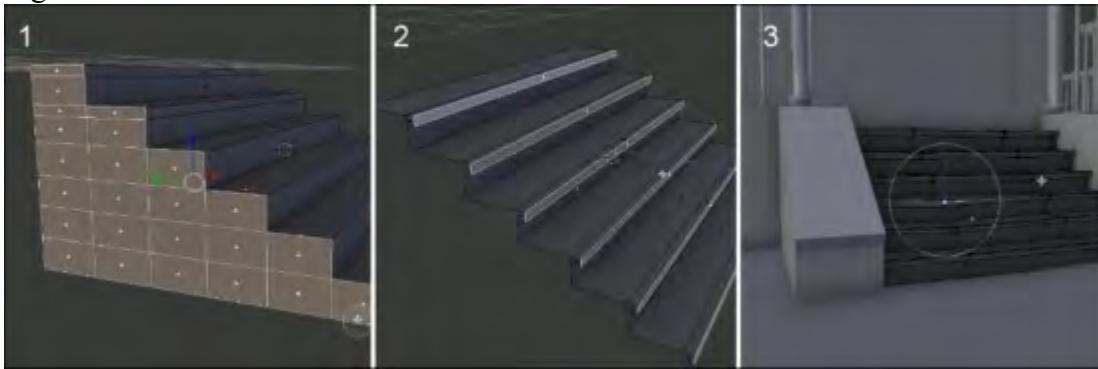
This will allow us to rework certain objects and add the **Subdivision Surface** modifier with ease. For instance, for the roof, we will detach the center part and add tight edges near the four corners with the **Bevel** tool. We can also increase the windows' resolution by adding and placing some edge loops to form a nice round shape in the corners.

All the improvements that we've done here are, in our case, pretty easy because we don't have to get a realistic result. We are taking a more "cartoony" path, and in the next chapter, we will paint textures by hand in order to add more details.

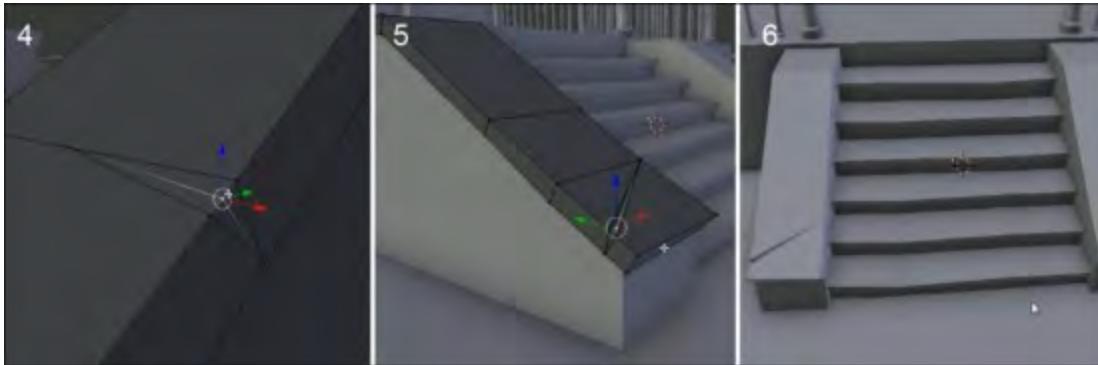
Breaking and ageing the elements

It's now time to refine some elements in order to give them an old/destroyed look. This will mainly come from the texturing work that we will do later, but we can still work on the geometry a little bit:

1. We will begin by working on the stairs. We will delete the side faces that are invisible. (Refer to step **1** in the following screenshot.)
2. Let's add loop cuts below the angle of each stair. This will allow us to extrude a face in the front direction of the top of each stair. (Refer to step **2** of the following screenshot.)
3. We will then add two edge loops at the center to add more resolution.
4. Now with **Proportional Editing** turned on (*O*), we can select some vertices on the stairs and move them to break the shape a little bit. (Refer to step **3** of the following screenshot.)
5. We will now enter the stair ramps object and delete the one on the left. We will have to delete the invisible faces, those that are in the ground and in the house.
6. After this, we can move the origin back to its center (*Ctrl + Alt + Shift + C*) and duplicate the object (in the **Object Mode**) with *Shift + D* in order to move it back to the left-hand side.
7. We can then add definition to the object on the right-hand side with the **Loop Cut** tool.
8. At the top of the ramp, we will use the **Knife** tool and cut around a loop cut that we had made before. We will then push this hole inwards. (Refer to step **4** in the following screenshot.)
9. We can then repeat this process elsewhere on the other object. (Refer to step **5** in the following screenshot.)
10. The stairs are now finished. (Refer to step **6** in the following screenshot.)
11. We can also add a cut in the back wall of the house and move the vertices a little bit to add some sag.



We now encourage you to go over each object and slightly move the geometry, with the **Proportional Editing** tool turned on (*O*), in a random manner in order to break each object's silhouette. You can also rotate the objects, such as the curtains. This is a house that is not new, plus it's probably abandoned!



Simulate a stack of wooden planks with physics

We are now going to add little bit more details by adding a stack simulation of wooden planks in the front of the house. But instead of placing each plank by hand, we will take advantage of the physics engine of Blender that will do the job for us. In order to keep our stack simple and manageable, we will use the instancing principle that we've seen before. Let's start with the modeling of the plank:

1. The plank will be very easy to model by starting with a cube primitive.
2. We will then rescale this cube to make it thinner and larger. Note that if you do this in the **Object Mode**, you will have to apply the scale with *Ctrl + A*.
3. In order that the plank catches the light on its edges better, we will add a small bevel to it. There are two ways of adding bevels in Blender. The first one is the one that we've already used before with *Ctrl + B*. The other method is by adding a **Bevel** modifier. That's what we've done here. Note that if you want, you can apply the modifier too.
4. We will now duplicate the planks by instancing them (*Alt + D*). You need to place them one on top of the other. In order to add a little bit of randomness, we will rotate them slightly in an unordered way.

Note

About the Bevel modifier

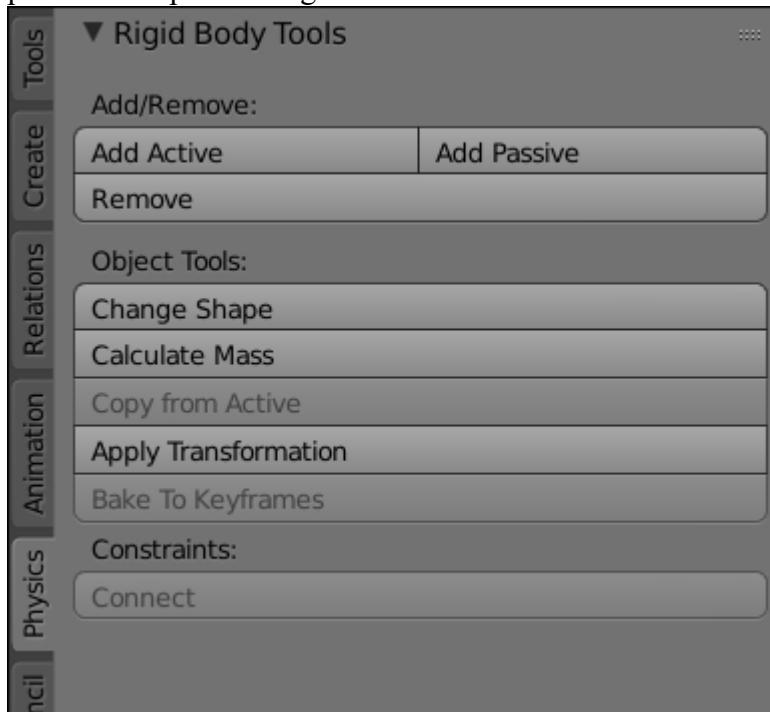
The **Bevel** modifier is nice because it is applied on the whole object, so we don't have to manage a lot of geometry while we are in the **Edit Mode**. We can adjust the **Width** slider to tighten or enlarge the effect of the bevel. The **Segments** option allows us to choose the number of cuts the bevel will be made of. The **Profile** of the bevel corresponds to the direction of the bevel; if it's negative, it will go inwards.

Creation of the simulation of a stack of planks

We will now create our simulation. In any rigid body simulation, the objects have some properties that define them. For instance, you can set their mass, velocity, or simply let gravity act on them as the sole force. In any decent physics engine, you can have static and dynamic objects. A static object is an object that, as its name implies, can't move at all but will be considered in the simulation when collisions occur.

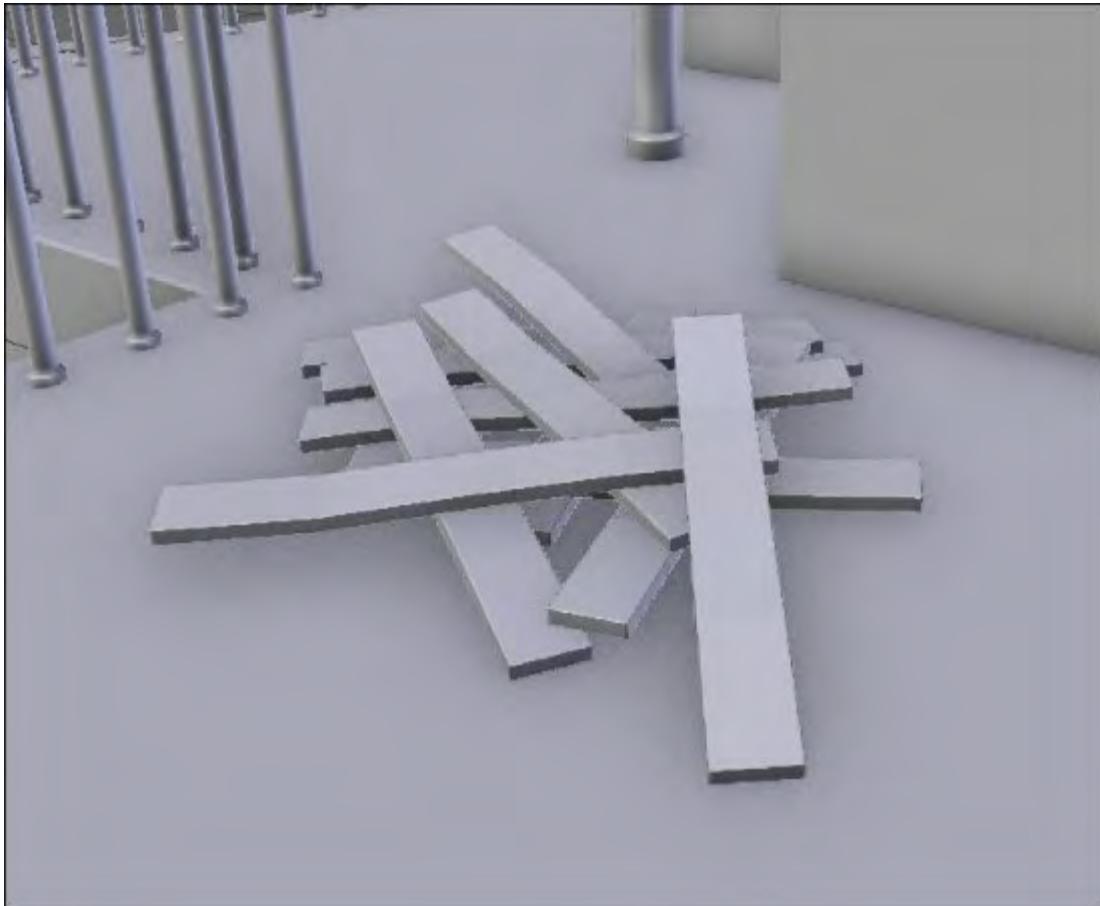
A dynamic object is an object that can receive forces. In Blender, static objects are defined as **Passive** and dynamic objects as **Active**.

1. We will select all our planks, and in the **Physics** tab, in the left 3D view panel (*T*), we will press the **Add Active** button. They will have a green outline.
2. Now we will set the ground object as passive by pressing the **Add Passive** button so that the planks don't pass through the house.



The Physics tab

3. In order to simulate our stack, we will launch the animation. To do this, we will use the *Alt + A* shortcut or press the **Play** button of the **Timeline** editor. Note that if the simulation doesn't seem to launch, you can replace the Timeline bar on the first key frame.



The final stack of wooden planks

4. As you may have seen, there is an orange line on the Timeline that tells us that a simulation has been cached, but as soon as you go backwards in time, the simulation will be removed. So after the simulation has been completed, we will have to apply the placement of each of the planks. In order to do this, we will select them and click on **Apply Transformation** in the **Physics** tab. We can now safely replace the Timeline bar at the first frame, and our stack will rest still.

Modeling the environment (8 pages)

Now that we've finished the house modeling, we will improve our scene with an environment composed of a cliff, a barrier, a cart, and some rocks.

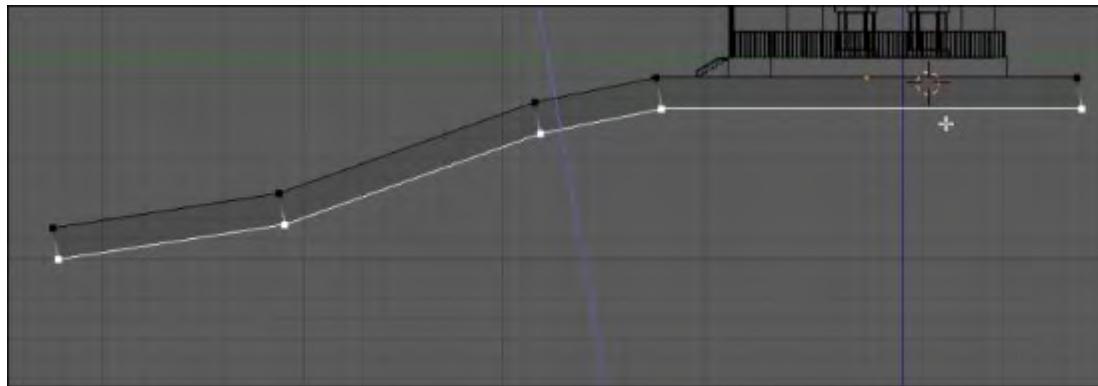
Modeling the cliff

Let's now model the cliff:

1. We will start by modeling its ground part. In order to do this, we add a plane and scale it.
2. We will then move the ground, so the house is placed above it.
3. We will use the scale tool in order to make the ground wider.
4. In the side view, we will enter the **Edit Mode** and activate the wireframe (*Z*). We will select the two vertices in the front of the plane, and by pressing *Ctrl* and the LMB, we will extrude the cliff profile to the left.
5. We will then reshape the geometry from a top view by moving the vertices with the Grab (*G*) tool. The cliff should be narrower where the house is.
6. The whole geometry will then be extruded down to form the height of the cliff.

Note

Note that you may have the normals of the face pointing inwards causing lighting mistakes. In order to recalculate them, use the *Ctrl + N* shortcut.

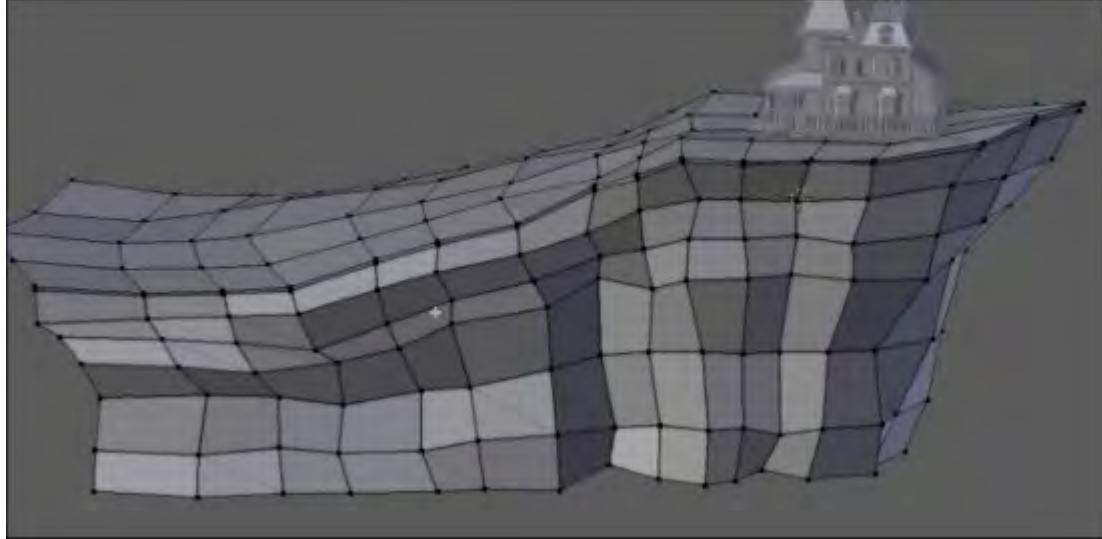


Starting the cliff modeling

7. At this point, we can add some new horizontal loop cuts (*Ctrl + R*) in order to add definition to our model.

Now our goal is to give the impression of a rock with the few polygons that we have. To get the shape, we will go in the top view, and with the wireframe turned on, we will select the edges of the contour of the cliff and move them to form ridges and a valley. This will give an angular look. Remember that all the details will come with the texturing process in the next chapter, so we only have to give the overall shape here.

1. We can improve the model with some randomness. To do this, we go in the **Tool** tab of the left 3D view panel, and under the **Deform** section of the **Mesh Tools** subpanel, we click on **Randomize**. The effect of the tool can be tweaked with the **Amount** slider in the last tool option panel. This tool will simply push all the individual selected components in a random direction.



The final cliff

2. Lastly, we can arrange the global shape of our object with **Proportional Editing**. For instance, we can scale the tip of the cliff with it.

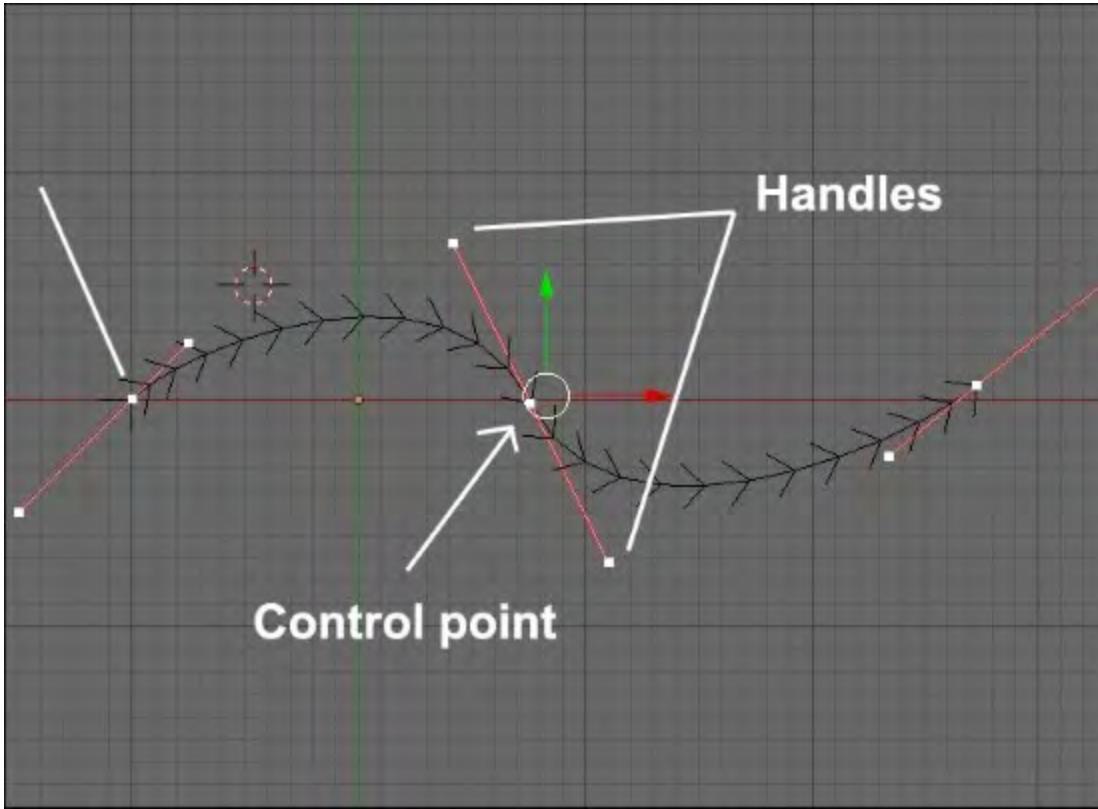
Modeling a tree with curves

We are now going to learn a new way of modeling certain objects with curves. Curves are entities that can be useful to model ornaments, shoelaces, ropes, tree branches, and so on. In our case, we will model an entire tree with them.

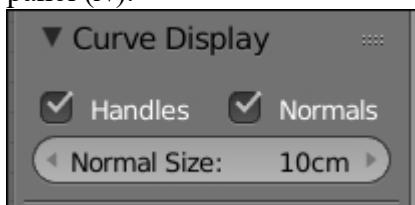
Note

A word about curves

A curve is defined by control points that are connected together. Each point has a handle type. This can be changed by first selecting the control point that you want and by pressing the **V** key. By choosing the **Vector** type, the point will be angular. We are choosing **Automatic** or **Aligned**, so you will have Bezier handles that define the smoothness of the point. Handles always work in pairs. You can break their link by selecting the handle that you want to disconnect and by pressing **V** and select **Free**. In the beginning, you will find that curves are hard to manipulate, but with a little bit of practice, it will become as easy as pie. But, of course, there is lot more to discover about them!



1. We will add a new curve of the Bezier type (press *Shift + A* and select **Curve | Bezier**). Then we will enter in the **Edit Mode**.
2. We will then turn off the handles and the normals of the curve. To do this, we will uncheck the **Handles** and **Normals** check boxes under the **Curve Display** subpanel in the right 3D view panel (*N*).



The Curve display options in the N panel.

3. When you start working on a shape that uses curves, it's easier to set all the control points as **Vector** (press *V* and select **Vector**).
4. Now we will leave the **Edit Mode** and add a new curve object of the bezier circle type (press *Shift + A* and select **Curve | Circle**). This circle will define the volume of our bezier curves.
5. We will select the bezier curve, and in the Properties editor, we will click on the object data icon. Under the **Shape** subpanel, we will then select the circle as the **Bevel Object**. As you can

see, the curve is now getting a volume defined by the circle. If you change the circle, its scale, for instance, will change the curve volume.

6. We will then close the holes on each side of the curve by checking the **Fill caps** option under the **Bevel Object** one.
7. We can now modify our curve to form the trunk of the tree. To do this, we can simply vertically rotate the two points that we already have and extrude the tip several times with *E* or press *Ctrl* and LMB. You can also subdivide the curve by selecting at least two consecutive points and pressing *W* and select **Subdivide**.
8. Now we can change the radius of each point by changing its value in the **Transform** subpanel in the left 3D view panel. This is how we will taper the trunk.
9. We can now add new branches to our tree by simply adding a new point by pressing *Ctrl* and LMB without anything else selected. This point could then be extruded.

We can also change the radius of the branch. If you want to quickly select a branch, you can either select one of its points and press *Ctrl + L*, or select one of its points and press *Ctrl* and the + numpad key several times.

10. We will now add more branches by duplicating the first one that we've created. The process simply involves the duplication and placement of a lot of branches that differ in size and radius.
11. Next, we can convert our curves to a mesh object. In order to do this, we will have to first decrease the subdivision of our branches and our circle profile object by going to their **Object data** tab in the Properties editor and by changing the **Preview U** (for the viewport) and **Render U** (for the final render) sliders. We can now select our curve object and press *Alt + C* and select **Mesh from Curve/Meta/Surf/Text** to convert our curve object into polygonal geometry.



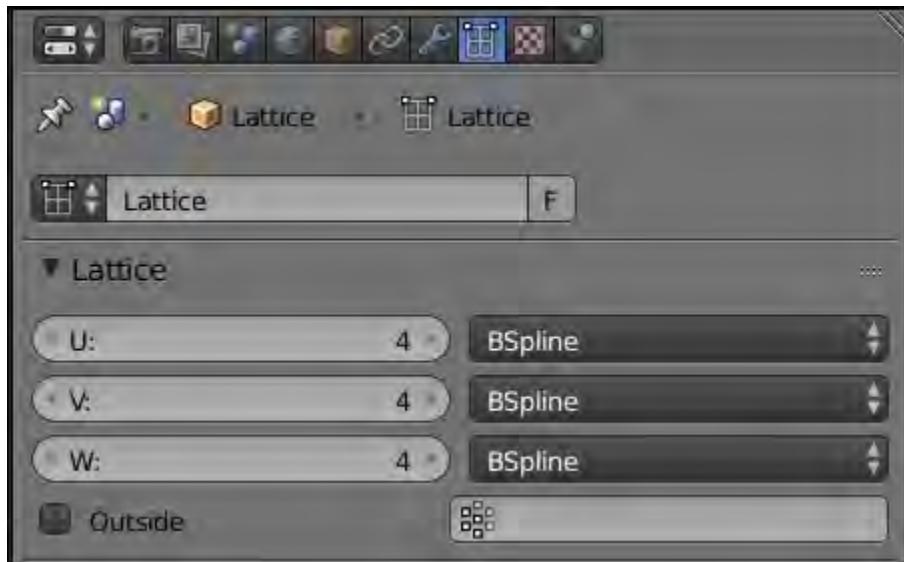
The curves option in the Properties editor

We will now show you how to deform the tree with a special type of object called a lattice. This will serve us to change the overall shape of it.

Note

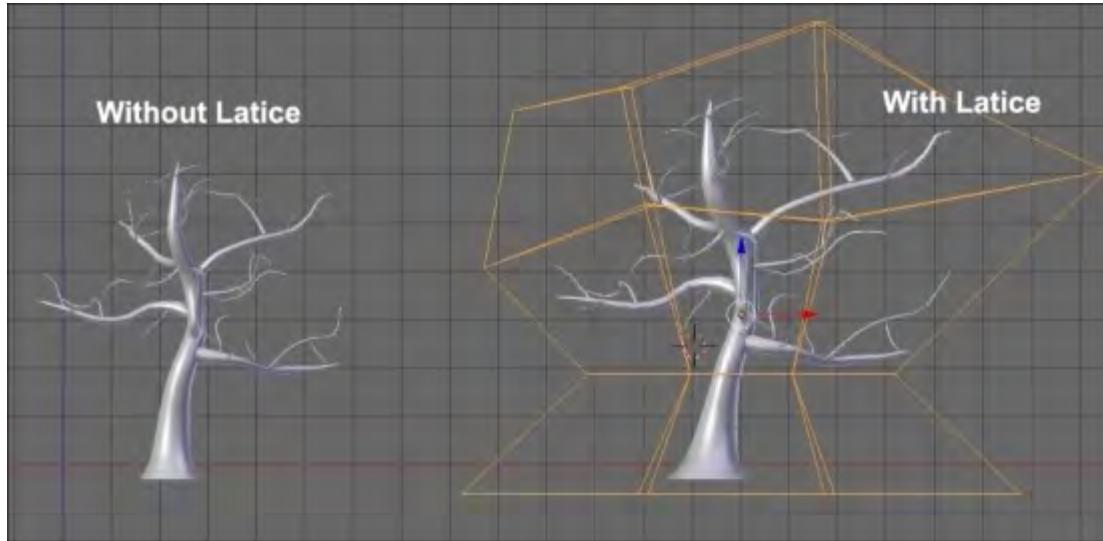
What exactly is a lattice?

A lattice is a very useful object that is bound to another object. It is a cube that can be subdivided along three axes: U, V, and W. This needs to encapsulate the object to which it is bound. Then, when we move its control points, it performs a sort of a projection that allows us to deform the bound mesh.



The **Lattice** options in the Properties editor.

1. We will add a new lattice by pressing **Shift + A** as usual.
2. Now that we have a new lattice, we can place it around our tree. No geometry should be outside the lattice or it will cause problems.
3. We will now increase the subdivisions of the lattice in U, V, and W to **4**. This will give us enough control points in order to tweak our tree.
4. Now it's time to bind our lattice to our tree by simply selecting our tree and adding a new **Lattice** modifier. We will now have to specify to Blender the lattice object that we want to bind to our mesh in the **Object** parameter. In our case, we only have one lattice so it's easy to find it in the list, but if you have many lattices, don't forget to name them.
5. We can now enter **Lattice Edit Mode** and move the control points as if they were vertices.

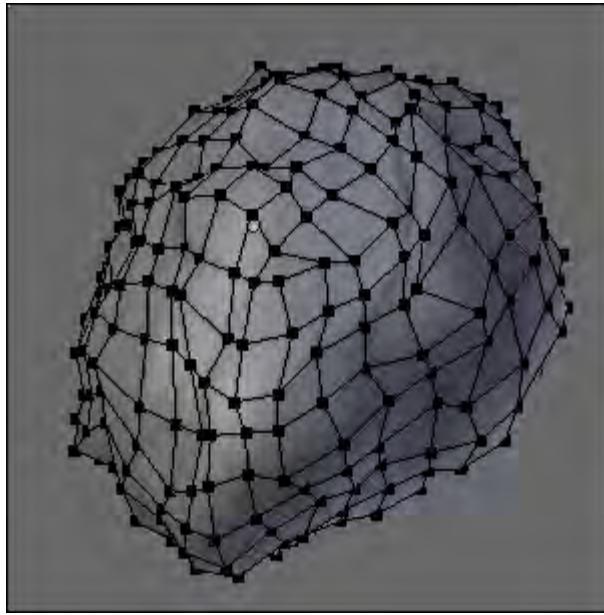


The final tree with its lattice

Enhancing the scene with a barrier, rocks, and a cart

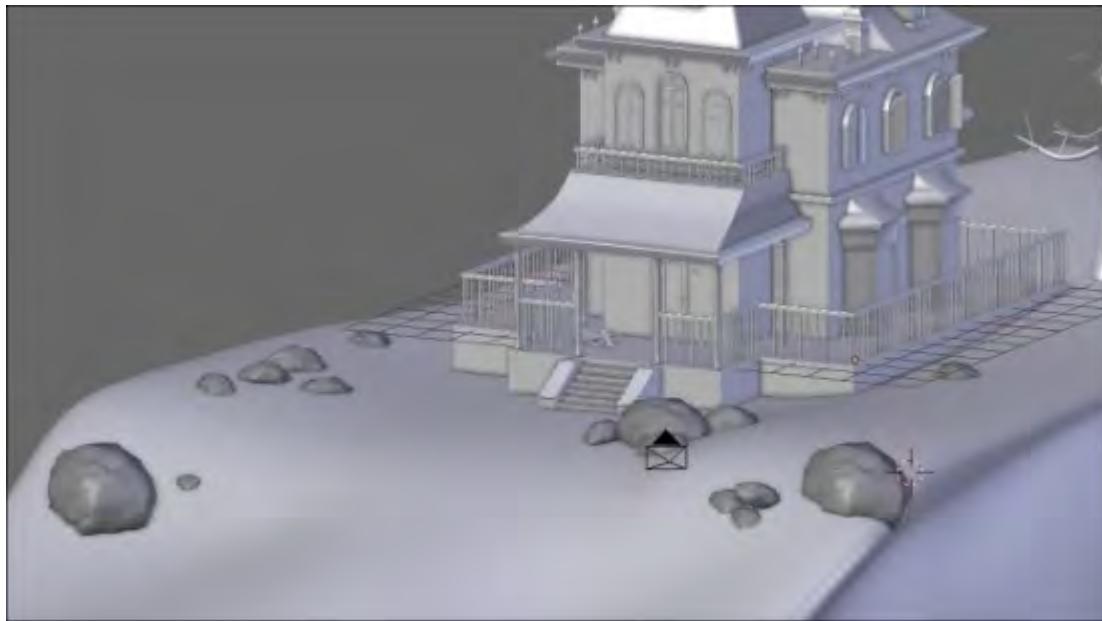
We will now enhance our scene by adding some cool assets using the techniques that we've seen in this chapter:

1. Let's start with the easiest asset, the rock. We will start this element with a cube.
2. Now in **Edit Mode**, we will use the **Subdivide smooth** option (press **W** and select **Subdivide smooth**).
3. In the last tool subpanel of the left 3D view panel, we can tweak the **Fractal** slider in order to add some randomness to the subdivision.
4. We can now repeat steps **2** and **3** several times.



A single rock

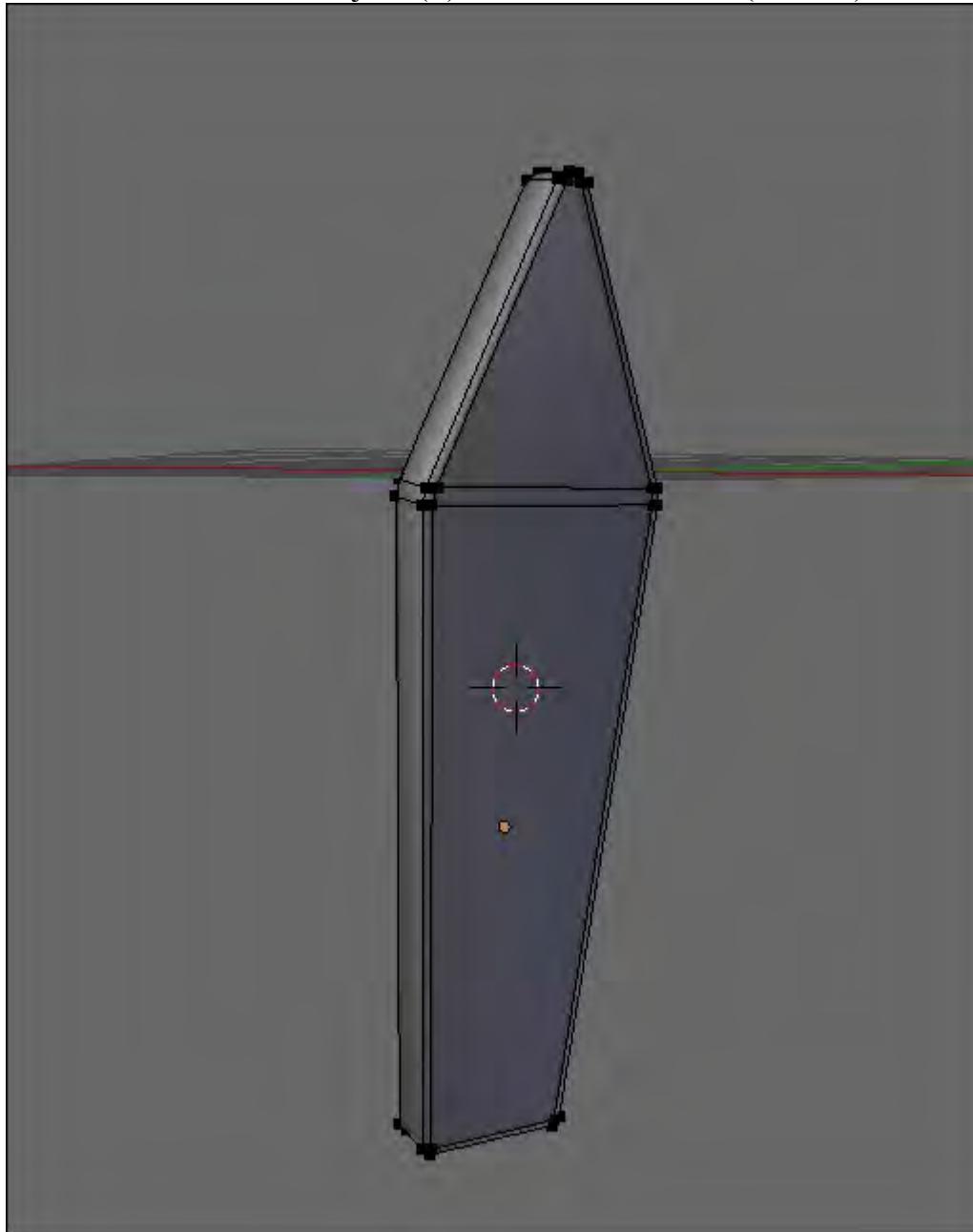
5. Now that we have a pretty spherical rock, we will use a **Lattice** modifier to shape it more like a rock. In our case, we will set our lattice with three subdivisions in U, V, and W, and after we've done the job, we will apply the modifier on the rock.
6. We can now use the instance duplication (*Alt + D*) tool in order to populate our terrain with rocks. This is a little bit of cheating, but we don't have to do any other objects in order to give the impression that there are many different rocks. Indeed, we can simply rotate and scale the rocks in a random manner! In this case, the Free Rotation tool (press *R* key twice) is quite useful.



The rocks are now placed all around the house.

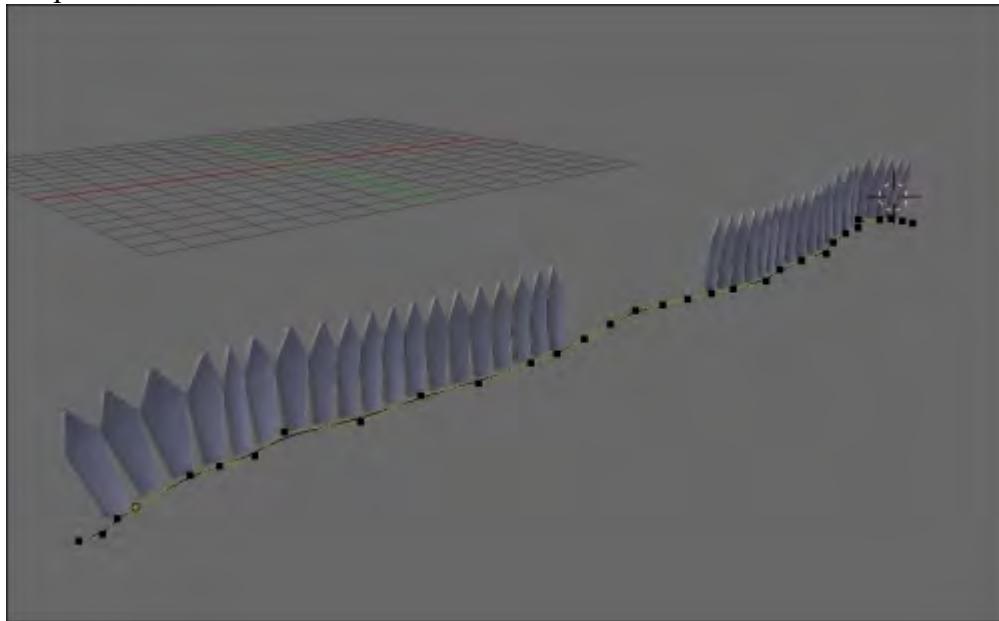
We will now create a barrier in front of the house with a very nice modifier.

1. We will first add a new cube that we will scale on the x and y axes to make it thinner.
2. Next, we can select the top face in the **Edit Mode** and move it to change the height of the plank.
3. With the top face selected, we will extrude the top part of the plank and scale it on X.
4. We can then select all the objects (**A**) and use the **Bevel** tool (**Ctrl + B**) to smooth their edges.



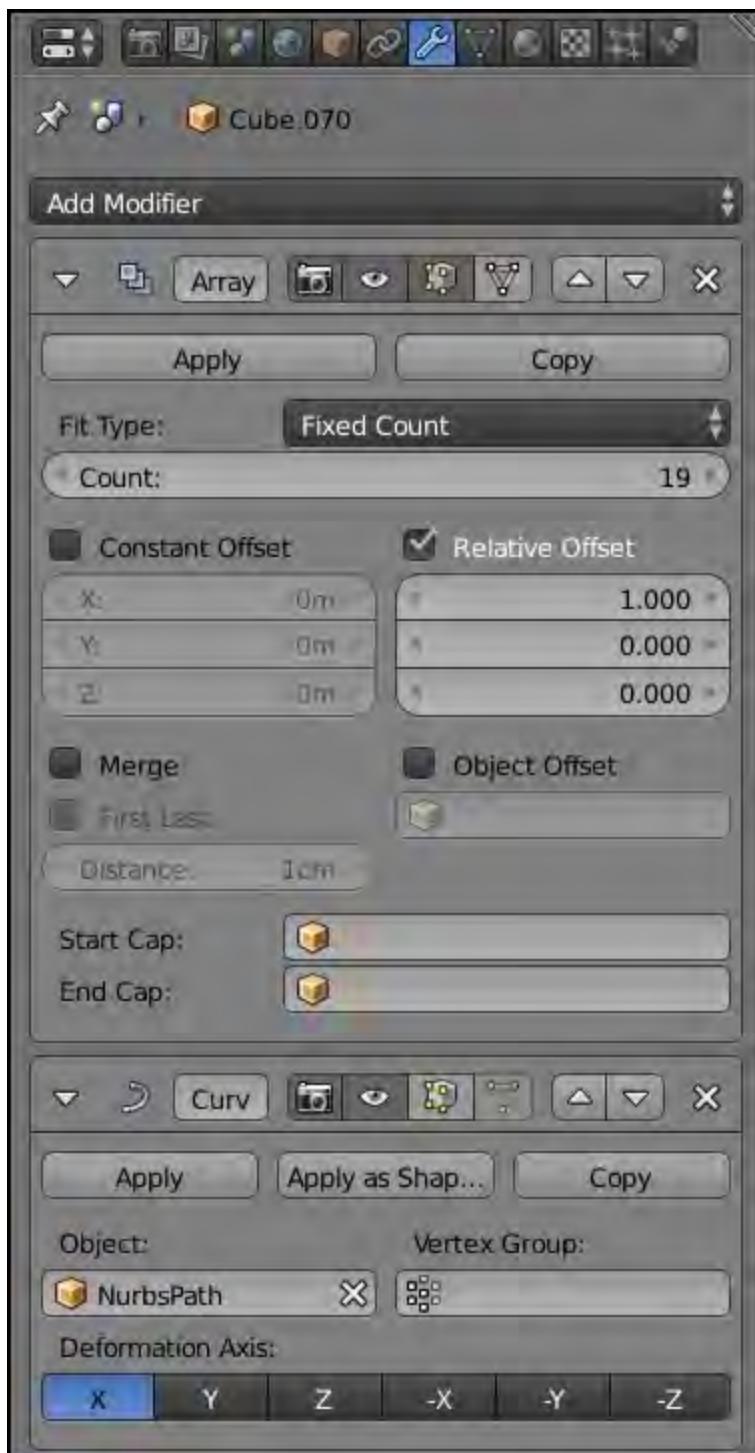
The barrier plank

5. We can now place it on the far left of the cliff in front of the house.
6. The next thing is to use the **Array** modifier in order to do a line of planks to the middle of the house.
7. We can duplicate this barrier piece on the other side.
8. In order to break the boring alignment of these objects a little bit, we will use a path curve (*Shift + A* and select **Curve | Path**) that will guide the planks.
9. The path object looks like a curve. We can select each control point, extrude them, and subdivide them. We will extend the path object from the far left to the far right of the cliff and subdivide it several times (*W* and select **Subdivide**).
10. We can now break the path by moving each control point in a random way.
11. On the barriers, we will add **Curve** modifiers and select our path as the object. In our case, the deformation axis is *x*. Be careful that curve modifier is placed above the array of the stack. Indeed, we want to deform only one plank. As you can see, we now have our barrier following the path.



The final barriers with their curve

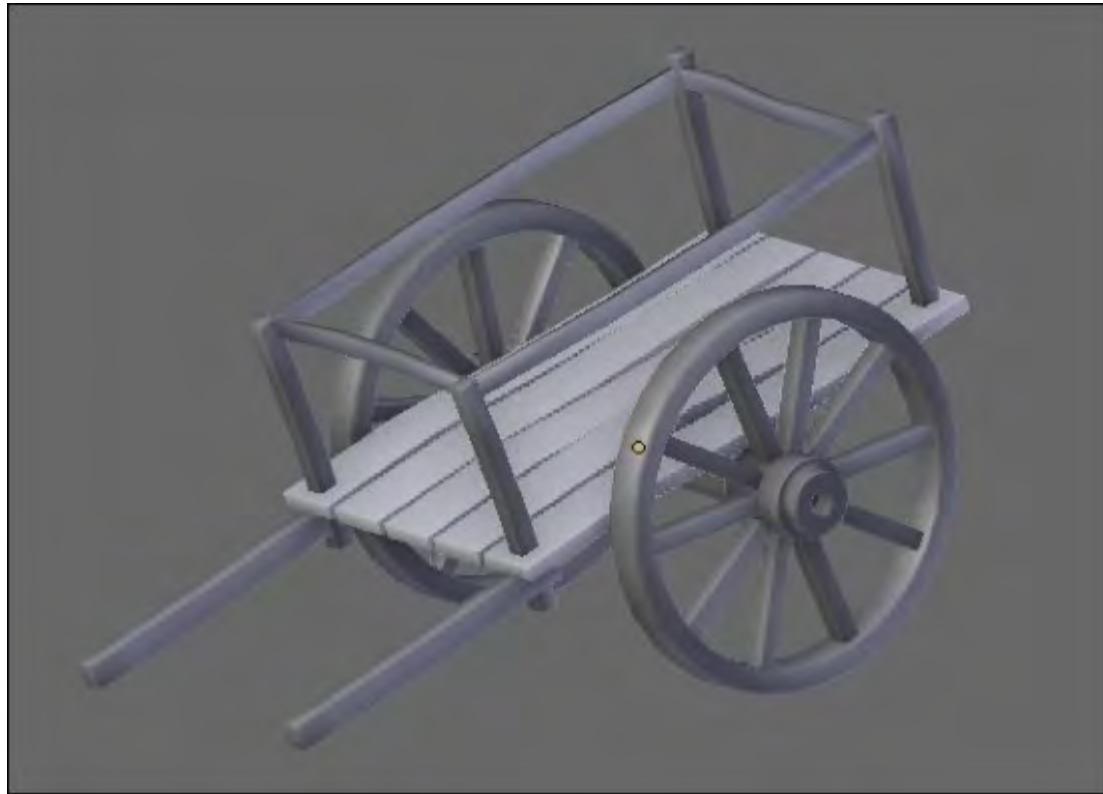
The following screenshot shows the barrier modifiers:



The barrier modifiers

It's now time to model the cart, which is a more complex object, but still simple to do with all the tools you've learned until now:

1. Let's start by duplicating one of the planks that we used in our physics simulation. This time we want to modify it without breaking the ones that are part of our simulation, so we press the ***U*** key and select **Object & Data** to make it a single user.
2. We can now use an **Array** modifier to make the bottom part of our cart. We can specify a little margin between each of them.
3. Now we will apply the modifier and tweak each plank so that they don't look the same.
4. We will now add a small cube and change its height. We can break its silhouette a little bit with **Loop Cut (Ctrl + R)** that we will move, and finally, add a bevel.
5. Now we can duplicate this object (***Shift + D***) at each corner of the cart.
6. We can now add a small plane that we will extrude by following the outline of the cart. We will add a **Solidify** modifier and apply it.
7. We can now add a new cube that we will shape into handles. As always, it's a matter of loop cuts, bevel, and placement.
8. Now, the last thing we need is two big wheels. To do this, we will add a circle, and in the **Edit Mode**, we will extrude it along the normals (press ***E + Alt + S***). We can also extrude the wheel to give it a thickness. Remember to check your normals' orientations and clean them with ***Ctrl + N*** if needed.
9. Now we will add some radius. To do so, we first select one of the inner faces of the wheel, duplicate it and extrude it. We can select the whole radius and press ***P*** for **Separate selection**. We will also need to change its pivot point at the center of the wheel (press ***Ctrl + Alt + Shift + C*** and select **At center**).
10. We then duplicate the wheel with ***Alt + D***, and without validating the action, we will press ***R*** to rotate it. This is because now we can press ***Ctrl + R*** to repeat the duplication and rotation process. What a time saver!
11. In order to cap the wheel, we will add a cylinder that will be extruded several times.
12. Finally, we can duplicate the wheel on the other side with ***Alt + D*** and add a cylinder that will connect both wheels. Remember to delete the unneeded faces on each side.



The final cart

As you can see, the items that we created are quite simple, so we encourage you to add more of them, such as a dead trunk or a scarecrow. Be creative, you have all the tools needed to do that yourself.

Organizing the scene

This is maybe the largest scene you have ever made, in terms of details and number of objects. So we will take some time to learn how to organize ourselves in order to have a more manageable scene.

Grouping objects

One interesting thing that Blender has to offer is the grouping tool. Let's see how it is working:

1. We will group objects that have a logical relationship between them. So we first select every part of the house, such as the foundations, the walls, the curtains, and the fireplace, and we will place them in a group by pressing *Ctrl + G*.
2. If we look at the last tool options in the left 3D view panel, we can enter a name for our newly created group.
3. Now we can select barriers and create a new group with them.
4. We'll leave the cliff alone, so we don't have to put it in a group.
5. The rocks will also be part of a group.
6. We will also group all the cart pieces.

Note

Using groups

Groups are very useful as they allow you to organize a scene better. You can see all the groups that are in a scene in the outline by changing the **All Scenes** drop-down menu to **Groups**. Here you can select them, disable their render, and hide them. You can also select one object that is part of a group in 3D view and press *Shift + G* and select **Group** to select all the other objects of the same group. You can add or remove an object from a specific group in the **Object** tab of the Properties editor under the **Groups** subpanel.



Working with layers

Another thing that can help your scene organization are the layers. They are located in the 3D View header. You can move objects on specific layers with the *M* key. A popup will show you all the layers.

You can move an object on multiple layers by pressing shift and selecting the layers that you want. The layers that have objects are marked with a little circle. You can display multiple layers at a time by pressing the *Shift* key and clicking on them in the header bar. Note that you can also see the layers in **Scene** tab of the Properties editor. In our case, the house is on layer one and the other elements on layer two.



Layer one is selected, but we can see that there are objects on layer two

Summary

You have completed the first part of the Haunted House project! You have learned some tricks to save time, such as how to instantiate objects or how to use the array modifier. You have also learned another modeling method with curves. You will clearly see the purpose of duplicating objects by instance in the next chapter where we will UV unwrap the objects. But don't think that it's over with the modeling process here. We can later change our objects after the texturing part in order to break the likeness between each instantiated object. Let's texture our scene!

Chapter 5. Haunted House – Putting Colors on It

This chapter will be devoted to the texture creation pipeline. We will explain new ways to you for the UV unwrapping processes, such as the **Project From View** or the **Smart UV** method. Then we will cover the basis of the powerful Texture Paint tool in order to create hand-painted textures for our house and environment. You will also learn about the tiling method in order to save time. In order to enhance the final set, we will show you how to create transparent textures, such as grass or grunge to age the house. In this chapter, you will not learn how each object is unwrapped and painted step by step, but you will gain a thorough understanding of the process. Finally, you will learn how to produce a test render with Blender Internal. This will be a good transition to the next chapter. So let's dive into the texture creation process with Blender!

In this chapter, we will cover the following topics:

- Learning the different ways to project UVs on an object
- Painting your model with the Texture Paint tool
- Creating hand-painted and tile-able textures
- Baking diffused textures on proper UVs
- Making transparent textures
- Creating a draft render in Blender Internal

Unwrapping UVs

We will now cover the UV unwrapping process that will later allow us to add textures to our objects.

Using Project From View

In order to quickly unwrap the UVs of the walls of the house, the shape and size of the wall still being accurate, we can use the **Project From View** method.

For this to work, we need to align the walls on the axis of the world coordinates (X, Y, Z). Indeed, we are going to make use of the angle of the 3D View of Blender to project the UVs. So, for each wall, we are going to rotate our view so that the wall that we are going to project is flat. This method only works for objects that are flat and aligned, so don't use it for organic shapes:

1. We will enter the Orthographic mode (5) and then enter the view that corresponds to the face that we want to unwrap. For instance, in order to unwrap the UV of the front wall we go into the front view (1).
2. In the **Edit Mode** and in the **Face Mode**, we will select all the polygons of the 3D model that we want to unwrap and be careful with the beveled parts.
3. In the **UV Mapping** menu (*U*), we will select the **Project From View** option.

We will obtain a perfect UV Island that is well proportioned according to the mesh. Usually, we don't need to tweak UVs with this method, except when we are willing to scale them. To check the size of our UVs, we will use a UV grid texture.

4. We will create a new editor by selecting **Split Area**, then we will go to **UV/Image Editor**.

5. We will need to create a new image. So we will chose **Image | New Image** and **Generated Type | UV Grid**.

This UV grid allows us to adjust the island's size at the uniform scale for all walls of the house.

We will use **Project From View** for all the walls as well as the big flattened surface such as the platform of the terrace. For the later one, we will avoid having seams that are too visible:

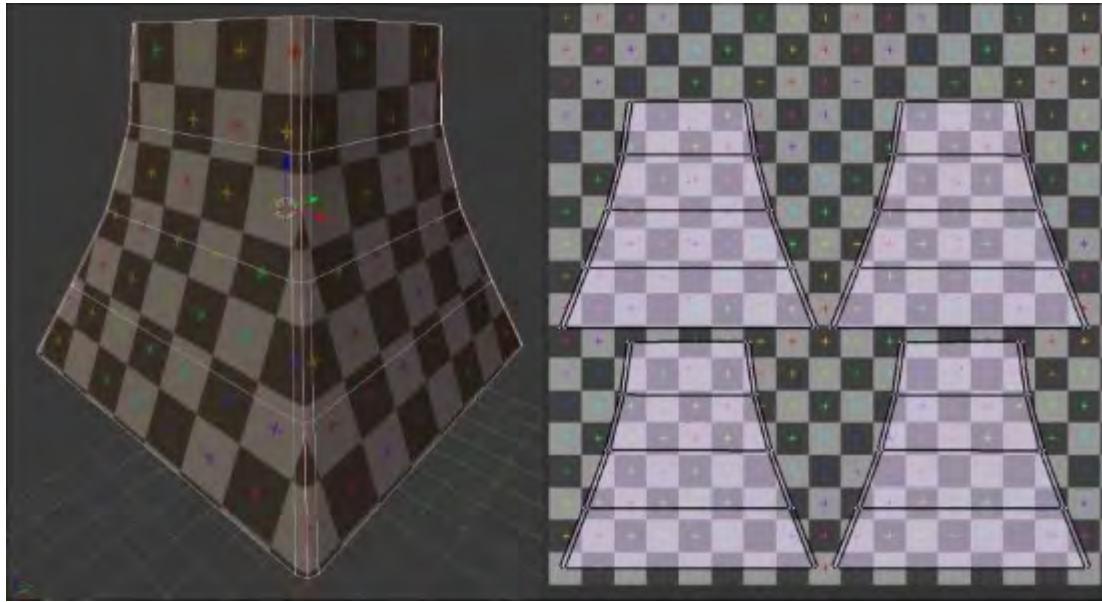
1. We will select all the top faces with the bevel part. We will then go into the top view (7), and we will do a **Project From View** unwrap.
2. Still being in the **Project From View**, we will select and unwrap the polygons of the sides, one after the other.
3. To make some seams invisible, we need to weld the vertices of some of the side islands of the platform with the central island. In order to weld the vertices in the **UV/Image Editor**, we will select them and press **W | Weld**. This acts like the merge tool in the 3D Viewport but in the UV space.

Unwrapping the rest of the house

For the other parts of the house—that is, the front door, windows, fireplaces, roofs, staircase, columns, bars, railing, and other little objects that form the house—we will place the seams to demarcate the UV islands like we did before with the Alien Character – Creating a Proper Topology and Transferring the Sculpt Details in [Chapter 4, Alien Character – Creating a Proper Topology and Transferring the Sculpt Details](#). We keep using the UV Grid to check the scale.

We will start with a pretty simple object, that is, the roof of the front block of the house:

1. We will place the seams to delimit the four sides evenly. We will select the desired edges with a right-click, and then we select **Mark Seam** in the **Edges Menu** (**Ctrl + E**).
2. We will make an automatic unwrap (press **U** and select **Unwrap**) that allows us to reveal the UV without any deformation, but to check this we will use the **Stretch** option in the right panel (press **N** and select **Display | UVs | Stretch**). You will recall that blue means there are no noteworthy deformations.
3. We must keep a little space between the different UV islands. Indeed, we will have a small gap that exceeds the islands later with the baking process. We will need to provide sufficient space in order to not see the seams on the 3D mesh displaying the textures.
4. We will reduce our UV islands and try to keep the same scale for the walls and the platform of the terrace. The position is not important for the moment.



We will continue with a slightly more complex object: the bay window.

You may have observed that the object has formed recurrences. It is angular. So we will avoid laying the seams directly on steep angles and take advantage of the beveled geometry if possible. We will also try to get a UV continuity on the window frame, the lower part and the upper part of the window frame. We must, therefore, make some adjustments after the UV unwrap.

The goal by doing the UV is to try to reproduce a flattened shape of the 3D mesh with the least distortion possible. This is done as follows:

1. We must place the seams at the roof level of the bay window. We will only use seams on this part of the object. We will cut the roof into three parts and three others for the bottom part.
2. As we have detached the other parts of the bay window before, we don't need to place the seams. If, in your case, all the faces are welded, take the time to place the seams.
3. We will select the entire object, and we will perform an **Unwrap** operation (press **U** and select **Unwrap**). This gives us seven UV islands that we can place in a corner for the moment.

We don't have to select all the faces of an object to unwrap the UVs; we can select and unwrap only a few selected faces, but we will lose the scale based on the other faces. This forces us to make some adjustments.



1. We will check whether there are deformations with the **Stretch** option (press *N* and under **Display** select **UVs and Stretch**).
2. Now we will align the edges that need to be like the window frame. We will select the edges that must be vertically aligned, and we will scale them on the *x* axis (use the *S*, *X*, and *O* shortcuts).
3. Similarly for the edges that need to be horizontally aligned, we will select them and we will scale them on the *y* axis (use the *S*, *Y*, and *O* shortcuts).
4. Once the object is aligned, we will place a UV Grid texture to check the scale.

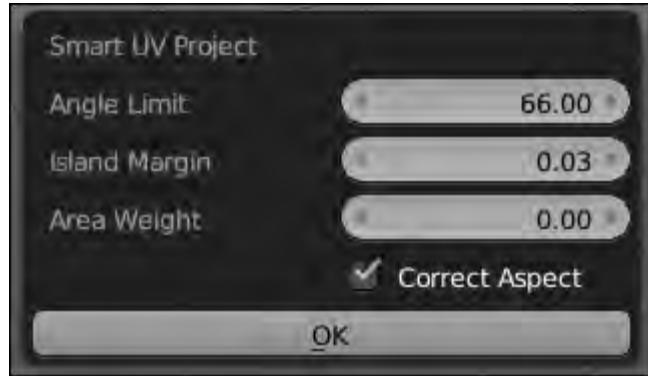
Aligning the edges is a process that can be long but it is important to do this for objects such as pipes or angular structures. There are add-ons that can save us some time, such as **Quad Unwrap**.

In this way, we will continue unwrapping the UVs of other objects that compose the house.

The tree with the Smart UV Project

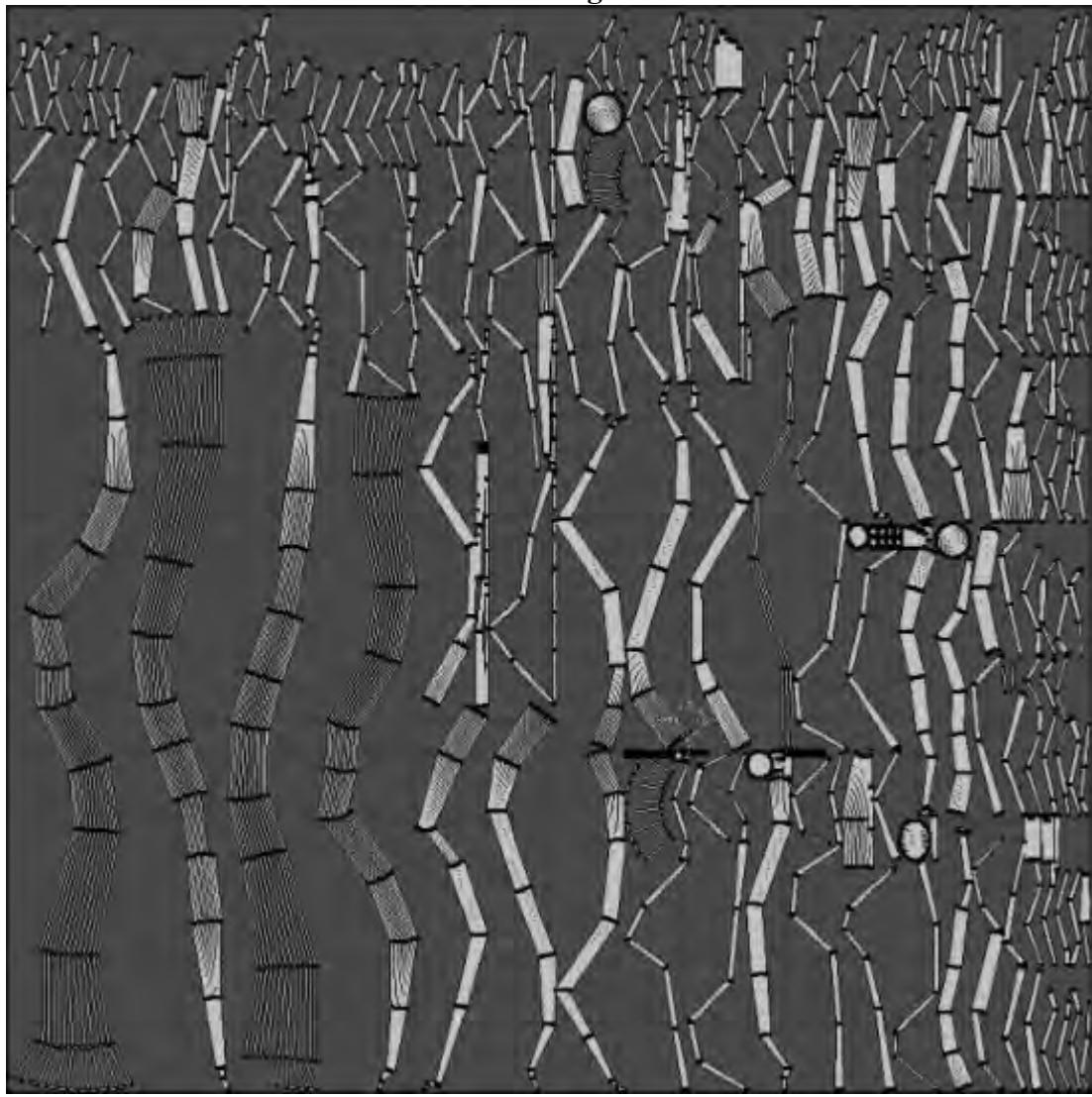
We will now learn a new unwrapping technique with the tree. The tree will be quite far in the scene, so we don't need to have perfect UVs on it. This is why we are going to use an automatic method that Blender provides called the **Smart UV Project**. This is done as follows:

1. In order to unwrap an object with the **Smart UV Project** method, we don't have to put seams on it. So, in order to UV unwrap our tree, we will need to select it entirely in the **Edit Mode**, press *U*, and select **Smart UV Project**.
2. As you can see, we now have the ability to tweak some options. The most useful is **Island Margin**, which allows us to choose how much space there is between each UV Island.



The Smart UV Project options

3. Now we can see our new UVs in the **UV/Image Editor**.



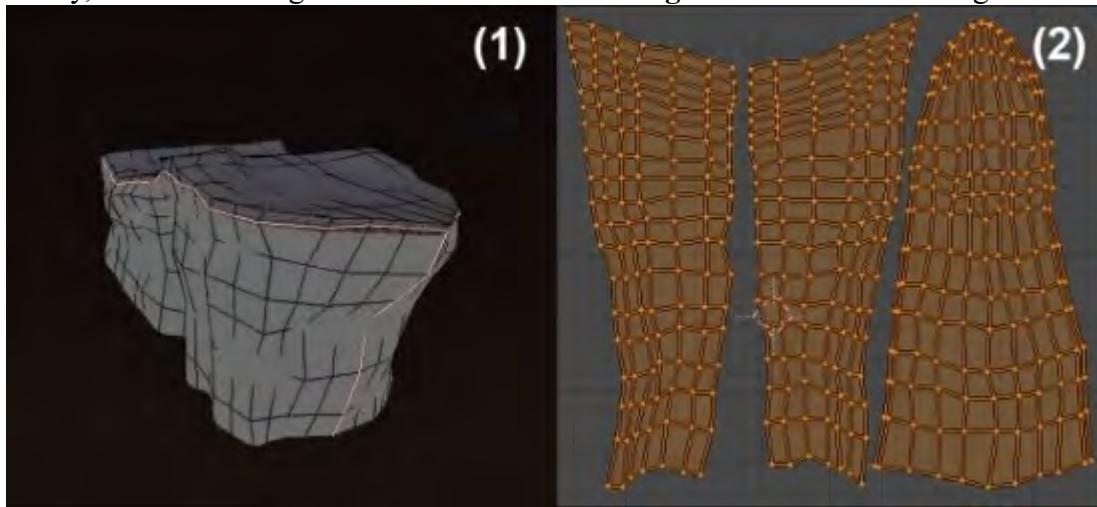
The Smart UVs of the tree

This method is quite useful but not very accurate. It generates a lot of seams that can cause visual artifacts. This is why it is only interesting for objects that are far away, small, blurred (in the depth of field), or with a small number of polygons. In the case of our tree, the wood texture will also be quite repetitive, so this will do the trick!

Unwrapping the rest of the environment

There are other methods to unwrap objects such as cube, sphere, or cylinder projections, but in our case, they won't be very useful. In order to unwrap the rest of the environment, we will use techniques that we have already seen in the Alien project. If you don't remember how to put seams on an object, we encourage you to read [Chapter 4, Alien Character – Creating a Proper Topology and Transferring the Sculpt Details](#), again. To refresh our mind, we will just do the cliff together as follows:

1. The cliff will be separated into three islands. The first seam separates the top where the ground will be from the lower section. We simply select a horizontal edge loop near the top and mark it as a seam (**Ctrl + E** and select **Mark seam**).
2. We will now separate the sides into two islands by placing a vertical seam from the front tip to the bottom of the cliff.
3. We will then select our whole object, press **U**, and select **Unwrap**.
4. Lastly, we can rearrange our islands in the **UV/Image Editor** in order to align them vertically.



The seams and the UVs of the cliff

Now that you are nearly a pro, you can unwrap the rock and the barrier alone by following the same method.

Tiling UVs

Now comes the interesting part! We will show you a very useful technique in order to save time with the texturing process, so we won't have to paint our objects entirely. So let's learn a little bit more about tiling.

What is tiling for?

The main goal of tiling is to allow a texture to repeat itself on the mesh. For instance, in the case of a very large wall, it would be very tedious to paint each brick one by one over the entire wall. So what is preferable to do is to use a "tileable" texture and scale the UV islands. Scaling the islands will simply repeat the texture. Later, we will show you how to paint a tileable texture within Blender. If you remember the UV unwrapping part of the alien project, we had mentioned that all the islands will be placed in the UV finite space. This was because we didn't want repeated textures. But in this project, we can use this to our advantage. But it doesn't mean that we won't keep our UV islands in the finite UV space too. This is why we are going to need different UV coordinates on each object.

The UV layers

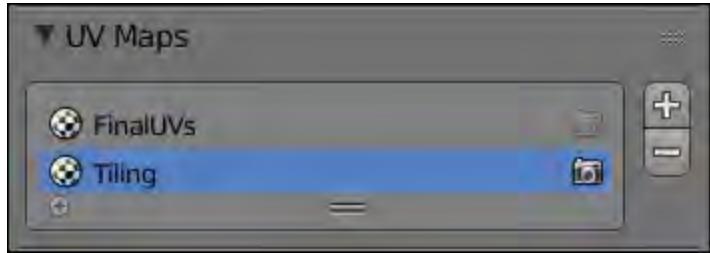
As we said before, we need to find a way to keep our proper UV (in the UV "square") and create a new set of UVs. Blender allows us to use multiple UV layers on each object. We can use a layer when we want to have another UV layout while maintaining the UV that we've already done before. It is similar to storing different UVs on the same object. We will show you how to create a new layer that will contain the scaled islands for tiling. As this will be the same for many objects, we will just show you the method for the wall of the house, and let you repeat the process for the rest of the objects:

1. We will select the wall and enter **Edit Mode**.
2. We will then click on the **Object Data** icon in the **Properties** editor, and under the **UV Maps** subpanel, we will click on the + (plus) icon. This will create a copy of the current UV set of our object.
3. We will now need to change the UVs on this particular layer. Be sure that it is selected, and in the **UV/Image Editor**, scale the current UVs with the **S** key. You can always check the tiling effect with a checker. Our wall has now two sets of UVs.

Note

The UV layers options

There are very few options to manage our different sets of UVs. The first one is the little camera icon that tells Blender which UV set should be used at render time and the second one is the ability to rename a set by double clicking on its name. The only thing that you need to think about is which set to select.



Now that you know the technique, you can add a **Tiling** UV set for each object.

Adding colors

Now that we have the proper UVs on our objects, let's dive into the fun part, that is, the texturing. It is the more artistic part of the process, so let's start by discovering the Texture Paint tool of Blender.

Basics of the Texture Paint tool

The **Texture Paint** tool is a mode that allows you to paint directly on a 3D object in the 3D Viewport while applying color to the texture. This requires having textures with a sufficiently high resolution. One of the interesting points is to paint on a 3D polygonal mesh with a low density.

To observe the paint of our textures in 2D, we need to split the 3D Viewport in two and switch the second type editor to **UV/Image Editor**.

To activate this, we first select an object, click on the **Mode** drop-down menu in the **Header**, and switch to the **Texture Paint** mode.

If you don't have any UVs, a message will warn you in the left panel (*T*) of the 3D Viewport. You can generate UVs automatically with the **Add Simple UVs** option, but it is much better to unwrap them yourself as we saw earlier.

In the **Slots** tab, there is an important parameter, that is, the **Painting Mode**. It gives a choice between two options. The **Material** option allows us to paint automatically linked textures to a material in Blender Internal. The **Image** option allows us to paint the texture without necessarily having a material linked to the object. For this first approach of the Texture Paint tool, we will be especially interested in the **Material** option.

If in **Texture viewport** Shading Mode (*Z*) your object displays a pink color and you see the message **Missing Data** in the left panel (*T*). Select **Tools**, to correct this; you will need to click on the **Add Paint Slot** option. Here, several texture types are available. This will automatically create a texture corresponding to a slot of the material with the required settings during the painting phase.

We can start testing a **Diffuse Color** map. Several options are proposed. They are the same as when we create a new texture. You can rename the texture and choose the height, the width, and the color with an alpha value. You can also choose whether you want an alpha layer (it is the opacity), the type of texture to generate, and finally, the 32-bit float option. Press **OK** to create this texture. A new material is then automatically created if there are none of them. You can visualize it in the **Material** editor on the right-hand side of the work space.

To modify this, you can change the name with a double left-click on the name.

It is possible to create several stacked textures one above the other like layers in the material. You must select the one that you want to paint in the **Slots** tab of the left panel. The bottom slot is the one that is first visible. You can also choose the Blend Type to mix pixels. There are the usual Blend Types (add, subtract, multiply, and so on.) that we can find in every decent image editing software. The **Slots** tab allows us to also change the UV layers, which can be very useful.

Now that you know the basics to generate and manage a texture for painting, we will look at the brushes.

Discovering the brushes

As in the **Sculpt Mode** that we saw previously, we have multiple brushes in order to paint our texture. They all have some specific purpose that we will test on a simple sphere object in a new scene file. Be aware that the goal here is not to do something beautiful but to test our brushes.

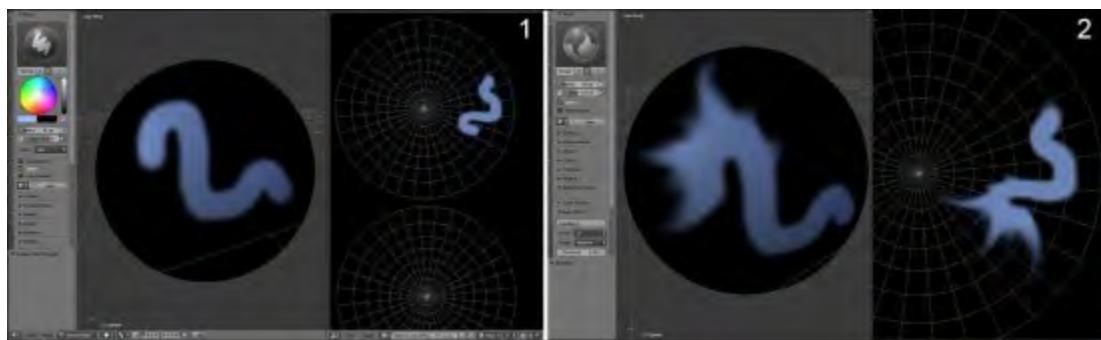
The TexDraw brush

This is the brush that allows us to paint the desired color in a localized manner.

You can use the blender mode in order to create effects. For instance, the **Add** mode is very useful for lighting texturing effect (refer to **1** on the following screenshot).

The Smear brush

The Smear brush allows us to move the color while blurring it. It is very useful to create some blown or flame painting effects. If you change the strength parameter to a higher value, you can stretch your paint to a higher distance (refer to **2** on the following screenshot).

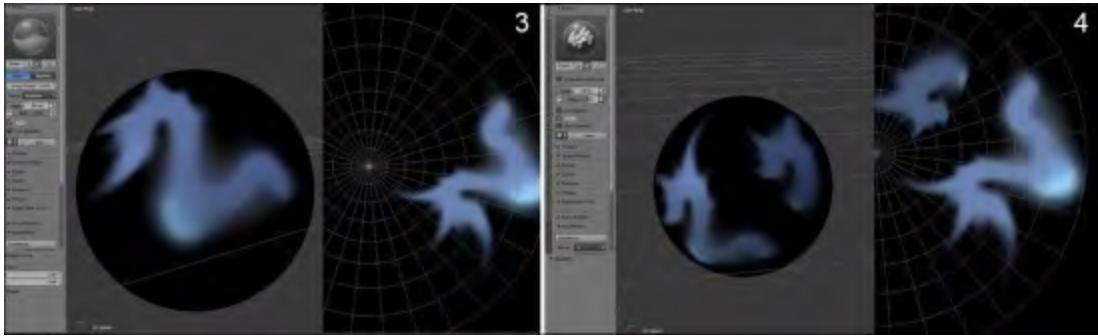


The Soften brush

This brush allows us to blur the painting. It is useful to mix the colors and create gradients (refer to **3** in the following screenshot).

The Clone brush

This brush allows you to copy a specific zone on another place. This is very useful when you need to fill some untextured space or when you want to correct the seams. You select the zone that you want to copy by placing the 3D cursor on it with *Ctrl* and LMB (refer to **4** in the following screenshot).

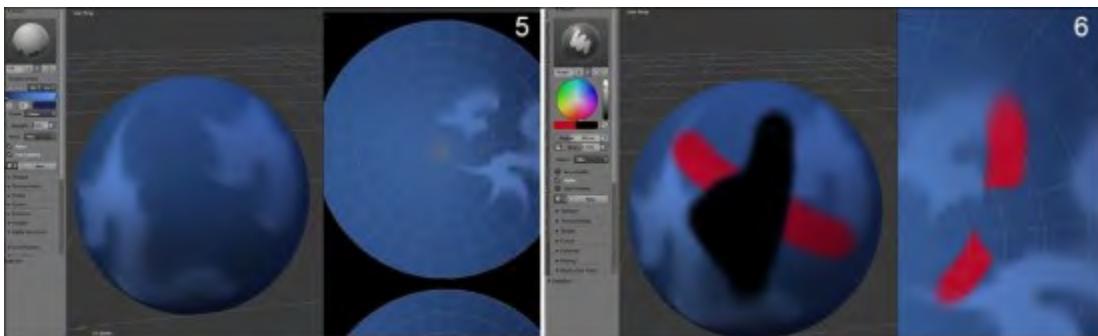


The Fill brush

This is a new brush that Blender has had since version 2.72. This brush allows us to fill the whole object with the selected color. With the **Use Gradient** option, you can do a gradient that stretches over the whole object. Remember to set the strength parameter to **1** to have a sufficient opacity. A line under the mouse cursor will inform you where the start and the end of the gradient will be. You can also use the **Multiply Blend** mode while using it (refer to **5** in the following screenshot).

The Mask brush

As with the **Sculpt Mode**, it is possible to mark a zone that you want to avoid painting. To do this, you will create a stencil image. Don't worry, Blender will ask you to create the image as soon as you create a mask, if it can't find one. You only need to click on the **New** button or select a preexisting image in the .blend file and validate the image settings like we are used to. To clear a masked part, press *Ctrl* and LMB. To remove your mask, you can remove the mask option in the **Slot** tab. Be aware that the masks are not visible in **Material Viewport Shading Mode** (refer to **6** in the following screenshot).



If you have a pen tablet, you can check out the small button on the right of the radius and strength parameters (an icon with a hand). This allows you to vary the amplitude of the parameter according to the pressure sensitivity of your stylus.

The Stroke option

The **Stroke** option allows us to completely modify the brushes' behavior. It is, therefore, important to focus on this for a little while.

First of all, there is the **Stroke Method** option that allows us to choose among several methods for applying the colors:

- **Space**: This is the basic method with a variable dot space.
- **Curve**: This is a new method since Blender 2.72 that allows us to paint in a well-defined curve with controllable points. Press *Ctrl* and left-click to create the points defining the curve. Each point can be controlled with the transform tools: Grab (*G*), Scale (*S*), and Rotate (*R*). In order to apply your painting, you need to press *Return*.
- **Line**: This method simply allows you to draw lines. You must do a pushed left-click to draw the line from one point to another. The paint is then projected onto the 3D mesh.
- **Anchored**: This allows you to drag your stroke. You first need to select the placement of the stroke or the texture you want to paint, and then, without releasing the mouse, you will be able to control its scale. This method is especially interesting when projecting a texture.
- **Airbrush**: This method could be used to project a multitude of little spots, for instance, you can change the radius so that it is smaller with **Rate** of **0.10** and **Jitter** of **2**. It is useful to create skin textures, for example.
- **Drag Dots**: This allows you to paint points or spots by placing them one by one.
- **Dots**: With a **Jitter** parameter at **0**, you get a textured line to paint. With a **Jitter** parameter at **1**, you will get a multitude of spots.

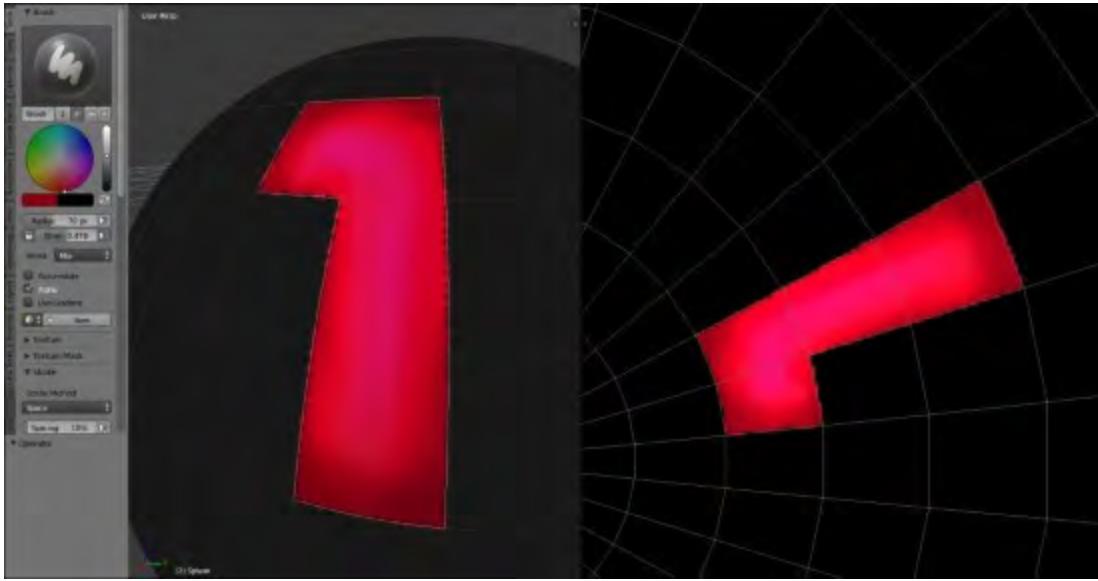
There is another key element that will determine the settings of your brush. It is the curve located just below the **Stroke** tab. It works exactly in the same manner as the **Sculpt** mode that we saw previously. Depending on whether you want a hard or thin brush to paint the details, remember to use and test several curve profiles. There are already several predefined shapes that can meet your needs.

Delimiting the zones of painting according to the geometry

So that we paint in a precise manner, it is possible to limit the zone that we want to paint by selecting polygons.

After you have selected the desired faces in the **Edit Mode**, you can go to **Texture Paint** and check the **Face Selection Masking for Painting** button on the left-hand side of the layers in the 3D View header. The icon shows a small cube with a checker pattern on a side.

You can now paint without fear of overflow.



Painting directly on the texture

If for any reason you have difficulties when painting directly on the mesh in the 3D View, you can also paint on the texture.

You simply need to select your texture in the **UV/Image Editor**, click on the **Mode** drop-down menu, and choose **Paint**. The **View Mode** is the one by default.

You have all the painting tools that you already know in the left panel (*N*). For your comfort, you can always set your view in full screen with *Shift + Space*.

Painting the scene

We are now ready to apply what we've learned previously about the **Texture Paint** tool on our haunted house. Let's start!

Laying down the colors

For any image that uses colors, it is necessary to lay down a color palette. This means that we will need to find the colors that will make up our image. In our case, for the house, we have chosen the following color codes:

- R: 0.259 – G: 0.208 – B: 0.149
- R: 0.180 – G: 0.141 – B: 0.102
- R: 0.149 – G: 0.102 – B: 0.082
- R: 0.337 – G: 0.318 – B: 0.310
- R: 0.188 – G: 0.145 – B: 0.055
- R: 0.251 – G: 0.192 – B: 0.075
- R: 0.780 – G: 0.596 – B: 0.231

- R: 0.212 – G: 0.267 – B: 0.373
- R: 0.176 – G: 0.063 – B: 0.067

We have the ability to create a color palette by clicking on the + (plus) button near the color wheel. However, in order to have an idea of the whole color scheme, we will start by fulfilling the 3D mesh that we had unwrapped object with the colors. This is done as follows:

1. We will select one of our objects.
2. We will split our screen into two, and we will open **UV/Image Editor**, if it's not already the case.
3. In the **Edit Mode**, we will move the UV in a corner. We will keep a space for a margin, and we will not place our UVs too close to the border.
4. We will create a new diffuse map with a resolution of 4096x4096.
5. With the Fill brush, we will apply the corresponding base color for the object.
6. We will select another 3D mesh.
7. In the **Edit Mode**, we will select **Diffuse Map** for this mesh, and we will move the UV near the UVs of the previous mesh. This will be easy as you will see the previous fill.
8. Again, with the fill brush, we will apply our color for this new object.

We redo these steps for all the objects. Since our objects' UVs are proportionally scaled, their size should be sufficient in order to place the maximum number of objects on the same diffuse map.

Be careful to not select the tiling layer for the UVs while filling your objects.

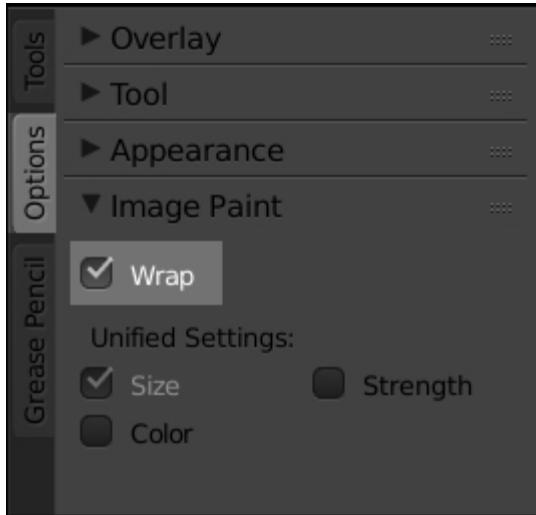
Tiled textures

It's now time to take advantage of our tiled UVs by painting our own tiled textures by hand! In this section we are going to show you how to create the roof texture step by step, and as the process will be very similar for the wood plank, ground, brick wall, and rock texture, we will only give you some advice in order to get a nice result. So let's get started by setting up our painting environment.

The settings of our workspace

One of the strengths of the Blender painting tool is to be able to paint in the **UV/Image Editor** in such a way that the strokes that you paint repeat themselves on the borders.

1. We will first open the **UV/Image Editor**, and make it full screen by pressing *Ctrl + Space* while hovering over it. After that we will create a new 1024 x 1024 Texture with a greyish dark blue color (**Image | New Image**).
2. We now need to enter the **Paint** mode by choosing it in the header drop-down menu. By default the mode is set to **View**.
3. Now we will activate the **Wrap** option in the **Option** tab of the **UV/Image Editor (T)**. This allows our stroke to repeat to the other side while painting on an edge.
4. Another nice feature is to check the **Repeat** option located in the Right panel (**N**) under the **Display** subpanel. It allows us to see the tiling effect.
5. Let's try this by painting on our texture with the basic brush and a different color. As you can see our strokes are repeated and we can see the tiling effect! Press *Ctrl + Z* to undo your testing strokes.



The **Wrap** option in the **Options** tab

Advice for a good tiled texture

Before starting the painting of our roof texture, we will give you some good advice that can lead you with a nice tiled texture. We first need to remember that the goal of a tiled texture is to give the impression of a pattern that repeats on a surface but in real life, even with a perfect wall pattern for instance, we can see differences between each brick. That's why we need to have a pretty homogenous texture.

We will need to balance the contrast of our tints so they don't disturb our eyes after the tiling. Another important thing to remember is that the pattern should be repetitive in some way. We cannot paint a computer keyboard texture in a tileable manner for instance, because the keys are not the same size and don't contain the same letters. But it can work with a lot of things such as a brick wall, concrete, wood, and so on. We also need to think about the scale of the elements that compose the pattern. For instance, in the case of our roof tiles, we don't want to have one that is very small compared to the others; it will break the illusion of repetition. So now that we know the pitfalls of the tiled texture art we can start working on our roof-tiled texture.

Painting the roof-tile texture

Let's start our roof tile texture from the texture that we've created in the **UV/Image Editor** in the previous section.

1. Before starting to paint our texture we will change our curve to be a little bit pointier. We can easily select a curve preset in the **Curve** subpanel.
2. The first thing that we need to lay down is the tile pattern. In order to trace that we will use the same tint as the background color that we chose when creating the texture but darker. We then re-size our brush size (*F*) and start to paint a row of 'U' shapes for the first top tiles. We need to space them proportionally according to the size of our texture. For the next rows we will do the

same thing with a little offset. The top-left part of the 'U' shape needs to touch the middle of the above ones. Note that if you are lost while having the repeat option activated, you can always help yourself with small helping markers that you can erase later (refer to **1** in the following screenshot).

3. Now that we have the basic pattern drawn, we can start to add a little bit more detail with the shadows. For this we are going to select a darker color, but still in the blue shades. Because shadows will be faded, we are going to decrease our strength a little bit (*Shift + F*). Here, the shadows that we are going to paint are projected because of the tiles that are placed above. This will act like contact shadows (refer to **2** in the following screenshot).
4. Now that we have our shadows we can start to add some scratches on the tip of each tile. You need to remember the fact that it needs to be as homogeneous as possible, so don't paint big scratches or it will break the tiling effect (refer to **2** in the following screenshot).
5. The last thing that we are lacking here is some highlights. When painting highlights we use a white color and decrease our strength and size. We then slightly paint the highlights where we think we have hard edges. For instance we can emphasize the scratches (refer to **2** in the following screenshot).
6. That's all, you've now completed your first hand-painted texture (refer to **3** in the following screenshot).



Steps for the roof tile texture creation

Quick tips for other kinds of hand-painted tiled textures

We aren't going to show you step by step how to do each tiled texture as it would require a lot of space and it would be a very repetitive task. Indeed when we are doing such a texture we first always create a texture with a flat color, and then lay down the pattern with a darker color. Once we are satisfied with the pattern and the way it tiles, we add the shadows, more details, and finally the highlights (the specularity).

As you can see on the wood texture, it is quite difficult to add the ribs and having a good tiling, so later we will need to take this problem into account on the objects that will receive the texture. But we can't add ribs on wood or it will look strange. For the ground we can add a little bit more detail, such as small rocks and crackles. The bricks are quite easy to do, but if you feel you can add more detail, you can easily paint moss between each brick.



Examples of other tiled textures painted in the UV/Image Editor

Baking our tiled textures

We are now going to project our tiled textures on other textures that correspond to the UVs of our different objects.

Why bake?

As we saw it with the Alien character, texture baking is very useful in order to capture relief, shadow, or color information. In the case of our haunted house, we are going to capture the color information of the tiled textures in order to have them on a large texture with the proper UVs. This lets us achieve our tiled patterns on one big map in order to add all the tweaks that we want later on. We could for instance paint the window contact shadows, add some grunge, and age our objects.

In our scene we aren't going to bake everything. So some objects are still going to use their tiling UV layer. It will simplify our work and still leave us with a nice result.

How to do it?

To obtain a successful bake, the manipulations will be quite similar to what we've done with the normal and ambient occlusion map of the alien. We will start by doing the baking of the walls.

1. We select our walls joined as one object and we go into **Edit Mode**.
2. In the UV Map subpanel under the **Data** tab we click on the camera icon on the right of the Tiling layer. This will tell Blender to use this layer for the render and baking process. We then select the first layer in order to project the details on it with the proper UVs that are normalized.
3. Still being in **Edit Mode**, we create a new map that we call **Color_walls_01** with a 4096 x 4096 resolution. We also un-check the **Alpha** checkbox. This image will contain the result of our bake.
4. We now go to the **Bake** tab from the **Render** tab.
5. Under the **Bake** button we select **Texture** as the **Bake Mode**.
6. In the **Margin** option we choose **10px**.
7. Repeat those steps with the objects that share the same UV space.
8. We can now click on **Bake** in order to start the process. Voilà! Your baked texture is now ready to be placed as a diffuse color texture on a material.

Creating transparent textures

One thing we haven't learnt until now is how we can produce texture with an alpha channel. Indeed this could be very useful in order to add some details on the previously baked texture (grunge or leaks, for instance) or even grass.

The grass texture

Usually, when doing grass, fur, or hair, we use the integrated particle hair system of Blender, but in our case we will show you a technique that can do the job as well and can save us render time. It will also accommodate very well with the style of our scene. This technique will simply consist of a plane mesh on which some grass strands will be projected; using the alpha we will be able to just render the strands. Note that this is a very common technique in the video game industry. So let's start our grass texture by first setting up our transparent texture!

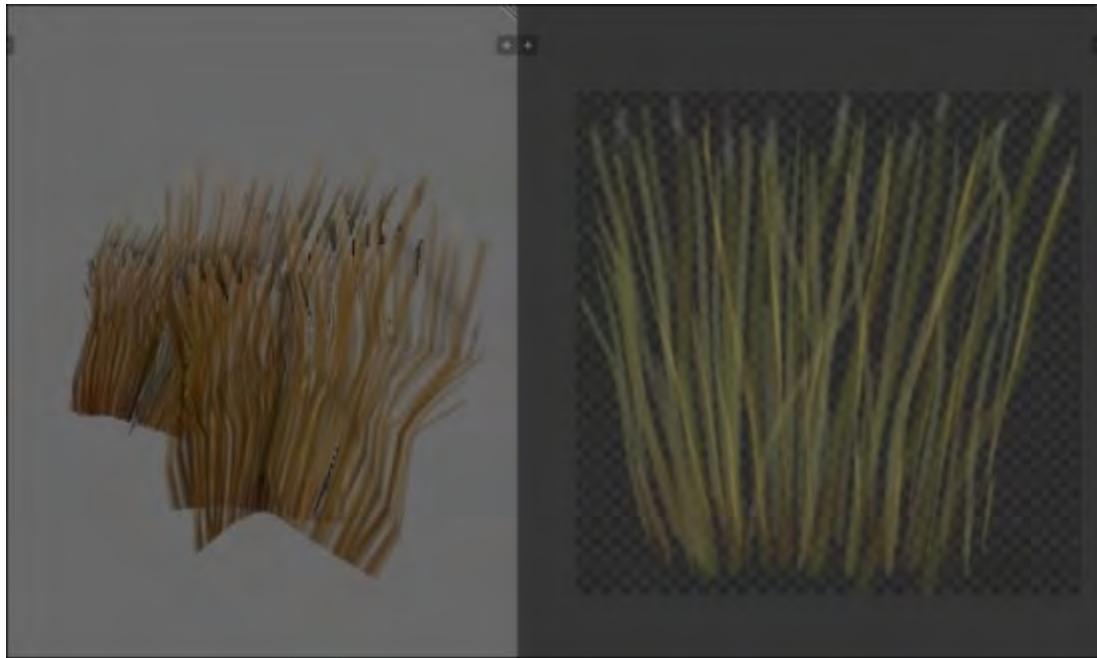
1. We will first go in the **UV/Image Editor** and create a new texture. In the color setting of the texture, we will change the alpha channel to **0** in order to have a full transparent image. We then leave the 1024 x 1024 resolution and validate our settings.



The grass texture settings

2. We can now use a pointy curve and start to paint some strands starting from the bottom of the texture to the top with a de-saturated green. We really need to think about painting a dense grass mound.
3. In order to add more realism to our grass texture, we can add some touch of yellow. We also need to add some white on the tip of each strand. It is quite important to use a reference when painting some textures; it helps to develop our sense of perspective and come with more believable results.
4. Remember to save your texture on your hard drive or you will lose it (**Image | Save as image**).
5. We can now place our texture on a new plane (**Shift + A**). We do a quick UV on it by simply pressing **U** and selecting **Unwrap** in the **Edit Mode**.

6. We will now create a new material that will use our texture. To do that, go to the **Material** icon in the **Properties** editor and press the plus icon. If you already have a default material, you can delete it with the minus icon.
7. So our material can understand the alpha channel, we will have to activate the check box of the **Transparency** subpanel and select the **Z-Transparency** mode with its Alpha value set to **0**.
8. Now we will tell our material to use our grass texture. To do this we click on the texture icon of the **Properties** editor and click on the first available texture slot. Under the **Image** subpanel we click on the far left drop-down menu and select our grass texture. The last thing we need to do is to activate and set the **Alpha** slider under the **Influence** subpanel to **1.0**.
9. We can have a preview of our texture in the 3D Viewport by activating the GLSL mode in the right panel of the 3D View (*N*) under **Shading**. Note that you will need to be in the **Texture** Shading mode (located under the **Viewport shading** drop-down menu in the 3D View header) and you also need to have lights.
10. We can now duplicate the plane object as an instance to have more grass. Note that you can also add a subdivision to the plane and in the last tool options you can change the **Fractal** slider in order to add a little bit of randomness. Remember that the render in the viewport is a preview, not the final render.



*The final grass texture in the viewport (left) and in the **UV/Image Editor** (right)*

Note

More about the color wheel window

When selecting a color in Blender we have many options. You can of course select the color that you want with the color circle or by changing the slider's values. In **RGB Mode** we can act on each red, green, and blue component plus on the Alpha channel. In **HSV Mode** we can change the hue (the tint),

the saturation, and the value of the color. If you put the saturation down to **0** the color will be on a gray scale. The **Hex Mode** allows you to type a hexadecimal value such as FFFFFF (white) or FF0000 (red). Hexadecimal simply means that instead of counting from 0 to 9 we count from 0 to F. It represents 16 possible values. The easy thing to remember when dealing with hexadecimal colors is that the first two digits represent the Red value, the next two digits represent the green, and the last two represent the Blue: RR GG BB. FF is the full color, 00 means no color. For instance 00FF00 is full green.



The grunge texture

The grunge texture will be useful in order to add details on the wall texture of the house. The technical process is the same as the grass texture. For the painting we simply use a dark brown color and paint some vertical leaks from the top to the middle of the texture.

Now we can stamp this texture on our wall.

1. We select the wall and ensure that its baked texture is selected in the **UV/Image Editor** while being in **Edit Mode**. Another thing you may want to do if you are still in **GLSL** mode is to create a new material with the baked texture set.
2. In order to paint our leaks we will use the **Anchored** stroke method located in the **Stroke** subpanel. It allows us to precisely place our leaks near the top and the bottom of the wall.



*The grunge placed on the house in the viewport (on the left) and the grunge in the **UV/Image Editor** (on the right)*

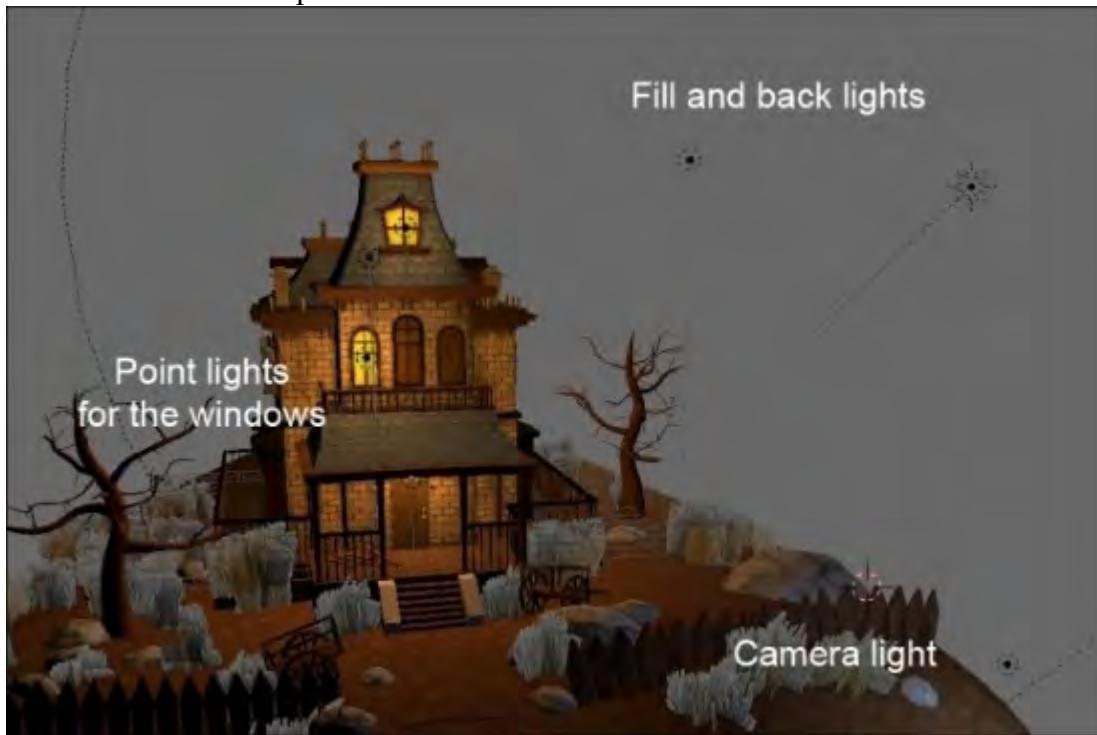
Doing a quick render with Blender Internal

Welcome to the bonus section! Here we are going to do a quick render with the Blender Internal render engine to get an idea of what the whole scene will look like. Note that this is totally optional as we are going to create a render with cycles later. In order to do our render, we will need to add a material for each object in the scene like we did for the walls but with their corresponding textures.

Setting lights

Now that we have all our materials created we will need to turn up the lights! Of course if you don't have light you won't see anything like in the real world.

1. We add a sun lamp (press **Shift + A** and select **Lamp | Sun**) and change its color to a grayish yellow in the light settings situated in the **Properties** editor. We leave its energy to a value of **1**. We place it behind the house and rotate it so it hits the back of the house.
2. The next light we will add is point light. This one is going to be much more intense so we bump up the energy value to **20** and also change its color to a light yellow. This one will fill the scene a little bit more.
3. Now we need to add lights behind the house to fake the lighting of the windows. In total we added three yellowish point lights with a value of **10** for their energy.
4. The last point light we can add will be near the camera, so we would have to place it correctly after choosing our point of view. This light allows us to see a little bit more of the close environment.
5. If you want to see the lighting effect in the viewport, it is best to turn off the **Texture Shading** mode and the **GLSL** option.



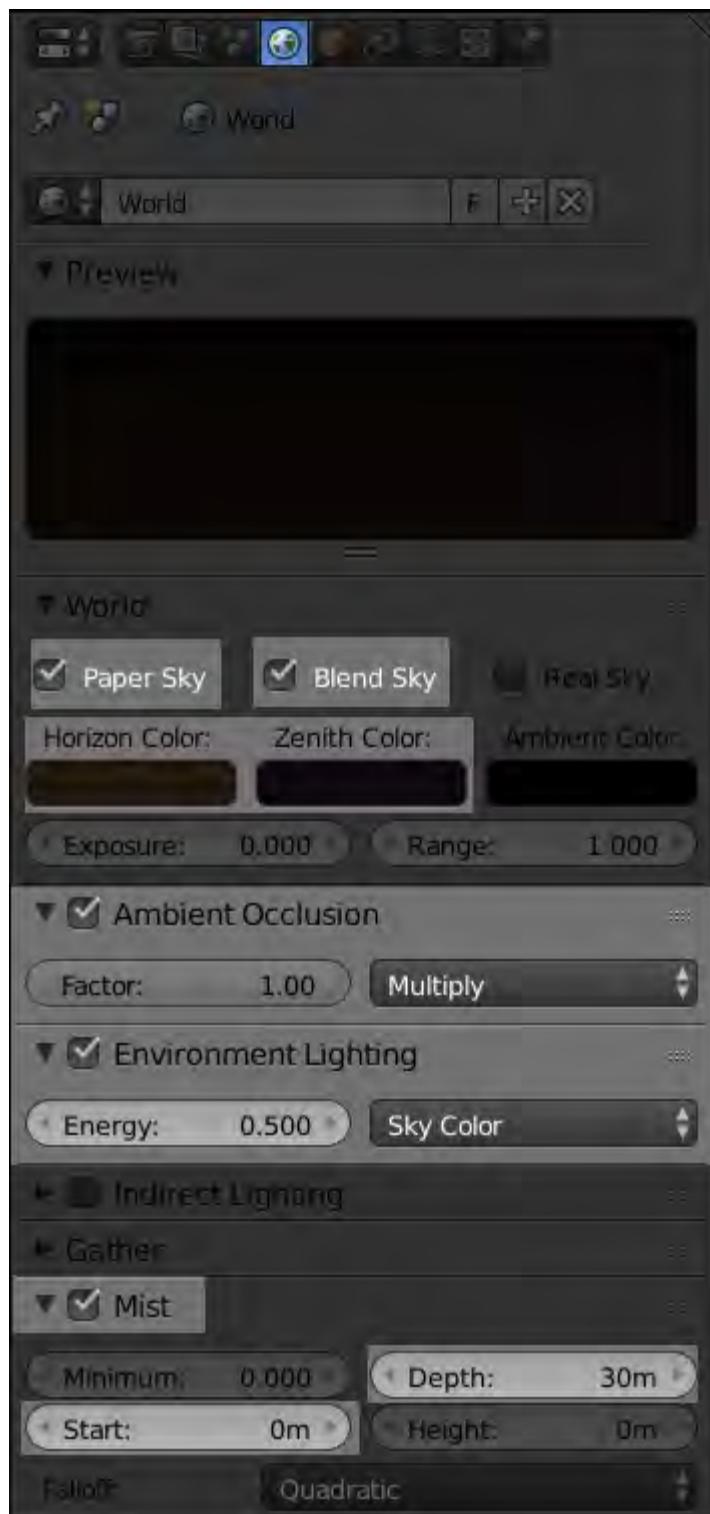
Placing the camera

We can now choose our point of view, by moving our camera. If you don't have a camera, you can add one (press *Shift + A* and select **Camera**). You can also add many cameras and switch between them by selecting the one you want to look through and pressing *Ctrl + P*. In order to look through your camera in the viewport you can press the *0* numpad key. Usually it's a good habit to change the focal length of your camera in the Camera setting tab in the **Properties** editor (click on the Camera icon). In our case we wanted a fleeting camera so we set a focal length of 20mm.

Setting the environment (sky and mist)

Now in order to improve our render we will change the sky color and add a mist.

1. To do that we go into the world settings in the **Properties** editor (click on the earth icon).
2. Under the World subpanel we check the **Paper sky** and **Blend sky** option; we change the **Horizon color** to a dark brown color and the **Zenith color** to an even darker color. It's not realistic, but this image is not intended to be realistic.
3. We can now check the **Ambient Occlusion** checkbox and set it in the **Multiply** mode.
4. The **Environment Lighting** option will serve to light the scene with the **Sky Color** option (and not **White**, the default option).
5. Lastly we can activate the **Mist** subpanel and change the **Depth** parameter to **30m**. We also set the start option to **0**. You may have to tweak those values in order to match your own scene.
6. Now you can press the **Render** button in the **Render** tab of the Properties editor or press *F12*. Note that if you want to improve your render with some automatic compositing you can go to the **Scene** tab of the **Properties** editor and under **Color Management** you can use a look preset; you won't have to re-render your scene! You can also tweak **Exposure**, **Gamma** and change the **CRGB** (Contrast, Red, Green and Blue) curve by clicking on the **Curve** checkbox. As you can see, your render is displayed in a **UV/Image Editor**, so you can save your image (**Image | Save Image as**). Congratulations, you've done your first render of the haunted house!



The world settings

The final haunted house after the Blender Internal render will look like the following screenshot:



Summary

In this chapter you completed the UVs and the textures of the scene. You have learned a technique to paint textures by hand with the Texture Paint tool of Blender and how to create and use tileable textures. You also learned more about baking. Now that we have introduced the rendering process with the Blender Internal render engine, we can start to learn more about the other render engine called Cycles, which has different approach. So let's dive into Cycles!

Chapter 6. Haunted House – Adding Materials and Lights in Cycles

This chapter will be devoted to the Cycles render engine. You will learn how to achieve a convincing render of the haunted house by understanding the different types of light work and by creating complex materials using the previously made textures. You will learn some nice tricks such as how to produce normal maps of our hand-painted textures without leaving Blender or how to create realistic-looking grass. You will also discover how to use the Cycles baking tool. In order to conclude our project, we will show you how to integrate a mist effect in the final composition.

In this chapter, we will cover the following topics:

- Understanding the essential settings of Cycles
- Using lights
- Painting and using an Image Base Lighting
- Creating basic materials with nodes
- Using procedural textures
- Baking textures in Cycles for real-time rendering

Understanding the basic settings of Cycles

To switch to the Cycles render engine, you must select it in the list of proposed engines that Blender offers in the menu bar. We will see in the first part of this chapter some of the very useful settings that should be known while using Cycles.

The sampling

If you directly try to make a render with Cycles without changing the parameters of Blender, you will certainly see some noise in the image. To make this less visible, one of the first things to do is change the sampling settings. Unlike Blender Internal, Cycles is a Raytracer Engine. While rendering, Cycles will send rays from the camera in order to generate pixels. The noise is due to a small amount of the samples. Cycles, therefore, needs more samples; the more sampling, the more accurate the final render.

The following sampling settings are in Properties editor. Just select **Render | Sampling**:

- **Render samples:** This is the number of samples for your renders. The more samples you add, the longer the rendering time will be.
- **Preview samples:** This is the number of samples Blender will calculate to preview your scene in the 3D viewport in Rendered Viewport Shading. A value between 20 and 50 samples is correct. This value depends on the performance of your computer.

The Render sample must be higher than the Preview sample.

Note

The GPU device

If you have a fairly recent CUDA®-compatible graphic card, you can opt for GPU rendering. This allows you to make renders very quickly and visualize your scene in the 3D viewport nearly in real time.

For this, go to **User Preferences** | **System** | **Computer Device** and select **CUDA**. Then, in **Properties**, go to **Render** | **Device** and select **GPU**.

Note that the most recent AMD GPU has been supported since Blender 2.75.

Clamp direct and indirect

This allows us to clamp the intensity of the rays of light launched from the camera. This can also help to reduce the noise effect, but it blurs the pixels together.

Light path settings

You will find the following light path settings in the menu:

- **Max and Min Bounces:** This is the number of minimum and maximum bounces a ray of light can do in the scene to render a pixel. This mimics the way photons bounce from objects in real life. The higher the value of the maximum bounce, the greater the precision will be, and the quality of the rendering will increase. A high value of minimum bounce will also improve the quality but may considerably increase the rendering time. It is advisable to set the same minimum and maximum value.
- **Filter Glossy:** This will blur glossy reflections and reduce the noise effect. You can put a 1.0 value.
- **Reflective and refractive caustics:** Caustics are light effects related to transparent and reflexive materials. We can observe this light effect with diamonds, for instance. This uses a lot of resources and generates some noise. By unchecking these two options, you can completely turn off these effects during rendering. In the case of our haunted house scene, we are going to disable them.

Performances

You will find the following performance settings in the menu:

- **Viewport BVH Type:** There are two types of this: **Dynamic** and **Static**. This is a way to let Cycles remember some of its rendering calculations for the next render. The **Static** option is highly recommended to optimize the rendering time, if you have no more polygonal modifications in your scene. Otherwise, you can use the **Dynamic** mode.
- **The Tiles:** This helps to manage the pixel groups that are to be rendered. So you can control their size along the *x* and *y* axes and also control the method to render the image (if you prefer to start rendering through the center of the image, or from left to right, and so on). The size of the pixel group to be rendered should be chosen according to your machine settings. If you are rendering with your CPU, you can choose a smaller tile size than if you were rendering with your GPU.

Note

For more information, you can have a look at the official Blender manual at these addresses:

<http://www.blender.org/manual/render/cycles/settings/integrator.html>

<http://wiki.blender.org/index.php/Dev:2.6/Source/Render>

Lighting

We are now going to look at a very important aspect of the rendering process: the lighting. Without lights, you won't see any objects, as in the real world. Good lighting can be hard to achieve, but it can give a nice atmosphere to the scene. One of the things that is true with a scene with good light is that you won't even notice the lights as they look like natural lighting. In order to get our lighting job done correctly, we are going to add a basic shader to every object in our scene.

Creating a testing material

Let's add a very basic material with Cycles in order to see the effects of the lights. In order to better understand the lights in the next section, we are going to create a blank scene and test our lights on a cube that is laid on a plane:

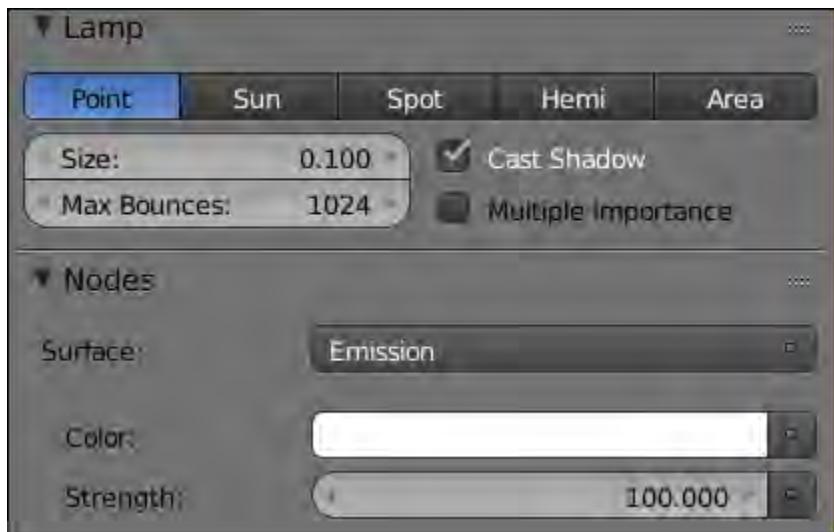
1. Let's start by creating a new scene and by adding a plane scaled ten times (**S > 10**) under the default cube.
2. We also want to delete the default light and turn the Cycles render engine on.
3. We aren't going to explore material creation in depth for now, so we advise you to follow these steps in order; later you will have more information about this process. We need to select the cube and open the **Material** tab of the **Properties** editor.
4. Now we will create a new material slot by clicking on the **New** button.
5. Under the **Surface** subpanel, we will click on the color and change its value to **1.0** in order to have a full white color. That's all for the material. It will be a handy material to test the different types of light.
6. The last thing we need to do is to add the same shader to the plane. To do this, we can copy the material of the cube. We will first select the object (or objects) to which we want to copy the material, in our case, the plane, and then we will select the object that owns the material that we want to copy. Then we will press **Ctrl + L** and select **Material**. The plane should now have the same white material.

Understanding the different types of light

The goal of this section is to understand how each type of light can affect our objects in a scene. We will give you a brief explanation and a short preview of what effects they can provide you with:

1. Before our tests, we will need to change the world shader that contributes to the lighting of the scene. If you press **Shift + Z** or change the **Viewport** Shading mode to **Rendered** in the 3D View header, you can see what the scene will look like, but you will see it in real-time in the viewport. Here you can clearly see the objects even if we don't have any lights. That's because the background color acts as if there was an ambient lighting.
2. If we go into the world settings of the Properties editor and change the Surface color to black, we will not see anything.
3. We can now add a light and have a better understanding of its effect without being disturbed by the world shader.

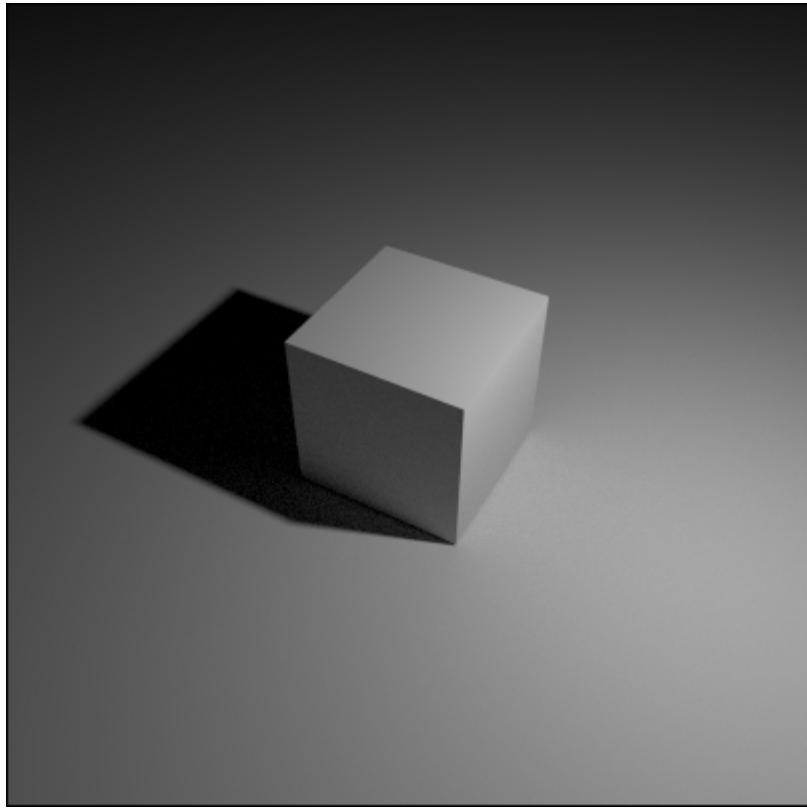
The settings of the lights can be found under the **Object Data** tab of the Properties editor (a yellow dot with a ray icon) while the light is selected. There are five types of lights (but four work in Cycles) that have many options in common: **Size** influences the hardness of the shadows they produce on objects and **Max Bounces** tells Blender the maximum number of bounces the light rays can travel. They also have the ability to cast shadows with the **Cast Shadow** checkbox turned on. Of course, they also have a strength that you can tweak in the **Nodes** subpanel. Note that, if you can't see the strength option, you need to click the **Use Node** button.



The shared light options

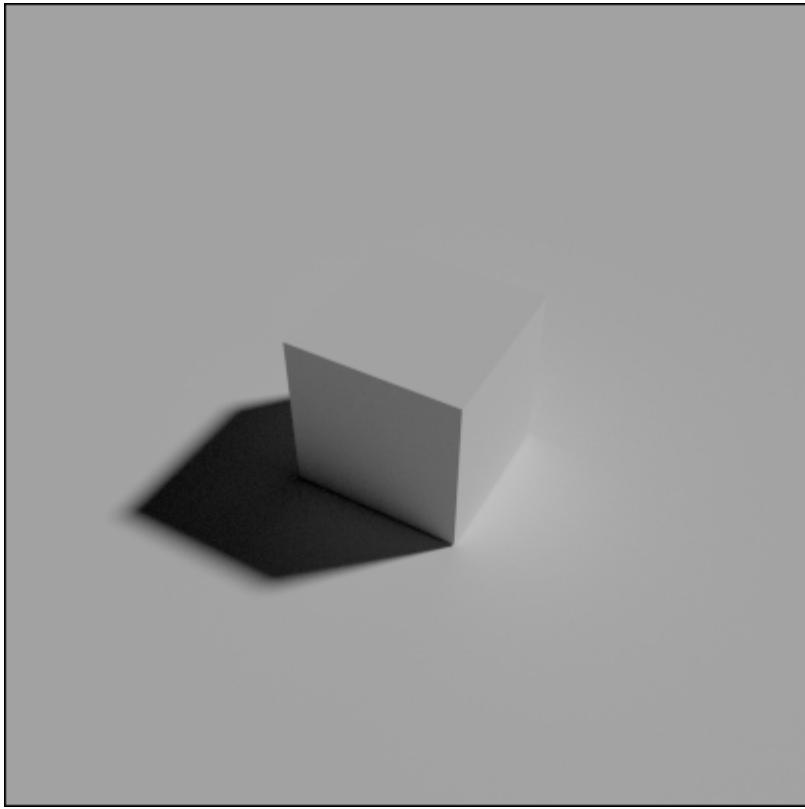
The different types of light are as follows:

- **Point:** As its name implies, it emits lights according to its position. It emits light rays in all directions, but these rays are limited to a certain distance from the center of the light. We often call it a spherical lamp.



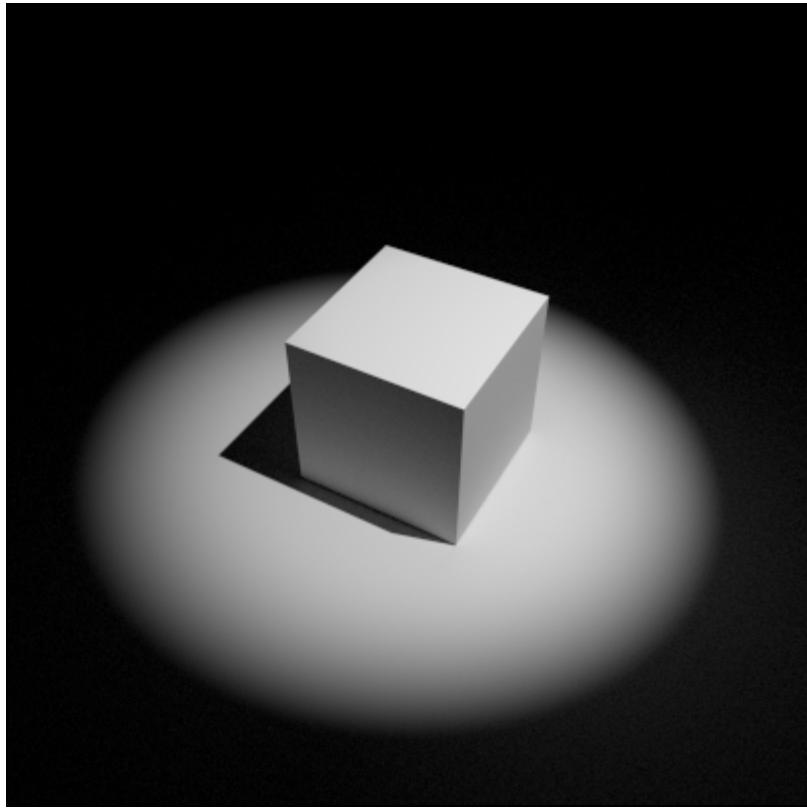
A point light with a strength of 500 and a size of 0.1

- **Sun:** As you can imagine, this type of light represents the way the sun works. As the sun is very far away from earth, we can admit the way we perceive its rays as parallel. So in Blender, the sun produces parallel rays, and we also don't care about its location, but only its orientation matters. With a small size, you can quickly represent the lighting of a bright day. You will mostly use it as a global light as it lights the entire scene. Also, notice that its strength should be less than the point light that we saw previously.



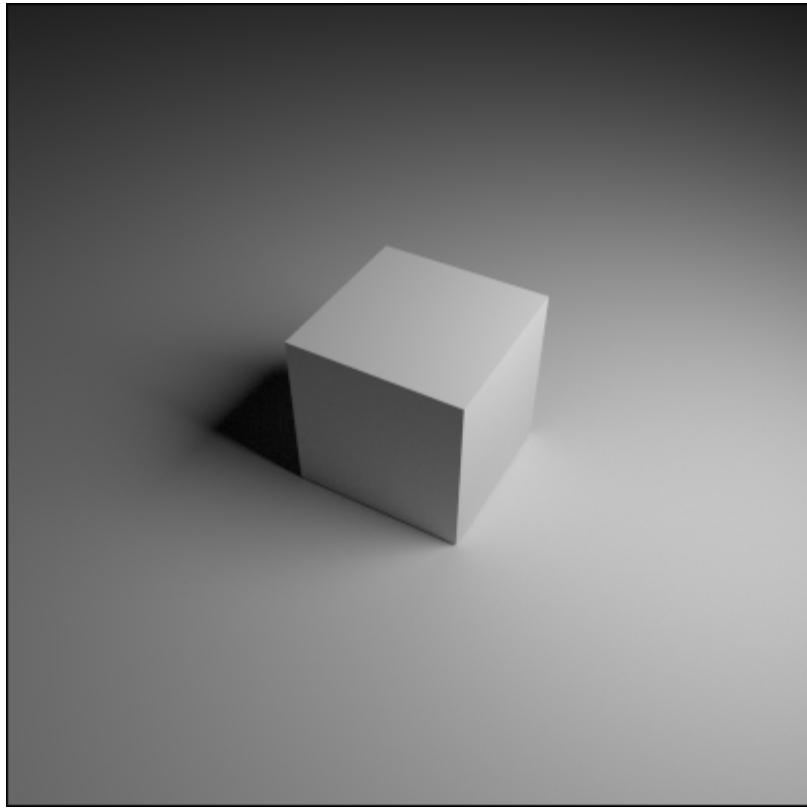
A 45-degree angle on Y and Z sun light with a strength of 2 and a size of 0.05

- **Spot:** The spot light is a conic light. It looks like the lamps used on stage in order to light the show presenter. The lighting that it will produce depends on its location, direction, and the spot shape. You can change its shape in the **Spot Shape** subpanel, and **size** will determine the size of the circle of light influence. The **blend** will define the hardness of the circle shadow. The circle size and the strength of the spot light will depend on its distance from the objects.



A 45-degree angle on a Y spot light with a strength of 5000, a size of 0.5, a shape size of 30 degree, and a blend of 0.8

- **Hemi:** For now, the Hemi type of light is not supported in Cycles. If you use it, it should react like a sun lamp.
- **Area:** This is one of the more common lights. It emits light rays from a plane according to a direction represented by a dotted line. It could be squared or rectangular. Its size, like for the other types, will affect the hardness of the shadows. The strength of the light will also depend on its distance from the objects. With this light, you can achieve a very precise lighting, so we strongly advise you to test this in order to be familiar with the way it reacts.

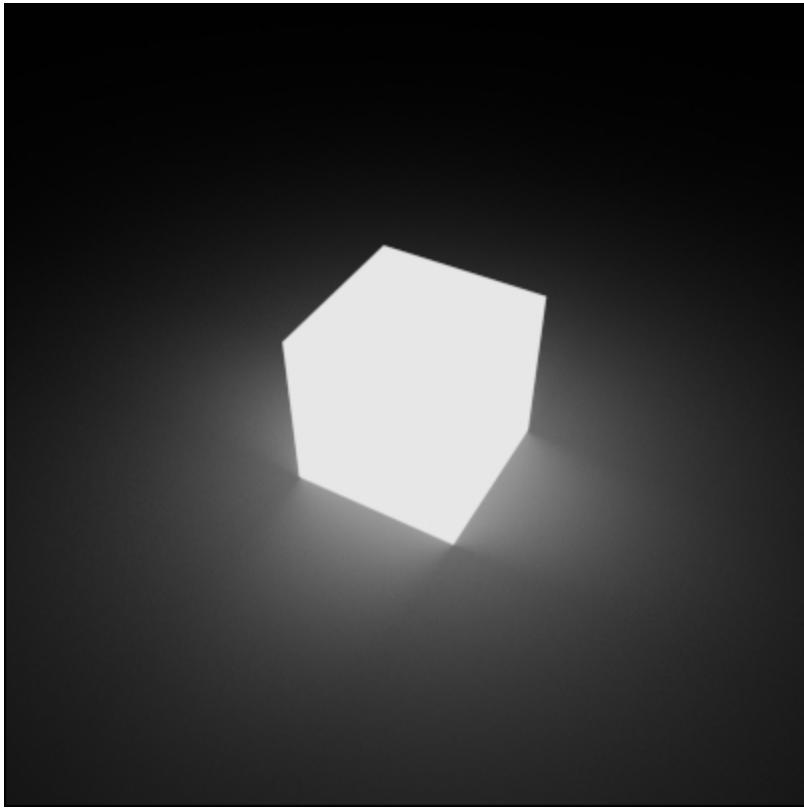


An area light with a strength of 500 and a square size of 5

Another option that many Blender users appreciate is using an emission shader to act as a light on an object (a plane, for instance). The emission shader is, in fact, the base shader of the other types of lights.

1. First let's add a plane.
2. Then, under the **Material** tab of the **Properties** editor, we will add a new material slot.
3. Change the diffuse surface shader from **Diffuse BSDF** to **Emission**.
4. As you may notice, if the plane is in the camera field, you can see it. We don't want this, so go into the **Object** tab of the **Properties** editor, and under the **Ray visibility** subpanel, uncheck **Camera**.

You may find this easier, but, in fact, with this method we lose a lot of control. The main problem we've found with this method is that we can't control the way the rays are emitted, for instance, with the area light. This can be useful when you want visible objects to emit light, but this is not very good for precise lighting.



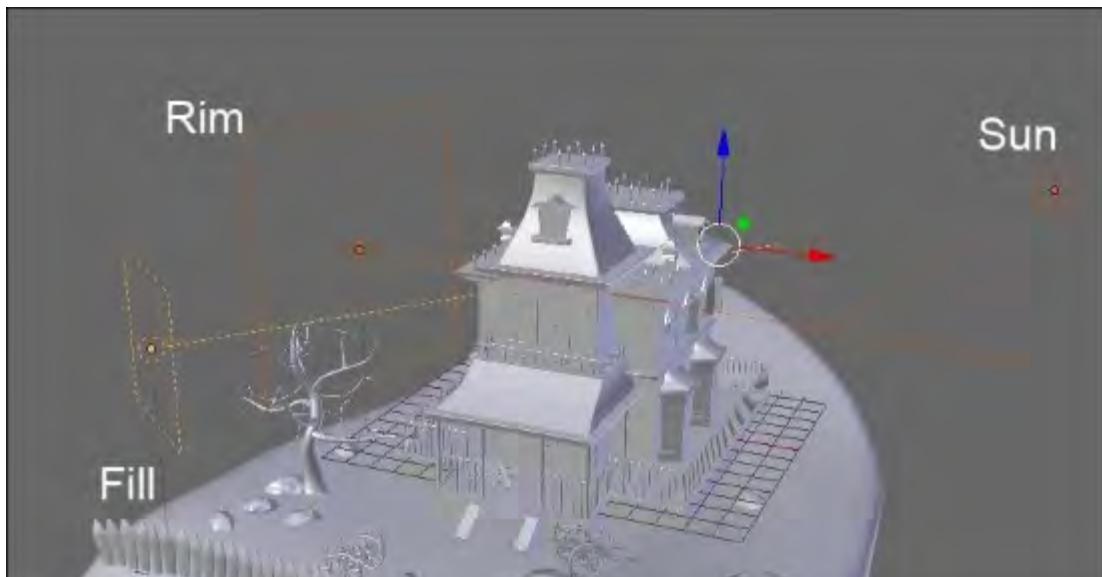
The cube with an emission shader

Lighting our scene

As you can see from the previous part, there are many types of light that we can manipulate in order to achieve a nice lighting effect. But in the case of our haunted house, we are only going to use area and sun lights because the other types of light are often used in specific situations.

1. We will start by opening our haunted house scene and saving it in another name (HauntedHouseCyles.blend, for instance).
2. Now we can delete all the lights that we used in the Blender Internal render.
3. While doing a lighting effect, it's a good to have an idea of the volume of your objects with a neutral material. So we will select one of the objects in the scene, remove its existing material, and create a new material in the **Material** tab of the Properties editor. As we did for our testing scene, we will change the color value to **1.0**.
4. Rename the material as **Clay**.
5. Now we will select all the objects in scene (**A**), and reselect the object that is the clay material while pressing **Shift** in order to make it the active object.
6. Press **Ctrl + L** and select **Material**. All objects will now share the same material.
7. We can now split our interface in two. One of the 3D views will display the camera point of the view (the **0** numpad key) and will be rendered in real time (**Shift + Z**) with a preview sampling of 50 (decrease it if you don't have a powerful computer).

8. The first light to be added is the sun. We will orient it, so it lights the right-hand side of the house. It will be nearly horizontal. Our goal here is to have a dawn lighting. The sun has a size of 5 mm in order to have harsh shadows and a strength of 1.0.
9. The next light that we will add will fill up the front of the house a little bit. It will be an area light that is a slightly tilted down and located on the front left side of the house. We want smooth shadows, so we will change its size to 5 m. Its intensity will be around 400. We can also change its color to be a little bit yellowish.
10. The last light will act as rim light. It will be an area light that comes from the back of the house on the left-hand side. Its size will be 10 m and its strength around 700. We have also tinted it a little bit towards blue.



A render of the lighting with our test material



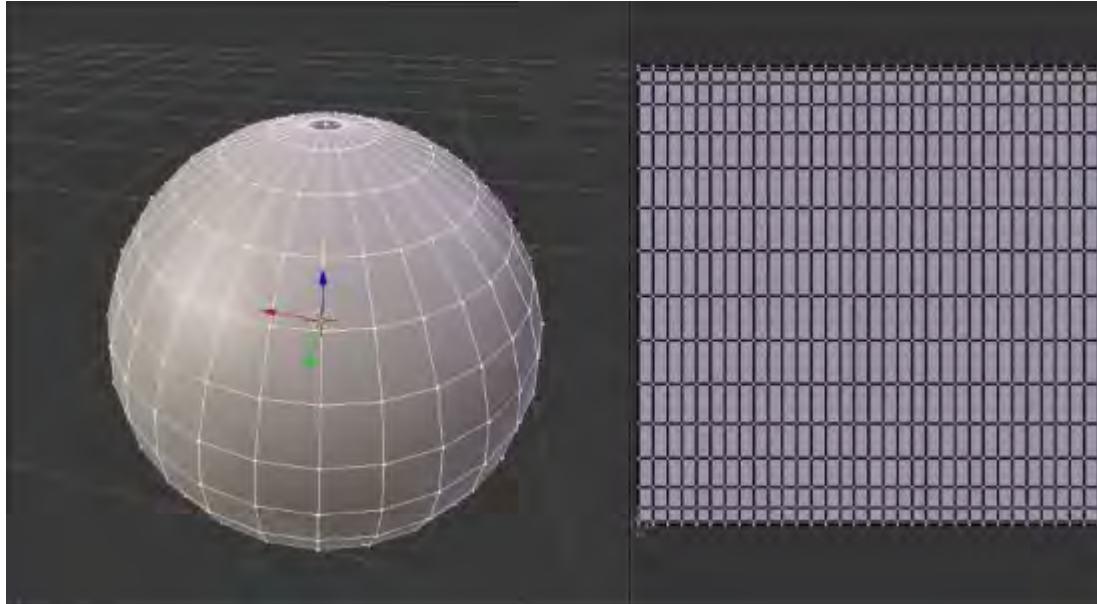
The lighting of the haunted house scene

11. That's all for the basic lights. The light settings are in constant evolution during the whole image creation pipeline, so don't be afraid to change them according to your needs later. Note that we are missing an environment lighting that we are going to set up in the next part of the chapter.

Painting and using an Image Base Lighting

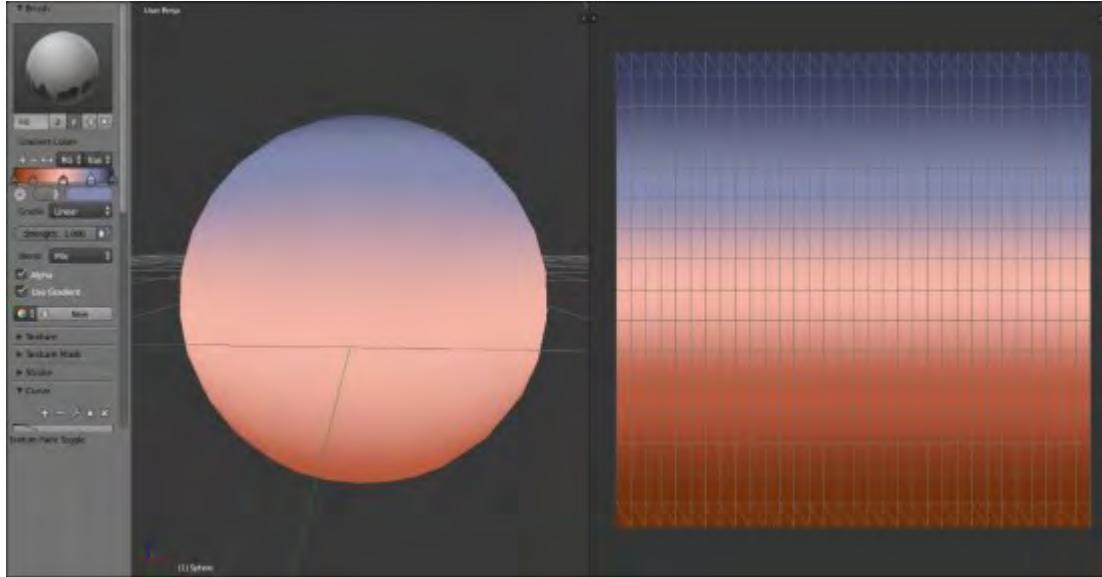
An **Image Base Lighting (IBL)** is a very convenient technique that allows us to use the hue and the light intensity of an image to lighten up a 3D scene. This can be a picture of a real place taken with a camera. HDR images provide very realistic results and may be enough to light a 3D scene, but for our haunted house scene, we will paint it directly in Blender with Texture Paint. This technique allows us to do complex lighting in less time and will enrich the lighting that we have prepared previously. We will start by seeing how to prepare the painting phase of a customized IBL:

1. We will open a new scene in Blender.
2. We will split the working environment in half with a **UV/Image Editor** on the right-hand side and a 3D View on the left-hand side.
3. We will add a UV Sphere at the center of the world (*Shift + A* and select **Mesh | UV Sphere**) on which we will paint the sky.
4. We will delete the vertices at the two poles of the sphere. We will make a scale extrusion (*E* and *S*) of the edges and slightly reposition them again for a well-rounded look. We will obtain a sphere pierced on both ends. It is important to have these holes for the UV projection.

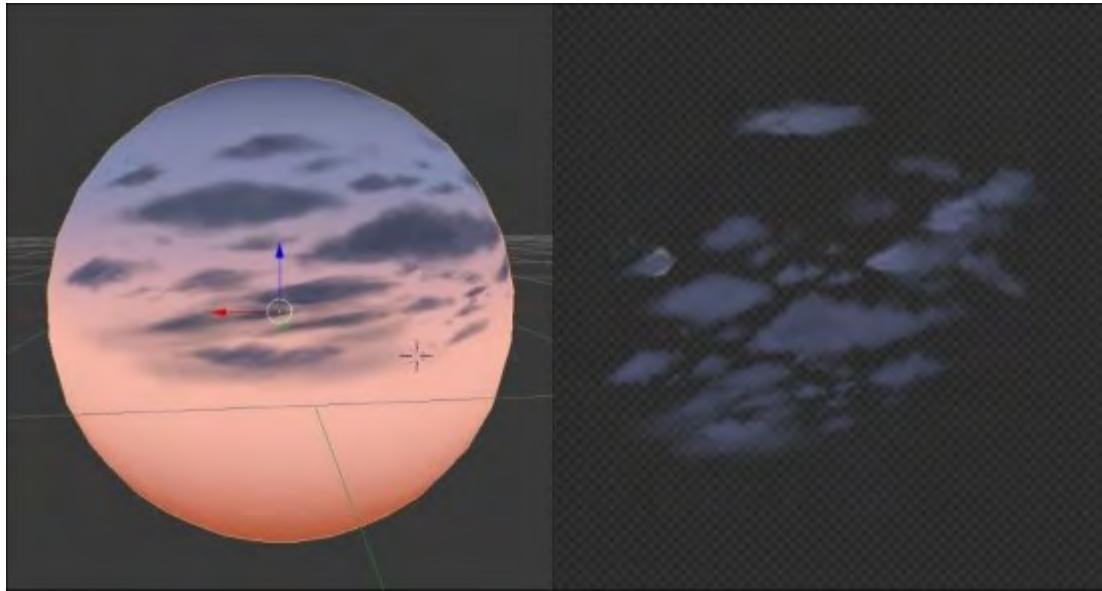


5. We will select all the polygons of the sphere (*A*), then we will apply **Unwrap Cylinder Projection (U)**. In the **Cylinder Projection** options on the left panel (*T*) of the 3D Viewport, we will change the **Direction** parameter by selecting the **Align to Object** option. This allows us to get straight UVs that occupy the most space on the entire UV Square.
6. Once the UVs are created, we will select the edge loops that form the holes at the poles of the sphere, and we will merge them each in turn to form a complete sphere. This will form triangles in the UV, but it does not matter.

7. We will again select all the polygons of our sphere, and we will then add a new texture by clicking on the **+ New** button in the **UV Image** Editor.
8. In Blender Internal Renderer mode, we will create a new material on which we will place our IBL texture. To better visualize the texture, we will check the **Shadeless** option (**Material | Shading | Shadeless**).
9. In **Texture Paint**, we can start to paint.

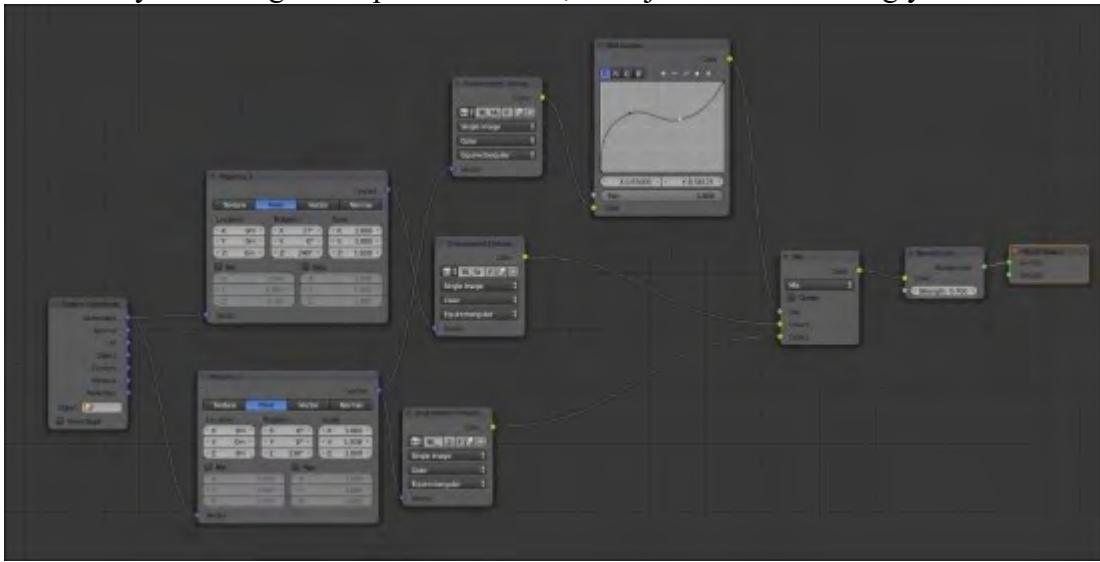


10. We will use Fill Brush with the **Use Gradient** option in order to prepare the gradients of the sky. We will use the following Gradient Colors from left to right. The color marker number **1** on the far left is **R: 0.173, G: 0.030, and B: 0.003**. The color marker number **2** is located at **0.18**, and its color is **R to 0.481, G to 0.101, and B to 0.048**. The color marker number **3** is at position **0.5**, and its color is **R to 0.903, G to 0.456, and B to 0.375**. The color marker number **4** is located at **0.78**, and its color is **R to 0.232, G to 0.254, and B to 0.411**, and the last marker at the far right is the color **R: 0.027, G: 0.032, and B: 0.085**. We will then apply the gradient upwards on the sphere.
11. On the texture in the **UV/Image** Editor, we can see some black near the poles, which may interfere with the lighting calculation. Thus, in the **Paint Mode**, we will take the nearest color of the black triangles by pressing the **S** shortcut (without clicking), and we will fill the black triangles with **Fill Brush** (without Gradient).
12. When this is finished, we can save this image as **IBL_Sky**.
13. In the **Texture Paint** mode, in the **Slots** tab, we will add a **Diffuse Color** texture that will be transparent this time. For this, we will check the **Alpha** box, and we will change the alpha value of the **default fill color** to 1 in the **Texture Creation** menu. This will allow us to create clouds on another texture while keeping our sky visible.
14. With the **TexDraw** brush and the **R: 0.644, G: 0.271, B: 0.420** color, we will draw a few clouds. We must think that there will be only the upper half that will be displayed on the framing of the haunted house. This part will have the greatest importance for the lighting. So we must focus on the upper half of the texture.



15. We will save the image as **IBL_Cloud**.
16. From this cloud texture, we will make a mask that allows us to properly mix the sky and the clouds. For this, we must save our image, with the **BW** (Black and White) option and not in **RGBA**, by naming it as **IBL_Mask**.
17. We will then return to the haunted house scene, and in the **Node Editor**, we will click on the **World** icon that is represented by an earth, and we will check the **Use Node** option.
18. We have two nodes that appear: **Background** and **World Output**. We will add an Environment Texture node (press *Shift + A* and select **Texture | Environment Texture**).
19. We will duplicate the **Environment Texture** node twice (*Shift + D*). We will place them one above the other and to the left.
20. In each **Environment Texture** node, we will open the IBL textures created previously.
21. We will add a **Mix RGB** node (press *Shift + A* and select **Color | Mix RGB**) that will allow us to mix our textures. We will connect the **IBL_Sky** Color Texture Image Output socket to the **Color1** input socket of **Mix Shader**, the **IBL_Cloud** Texture Image Color Output socket to the **Color2** input socket of **Mix Shader**, and the **IBL_Mask** Color Output Socket to the **Fac** input socket. We will keep **Mix** as the Blending mode.
22. We will add a **Mapping** node (press *Shift + A* and select **Vector | Mapping**) that we will duplicate once, and we will position them one above the other on the left-hand side of the **Environment Textures** node. We will rename them as **Mapping_1** and **Mapping_2**. We will connect **Mapping_1** to the **IBL_Sky** Texture Image node and **Mapping_2** to the two other **Environment Textures**.
23. We will add a **Texture Coordinate** node (press *Shift + A* and navigate to **Input | Texture Coordinate**) that we will position at the left. We will connect the **Generated** socket of the **Texture Coordinate** node to the **Vector Input** socket of the two **Mapping** nodes
24. To get a better contrast for the **IBL_Mask**, we will place a **RGB Curves** node (*Shift + A* and select **Color | RGB Curves**) between the **Environment Texture** node and **Mix RGB**. We will set the following two points: the first point at the **X= 0.24** and **Y= 0.65** position, and the second point at the **X= 0.65** and **Y= 0.58** position.

25. We have all the necessary nodes. To finish, we will need to modify the mapping of the **IBL_Sky** texture. Therefore, we will modify the **X= 6°**, **Y= 23.9°**, and **Z= 0°** rotation parameters. These values vary according to the painted texture, so adjust them accordingly.



To visualize your **Image Base Lighting (IBL)** better, you can display it in the viewport. In the **Solid** mode, in the right panel of the 3D Viewport, check the **World Background** option (**Display | World Background**).

Creating materials with nodes

It's now time to discover the material creation process with Cycles. In this section, we are going to create the basic shaders that are composed of our previously painted textures. The shaders won't be at their final stage here. Later, we are going to improve them with normal maps.

Creating the materials of the house, the rocks, and the tree

Let's start with the wall shader of the house:

1. We will first select the corresponding object.
2. We are going to duplicate the clay shader that we had added in order to test our lighting in the previous section. As you can see, it is used by 68 objects in the scene. If you click on the 68 button on the right-hand side of the material name in the material tab of the Properties editor, you will duplicate the shader and make it unique. At this time, we can now rename it as **HouseWall**.
3. We are now going to switch to the **Node Editor** in order to have more control on our shader. In fact, we can do everything in the Properties editor, but it will be quite hard to manage with a complex shader. So open a new editor and change it to a **Node Editor**.
4. As you can see, we already have **Diffuse BSDF** plugged into the **Surface** input of the **Material Output** node.
5. A diffuse shader has no shine on it. It looks flat. In real life, every surface is at least a little specular, so we are going to mix our diffuse shader with another shader that will bring us the shiny effect. To do this, we will first add a **Glossy BSDF** shader (press *Shift + A* and select **Shader | Glossy BSDF**) and place it under the diffuse shader. Don't connect it for the moment.
6. In order to mix the two shaders together, we will use a **Mix Shader** node (*Shift + A* and select **Shader | Mix Shader**). As you can see, this node has two shader inputs. Plug the BSDF output (green dot) of the diffuse shader to the first shader input of the **Mix** shader, and the BSDF output of the Glossy shader to the second shader input of the **Mix** shader. Now plug the **Mix** shader output to the surface input of the **Material Output** node. As you can see, both shaders have been mixed together. You can now use the **Factor** slider in order to choose which one is predominating. If you put a value of **0**, you will only use the shader connected to the first input (the diffuse), and if you put a value of **1**, you will only use the shader connected to the second input (the glossy one).
7. The blend between these shaders is not going to look right with any value, so we are going to connect a **Fresnel** node to the **fac** input (press *Shift + A* and select **Input | Fresnel**).

Note

About the **Fac** input

The role of the **fac** input is to control how both shaders will be mixed. Usually, the **Fac** input needs to be fed with black and white information where the amount of black tells us how much the first shader will be used, and the amount of white tells us how much the second shader will be used for the final output.

8. We can now change the value of the Fresnel to **1.4**.

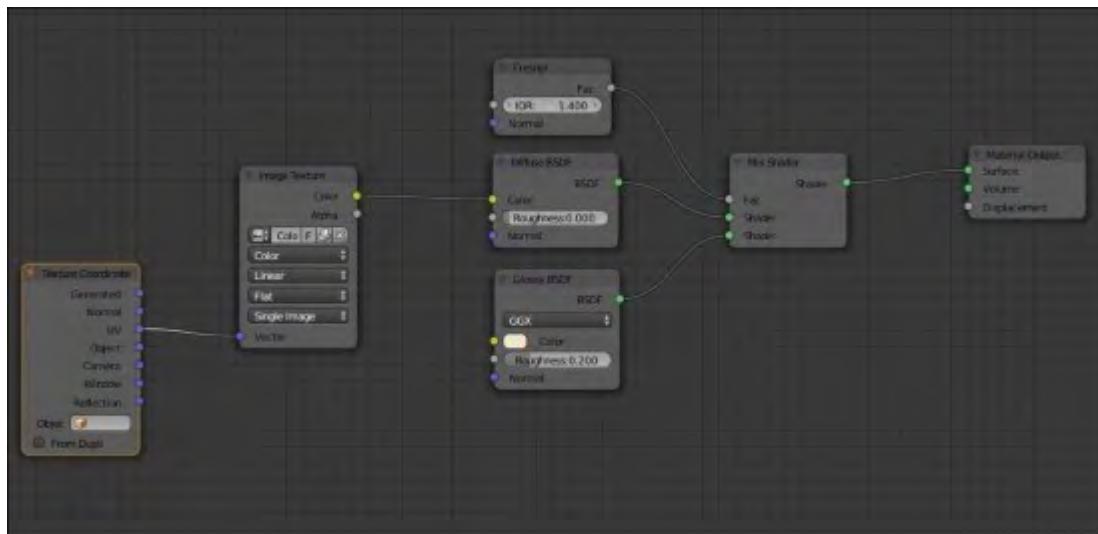
Note

About the Fresnel

The Fresnel node will produce a black and white texture according to the volume of the geometry. It will be calculated according to the light ray's incidence. We usually use a Fresnel node in order to catch the highlights better. You can use a pretty interesting add-on called node wrangler that allows you to quickly see the result of each node without shadows. In order to use it, right-click on the node you want to see while pressing *Ctrl* and *Shift*.



9. We will now change the glossy color to a yellowish tint. Don't forget to turn on real-time shading in order to have a preview of what this will look like in the render. You can also drag a rectangle to the zone you want to preview with the *Shift + B* shortcut while being in camera view. If you want to remove the rectangle zone, drag a new zone to the outer zone of the camera in the camera view.



The base of our wall shader

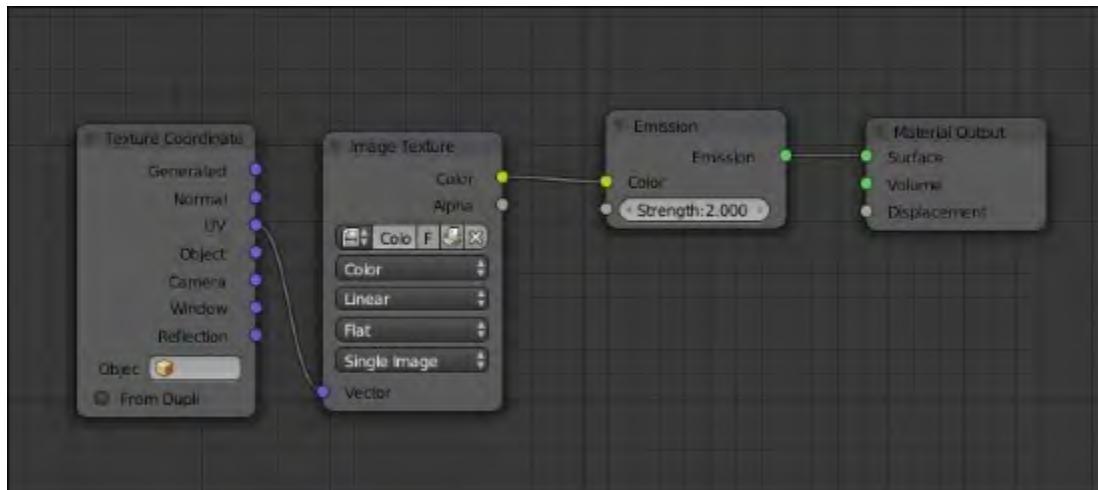
The last thing we will do with this material for now is plug it into our texture:

1. We will add a new Image Texture node (press *Shift + A* and select **Texture | Image Texture**).
2. We will need to connect the **Color** output of this node to the **Diffuse BSDF** color input.
3. It's also a good idea to add a **Texture Coordinate** (press *Shift + A* and select **Input | Texture Coordinate**) node and plug the UV slot to the **Vector** input of the **Image Texture** node.

By default, the Vector inputs are set to be UV, but with this node, we can clearly see the mapping method used for the textures.

1. We can now select one of the roofs and change its clay shader to the one we created because the roof shader will be nearly the same. Now, in order to break the link between the roof and the wall shader, we can press the button with the number of objects that share the same material in order to copy the material.
2. We will rename this shader to **HouseRoof1**.
3. The only thing we need to do for now is to change the texture of the Image Texture node to the corresponding roof texture.
4. We can now select all the objects that need to share the same material (the other blue roofs), and finally, we select the roof that has the shader that we want to share, press *Ctrl + L*, and select **Material**.
5. We will now repeat the process of creating a new material by copying it from the previous one, changing its name and texture information, and linking it to its corresponding objects.

We will now have a shader on the rocks, the tree, and all the different objects that make up the house. The only shader that will be different is the one on the top window. It needs to emit light. In order to do this, we will copy the previous material, delete the diffuse, glossy, and mix shader (*X* or *Delete*), replace them with an **Emission** shader (press *Shift + A* and select **Shader | Emission**), and plug the window texture to this. Now the top light is going to emit light! You can tweak the emission value if you want more light. As you can see, we need to do this for the other windows as well, but in the case of the other windows, we can't do this simply because they are not planes and their color information is located on the wall texture. That's why we need to paint a mask.

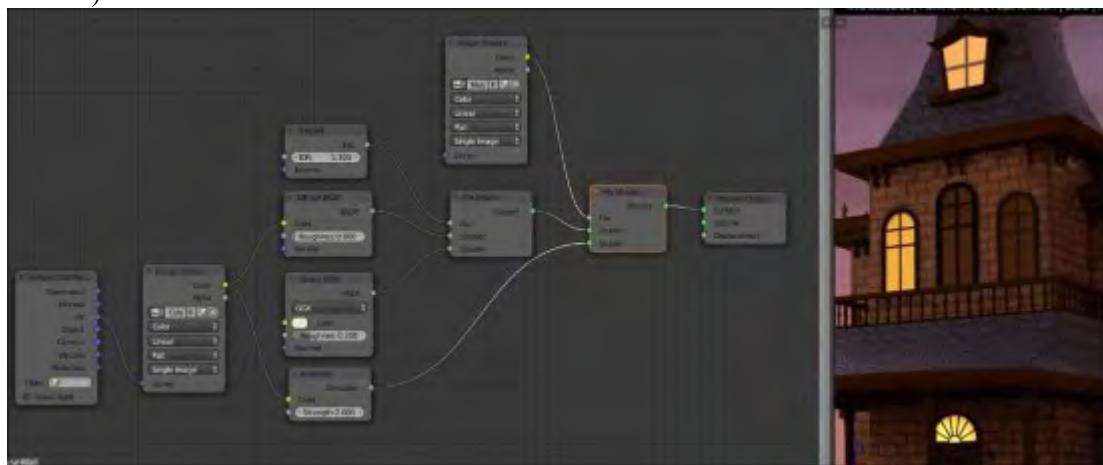


The top window shader

Adding a mask for the windows

We are now going to improve our wall material by creating a mask that will separate the windows that are shining from the rest. These windows are going to be painted white and the rest will be black. So when we plug the mask in the **Fac** input of a Mix material node, we will be able to choose an emission shader for the white parts.

1. We are going to paint our map in the **Blender Internal** context. Note that we can actually use the **Texture Paint** mode while being in Cycles, but this implies that we add and select a texture node that uses the texture that we want to paint.
2. So we will select the wall object, and in the **UV Image** editor, we will create a new 1024 x 1014 black texture. Usually, masks don't need large resolutions.
3. Now in pure white paint the windows that shine (the light yellow ones on the color map).
4. Let's go back to our wall material and add a mix shader just before the output node. If you want to save time, you can drag the node on the connection line of the previous **Mix** shader and the **Output** node. This will automatically do the connections for you.
5. The first shader input is already used by our old **Mix** shader. Now we are going to add a new **Texture** node with our mask and plug its output to the **Fac** input of our new **Mix** shader.
6. The second shader input will be fed with an **Emission** shader (press **Shift + A** and select **Shader | Emission**). In the **Color** input of this node, we will plug our color map (the same as in **Diffuse** shader).



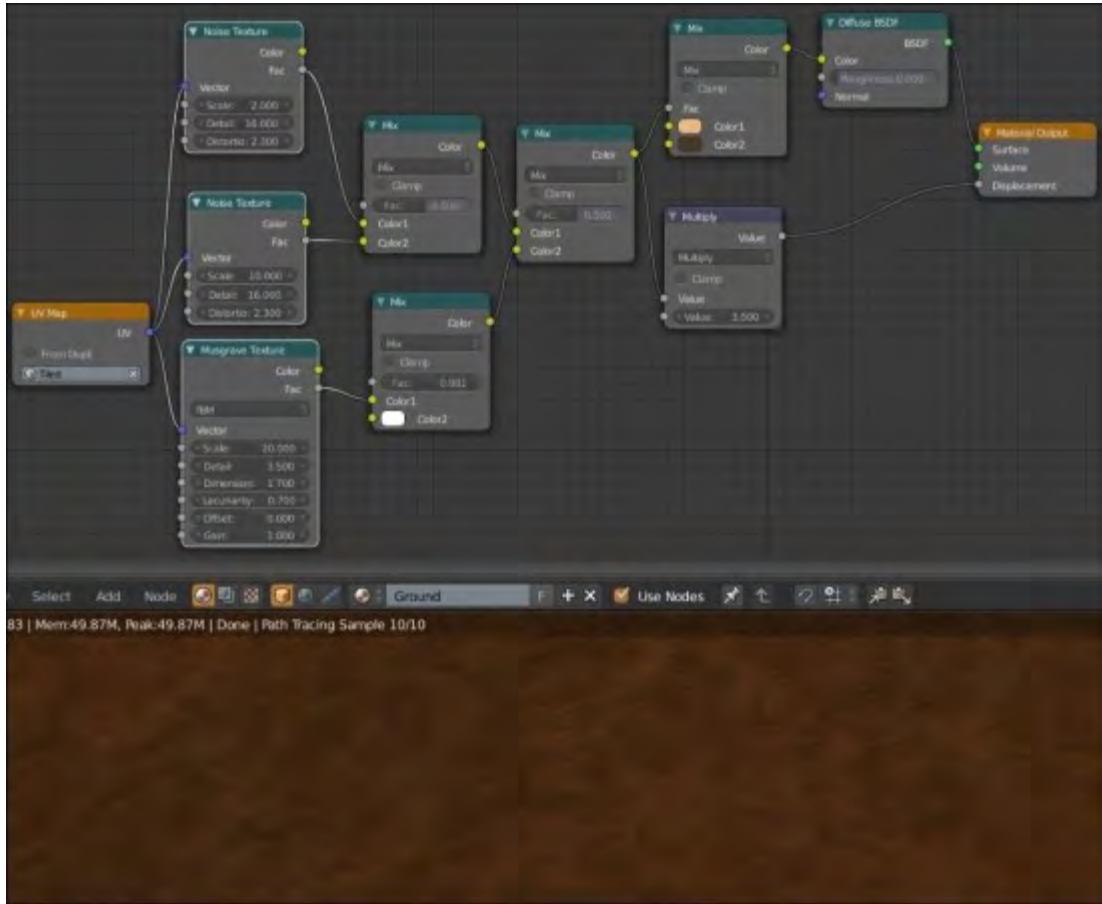
The wall material with the mask on the left-hand side and the result in the real-time rendered 3D view.

7. Now we can increase the strength of the Emission shader to 2.0. As you can see, now our shining windows (and only them) emit light!

Using procedural textures

One thing that could be very interesting when creating materials is generating their textures procedurally. In this render, we are going to replace the hand-painted ground by a procedural material. This is done as follows:

1. We will select the cliff and create a new material for it.
2. The next thing is to add a **Diffuse BSDF** node. The color input of this diffuse material will be fed with a mix of procedural textures.
3. Let's add a Noise texture node (*Shift + A* and select **Textures | Noise**) and duplicate it (*Shift + D*). The first one will have a scale of 2.0, and the second one will have a scale of 10.0. Our goal is to have a mix of both levels of noise. Both of them will use the Tiled UV layer, so add a **UV Map** node (*Shift + A* and select **Input | UV Map**), select the correct map, and feed the **Vector** input of the noise textures with the **UV Map** node output.
4. As these nodes are textures and not shaders, we aren't going to use the **Mix Shader** node but the **MixRGB** node instead. So we will add one of these nodes (*Shift + A* and select **Color | MixRGB**), and feed the inputs with their noise **Fac** output. Don't use the color output as we want a black and white mix here. Remember that you can always test your results with the Node Wrangler add-on (*Shift + Ctrl* and right-click on any node).
5. We are now going to mix this result with a **Musgrave** node (*Shift + A* and select **Texture | Musgrave**). We also need the Tiled UV for the vector input. We will set the **Scale** to **20.0**, the **Detail** to **3.5**, and the **Dimension** to **1.7**. Its effect will be pronounced in the final result, so we are going to mix it with white. To do this, we will add the **MixRGB** node and plug the first **color** input with the **Fac** output of the **Musgrave** Texture. The second color slot can be changed to white. We are going to change the **Fac** slider of the **MixRBG** node to 0.98; this will make Musgrave very subtle.
6. We can now mix our noises and the **Musgrave** results together with one more **MixRGB** node.
7. If we plug this directly to the color input of the diffuse, we will get a black and white result. In order to introduce color, we will need another **MixRGB** node, but instead of feeding the color inputs we are going to plug our texture in the **Fac** input and choose two brownish colors. Now we can plug the result to the color input of the Diffuse shader.
8. Lastly, we can plug the black and white texture to the displacement input of the Material output node. In order to raise the displacement effect, we can place a **Math** node in-between (*Shift + A* and select **Converter | Math**). We can change its operation to **Multiply** and use a **3.5** value.



The ground material with the procedural texture made with a noise and Musgrave combination on the left-hand side and, the result in the real time rendered 3D view.

Making and applying normal maps in Cycles

As we saw it previously with the alien character, normal map allows us to simulate a relief on a 3D mesh very efficiently. It would be good to generate a few of them in order to add some relief to our scene. We will explore a method to easily generate normal maps from tiled, hand-painted textures:

1. We will open a new Blender scene.
2. We will delete the cube (**X**) and then we will add a plane in the middle of the scene (press **Shift + A** and select **Mesh | Plane**) that we name as **Plane-1**.
3. We will split the screen in two parts to open the **UV/Image Editor** at the right-hand side.
4. We will add a **Multires** modifier to our plane and a **Displace** modifier. We will place the **Multires** modifier above the **Displace** modifier.
5. In the **Texture** tab, we will create a new texture by pressing **New**, and then we will load the tiled texture of wood, that is, **WoodTilePlank.png**.
6. We will check that the texture is loaded in the **Displace** modifier.

7. In the **Multires** modifier, we will check the **Simple** mode, and we will click on **subdivide** until we get to **level 8**. The more the mesh is subdivided, the more the Displace effect is accurate, but we must pay attention to the RAM of the computer.
8. We will modify **Strength** to **0.25**. This can vary depending on the texture. We must avoid important deformations on the mesh.
9. We will duplicate the plane (*Shift + D*), and then we will delete the modifiers of this new plane that we name as **Plane-2**.
10. We will select all the faces (**A**) of **Plane-2** in the **Edit Mode**, and we will do an unwrap (**U | Unwrap**) with a square shape taking all the UV surfaces. Then, we will add a new image. In the **UVMap Editor**, we will click on **New Image** (**Image | New Image**). We will name it as **WoodTilePlank_NM.png**.
11. We will move **Plane-2** to the same height as **Plane-1**.
12. Enter the top view (the 7 numpad key) for a better view of the mesh.

If the Displace doesn't give exactly the effect we want, we can make a few modifications with the **Sculpt** mode after applying the displace modifier by pushing on **Apply**. This is what we will do with the rock and the tile roof textures.

We can now bake a normal map as follows:

1. We will need to click on the **Smooth Shading** button, otherwise we will see the polygons on our bake.
2. Then, we will have to first select **Plane-1 (RMB)** and then **Plane-2** (press *Shift* and the RMB). This becomes the active object.
3. Now, in the **Properties** editor, under the **Render** section, we will expand the **Bake** subpanel.
4. The first option to choose is what type of map (or texture) we want to bake. So, in the **Bake Mode** drop-down menu, we will select **Normal Map**.
5. The next thing we'll have to check is the **Selected to Active** option that tells Blender to bake from the sculpture to the active object (our low poly plane).
6. Now you can click the **Bake** button. Don't forget to save your map (select **Image | Save As Image** or press *F3*), or it will be lost!

We will use the same process of Normal Map creation for every tiled texture. Once this is done, we can return to our shaders and apply our normal maps.

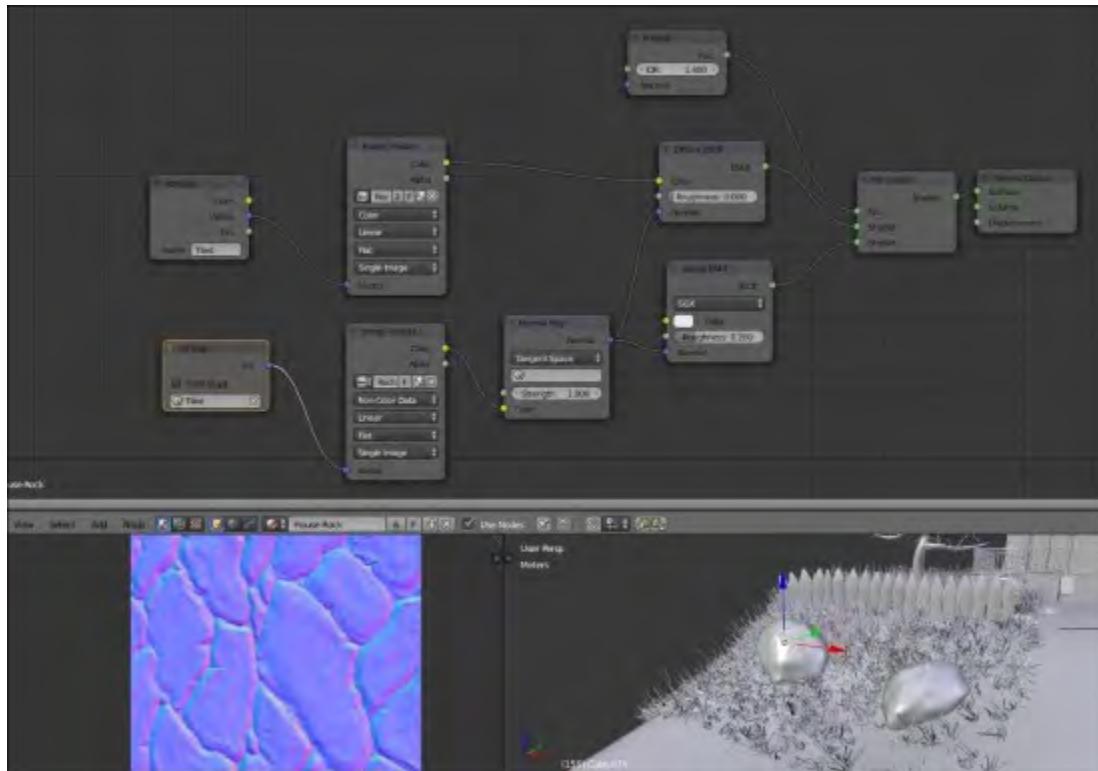
We will start with the House-Rock shader, which is very simple for the moment:

1. We will add a **Glossy** node (*Shift + A*) and navigate to **Shader | Glossy BSDF**, which we will mix with **diffuse** using a **Mix Shader** (*Shift + A* and select **Shader | Mix Shader**). To better visualize the normal maps, we will need glossiness.
2. We will add **Fresnel** (*Shift + A* and navigate to **Input | Fresnel**) that we will position in the **FAC** input socket of the **Mix Shader** node. We will put a **IOR** value of **1.4**.
3. We will add a **Normal Map** node (*Shift + A* and select **Vector | Normal Map**) with a **Strength** value of **1.0** and connect its output to the **Normal** input socket of the Diffuse and Glossy shaders.
4. We will duplicate an Image Texture node (*Shift + D*), and we will open the normal map texture named **Roch-Tilling-NM.png**. We must switch the **Color Data** option of this second Image

Texture to **Non-Color Data**. The data should not be interpreted as color data but as normal direction data.

- We will add a UV Map node (*Shift + A*) and navigate to **Input | Glossy UV Map**. Then we will change the UVLayer to **Tiled**. You will recall that the rocks have two UV layers, so we must select one.

We can apply this process for almost every shader using hand-painted tiled textures, except the brick walls in our case. It is a special case that requires that we bake the normal map on a larger map with a little modification of painting in order to hide the bricks behind the windows.



The normal map of the rock (low left corner) and its material in the nodal editor (top)

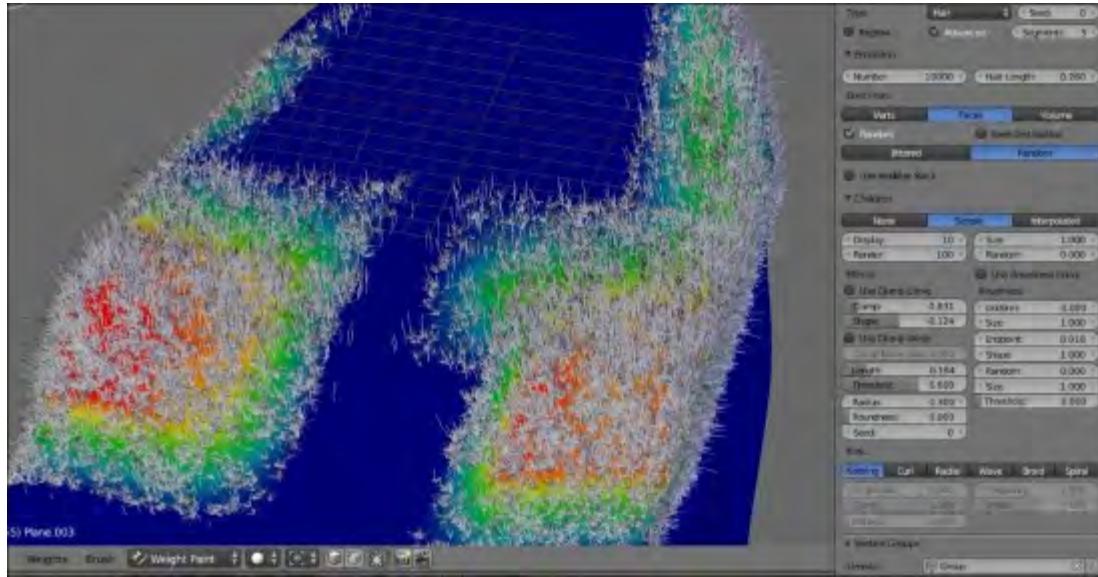
Creating realistic grass

In this section, you will see how we can create realistic grass in place of the actual grass planes. In order to create our realistic grass, we are going to use a hair particle system.

Generating the grass with particles

A particle system will be used here in order to generate the grass strand without modeling and placing them by hand:

1. We will first select the cliff and isolate it (*/ Numpad*). Then, we can go into the **Particle** tab of the **Properties** editor and add a new particle system. This will add a new modifier in the stack, but we can only control it here.
2. We will now have to change the type from **Emitter** to **Hair**. We can activate the **Advance** tab.
3. In the **Emission** subpanel, we can change the **Number** value to **10000**, which corresponds to the number of grass strands that we will have. We will also emit the particle from the faces in a **Random** manner. We can also change the **Hair Length** to **0.26**.
4. In the **Physics** subpanel, we can change the **Brownian** value to 0.120 in order to add more randomness to the grass.
5. In the **Children** subpanel, we can set the amount of strands we want to spawn around the main guides. In our case, we will activate the Simple mode and set **10** children for the **preview** (in the viewport) and **100** for the **render**. In the effect section, we can change the **Clump** value to **-0.831** so that the children start near the base of the guide strand and are sprayed out near the tip. We can also change the **Shape** value to **-0.124** to shrink the children in the middle a little. We will also change the **Endpoint** value to 0.018 to add more randomness.
6. We will now paint a vertex group with the **Weight paint** tool in order to choose where we want grass on the cliff. For instance, we don't want strands under the house. To do this, we will switch from **Object mode** to **Weight Paint** mode while the cliff is selected. As you can see, you have brushes as in the **Sculpt** mode. We can use the **Add** brush to add weight and the **Subtract** brush to remove weight. **Red** means that it will be full of grass, **Blue** means you won't have grass. Now, we can choose the vertex group that we have painted in the **Density** field of the **Vertex Groups** subpanel in the Particle System settings.
7. Now we can add a new particle system in order to add long grass. To do this, we will add a new slot in the particle settings. We will also copy the settings from the previous particle system but unlink them (the button with the number on the right-hand side of the name). We can lower the number of particles to **1000** and change the **Hair Length** to **1.120**. The number of children will be **5** in the **Simple** mode.

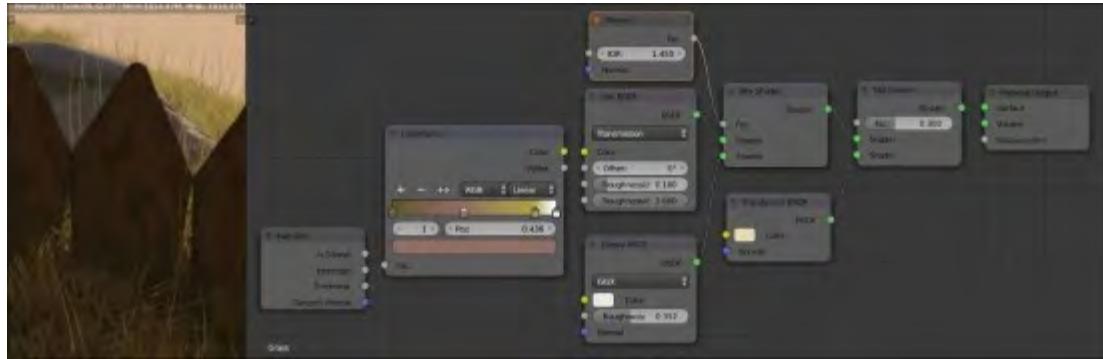


The settings of the grass

Creating the grass shader

Now the last thing that we will create is the grass material:

1. The first thing we will need to do is change the **Cycles Hair Settings** in the **Particle Systems** tab. We will change **Root** of the strand to **0.20** and set **Tip** to **0.0**.
2. We can now add a new material slot for the cliff and rename it as Grass. In the particle settings, we will change the material to **Grass** under the **Render** subpanel.
3. We will now select the grass shader and open the **Node editor**. The first node that we will add is the **Hair BSDF** shader (press *Shift + A* and select **Shader | Hair BSDF**) and change it from **Reflection** to **Transmission**. We will mix it using **Mix Shader** with a **Glossy BSDF** shader. We will change **Glossy Roughness** to **0.352** so that the glossiness is more diffuse.
4. Next, we will have to plug a **Fresnel** node to the **Fac** input of **Mix Shader**.
5. For the color of **Hair BSDF**, we will add a **Color Ramp** node (press *Shift + A* and select **Converter | Color Ramp**). For its **Fac** input, we will add a **Hair Info** node (press *Shift + A* and select **Input | Hair Info**) and choose the **Intercept** output. This will enable us to set the different colors along the strand. We will do a gradient that starts from a brownish color (to the left) to a desaturated green (to the right). Usually, the tip of the grass strand is white, so we will add a small amount of white on the far right of the color ramp.



The grass shader (to the right) and the result (to the left)

6. We will also mix the result of our Mix shader with a **Translucent BSDF** shader (press *Shift + A* and select **Shader Translucent**). Indeed, the grass is very translucent. We will change its color to a desaturated yellow and change the **Fac** value of the shader to **0.3**. We can finally plug our latest **Mix** shader to the surface input of the **Material** output node.

Baking textures in Cycles

Cycles allow us to bake textures as does Blender Internal, but there are some differences between the two render engines.

Cycles versus Blender Internal

As we have seen previously, texture baking in Blender Internal can be very efficient to produce normal maps, ambient occlusion, color textures, and many other kinds of maps that we won't cover here. All of this in a very short time. So you might wonder why it is interesting to bake in Cycles.

Cycles is a ray tracer render engine based on physical parameters with global illumination. It is then possible to get some very realistic renders in a much more efficient manner. Baking in Cycles allows us, for example, to calculate a few heavy special effects only once, such as caustics. When a render is baked on a texture, you can visualize the effect in real time. This can be very useful if you want to change the frame and make several renders. In this way, it is possible to create a realistic environment in real time.

However, in the context of a video game, if you have many dynamic assets you must pay attention. You could be limited by fixed lighting. Even though baking in Cycles can be very interesting, it has some faults. Doing a good baking without noise requires the same settings as a normal render, so you do need a high sampling value, which greatly increases rendering time compared to Blender Internal.

Note

For more information, you can have a look at the official Blender Reference manual at this address:

<http://www.blender.org/manual/render/cycles/baking.html>

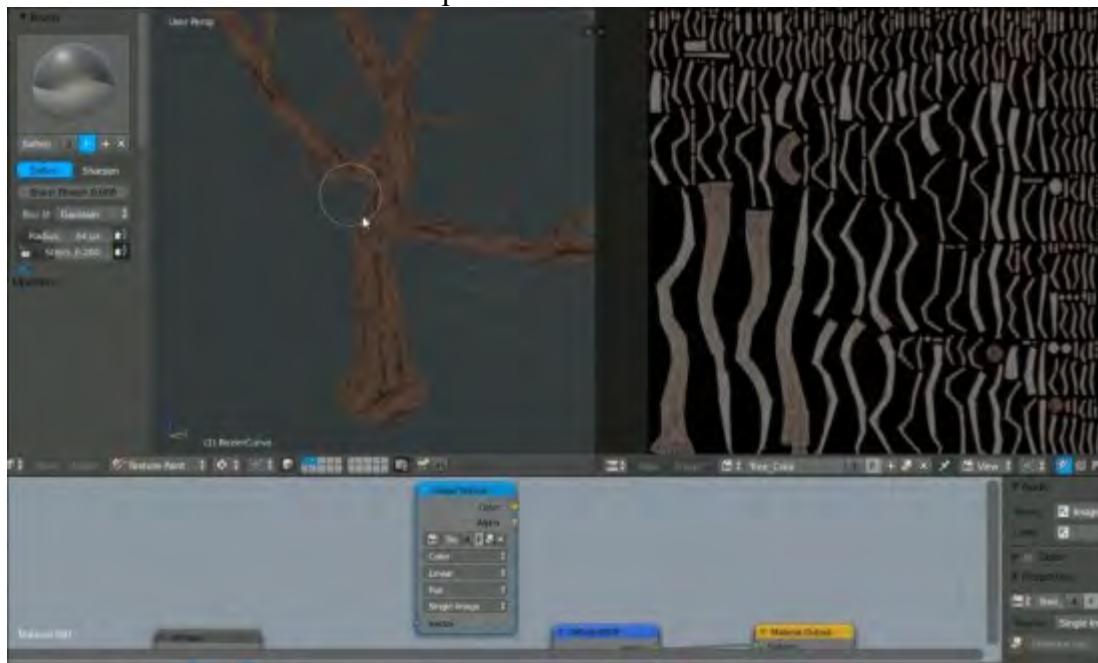
Baking the tree

We won't bake the maps of every objects in our scene with Cycles; however, we will see how to proceed with the 3D mesh of the tree as an example.

In order to optimize the render time, we will import our 3D mesh to another Blender window:

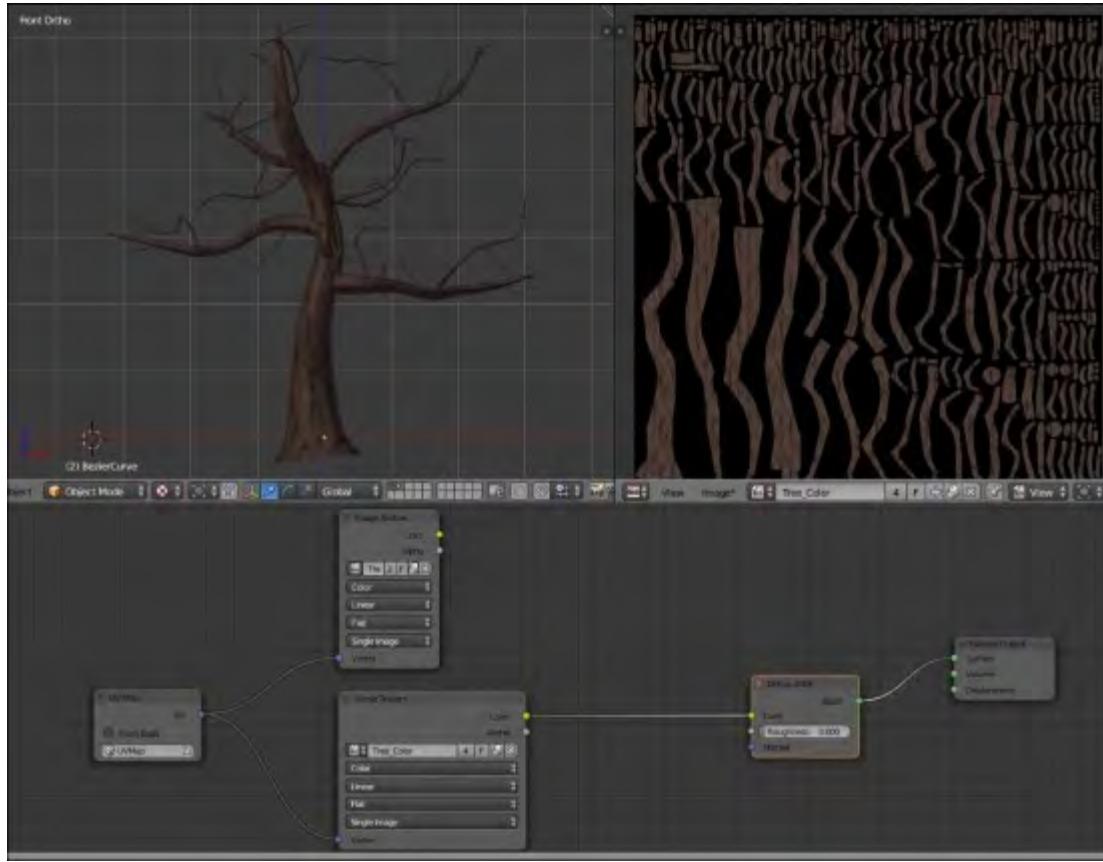
1. We will launch Blender a second time.
2. We will select the tree in the scene of the haunted house, and we will press *Ctrl + C* to copy it. You will see the **Copied selected object to buffer** message in the header of the work space.
3. In the other Blender window, we will press *Ctrl + V* to paste the tree. We will see the **Objects pasted from buffer** message.
4. In the same way, we will also import all the lights, and we must recreate the shader of the Image Base Lighting (we can append it from the main file). We don't modify the location of the tree and the lights in order to keep the same light configuration. But this wont be exactly the same lighting effect as we don't have the house here.
5. We will need a second UVs layer with UVs that are restrained in the UV square this time. We will use the *Ctrl + P* shortcut to automatically replace the UVs. Then we will adjust the margin in the options of the left panel of the 3D viewport.

6. We will start with color baking using the new UVs. For this, we will add an **Image Texture** node (*Shift + A* and select **Texture | Image Texture**) to the shader of the tree.
7. We will select all the polygons of the tree in the **Edit Mode** (*Tab* and *A*), and then we will create a new image named **Tree_Color**. A size of 2048 x 2048 is enough.
8. In the **Image texture** node that we have just created, we will select the **Tree_Color** texture. This node must stay unconnected.
9. Now we will need to go into the **Bake** tab in the **Render** options. Here, we will change the type to **Diffuse Color**. We will set the **Margin** value to **5**, and then we will press **Bake**.
10. When we have our color map, we must adjust the seams with **Texture Paint**. We will use the **Soften Brush** in order to blur the problems of the too visible seams.



Hiding the seams on the color bake

11. When the color texture is fine, we will again select the polygons of the tree in the **Edit Mode**, and we will create a new image (the same size) and rename it as **Tree_Combined**.
12. Now we can make another combined baking following the same process, which this time allows us to get all the lighting information on the texture. Make sure you open the good image in the **Image Texture** node with a high enough sample value. In our case, we have 500 samples to obviate the noise.
13. We can now go back to our haunted house scene and replace the tree with the one with new UVs (*Ctrl + C* and *Ctrl + V*); then we can replace the old texture with the **Tree_Combined** texture in our shader.



The combined bake of the tree

Compositing a mist pass

As a bonus, we are going to learn how we can create and composite a mist pass with Cycles. But to do this, we will need to do a render. So let's do a render with 500 samples:

1. We will first have to activate the mist pass in the **Render Layer** tab of the Properties editor. Now we can access the mist settings in the World setting panel. We will set **Start** to **0 m** and **Depth** to **37 m**.
2. In order to see the mist, we will need to composite it over the render. We will learn more about compositing in further chapters, so don't worry if we don't go deep into the subject right now. In the Node Editor, we will have to switch to the Compositing Node mode (the second button after the material in the header). We will need to check **Use Nodes** and **Backdrop** in order to see our changes in real time. As you can see, we already have a RenderLayers node plug in the **Composite** node (the final output of the image).
3. In between, we can add a **Mix** node (press *Shift + A* and select **Color | Mix**) and feed the first color input with the Image output of the **RenderLayers** node. The **Fac** input of the **Mix** node will receive a **Map Value** node (press *Shift + A* and select **Vector | Map Value**) with **Offset** of **0.105** and **Size** of **0.06**. For the input of Map Value, we will simply plug our **Mist** pass (the fourth output of the **RenderLayers** node). The **Map Value** will control the amount of mist we see.
4. In order to view the result in the **Node Editor** in real time, you will have to add a **Viewer** Node. To do this, we will simply press *Ctrl + Shift* and right-click on any node.



The final Cycles render of the Haunted House project

5. We now have a nice mist! In order to change the aspect of the final render, we can tweak the **Color Management** options as we did in the previous chapter. Congratulations, you've completed the Haunted House project.

Summary

This chapter was really robust, but you now understand how to create nice materials with the Cycles Render engine and how to light a scene properly. You also learned how to produce normal maps and how to bake your objects. This last technique would be very interesting for video games. We also covered Blender's Compositing tool to a slight extent by mixing a mist pass. Now let's create a new project!

Chapter 7. Rat Cowboy – Learning To Rig a Character for Animation

This chapter will cover the rigging and the skinning of a character. This character will be a Rat Cowboy that has been already modeled for you. Here, you will understand what the rigging process involves. We will start by placing deforming bones. After this, we will learn how to rig these bones with controllers and constraints such as IK or Copy. Then, we will skin our character so that the mesh follows the deforming bones. As a bonus, you will learn how to use shape keys in order to add some basic facial controls that will be controlled by drivers. The rig, which is covered here, will be basic, but you will have all the necessary knowledge to go further. We are going to use this rig to animate our character in the next chapter. Enjoy!

In this chapter, we will cover the following topics:

- Making a symmetric skeleton
- Using the basic bones constraints
- Rigging the eyes
- Correcting the deformation of the meshes with weight painting
- Improving the accessibility of the rig with custom shapes
- Using shape keys

An introduction to the rigging process

We are now going to discover the process of character rigging. The point of this is to prepare objects or characters for animation in order to pose them in a simple way. For instance, when rigging a biped character, we will place virtual bones that mimic the character's real skeleton. Those bones are going to have relationships between them. In the case of a finger, for instance, we will usually add three bones that follow the phalanges. The tip bone will be the child of the mid bone, which in turn will be the child of the top bone. So when we rotate the top bone, it will automatically rotate its children. On the top of the network of the bones, we will need to add some constraints that define automation so that it is easier for the animator to pose the character. The next step is to specify to the geometry to follow the bones in some way. For instance, in the case of a character, we will tell Blender to deform the mesh according to the deformable bones. This stage is called **Weight Painting** in Blender and **Skinning** is a common term, too. However, we will not always face a case where skinning is necessary. For instance, if you have to rig a car, you will not want to deform the wheels, so you will create a bone hierarchy or constraints in order to follow the rig. The entire process could be tricky at some point, but mastering the rigging process allows you to better understand the animation process and is the reason why having a good topology is so important.

Note

Anatomy of a bone in Blender

A bone has a root and a tip. The root corresponds to the pivot point of the bone, and the tip defines the length of the bone. Bones can have a parent-child relationship in two ways. The first method is by connecting them, so the root of the child is merged with the tip of its parent. The other method is by

telling Blender that they are visually disconnected while still having a parent/child relationship. Each bone has a roll that corresponds to its orientation on itself. When manipulating an **Armature** object, you can be in the **Edit Mode** to create the network of the bones and set their relationships, or you can be in the **Pose Mode** where you can pose the rig as if you were posing a marionette.



Rigging the Rat Cowboy

Let's do the rig of the Rat Cowboy. We are not going to show the modeling process here as you already know how to model proper characters from the Alien project.

Placing the deforming bones

The first thing that we will need to do for our rig is place the bones that will directly deform our mesh. These are the main bones. In Blender, a rig is contained in an **Armature** object, so let's go!

Let's begin with the process:

1. We will first be sure that our character is placed at the center of the scene with his feet on the *x* axis.
2. Now we can add a new bone that will be placed in an **Armature** object (press *Shift + A* and select **Armature | Bone**).
3. Next, we will enter the **Edit Mode** of our new **Armature** object, and we will place the bone in the hip location and rename it as **hips**. You can rename a bone in the right panel of the 3D view in the **Item** subpanel. Be careful to rename just the bone and not the **Armature** object.
4. We can now start to extrude the bones of the spine. To do this, we will select the tip of the hips and extrude it (*E*) twice as far as the base of the neck. It's very important that you don't move these bones on the *X* axis. We will rename the bones as **Spine01** and **Spine02** respectively. From the side view, be sure that these bones are slightly bent.
5. We will now extrude the left clavicle according to the Rat Cowboy's structure and rename it as **Clavicle.L**. The **.L** part is really important here because Blender will understand that this is on the left-hand side and will manage the right-hand side automatically later when mirroring the rig.
6. Now, from the tip of the clavicle, we will extrude the bones of the arm. Rename the two bones as **TopArm.L** and **Forearm.L**.
7. Now it's time to extrude the bone of the hand, starting from the tip of the forearm. Name this as **Hand.L**.

Note

Please note that if you want to follow the process step by step, you can download the starting file for this chapter on the Packt Publishing website.

8. In order to create the finger bones, we will start from a new chain and parent it back to the hand. This will allow us to have bones that are visually disconnected from their parents. To do this, we will place the 3D cursor near the base of the first finger and press *Shift + A*. Since you can only add bones when you are in the **Edit Mode** of the **Armature**, this will automatically create a new bone. We will orient it correctly and move its tip to the first phalange. We will extrude its tip to form the next two bones. It's important to place the bones right in the middle of the finger and on the phalanges so that the finger bends properly. Analyze the topology of the mesh to do this precisely. If you want, you can activate the Snap option (the magnet in the 3D view header) and change its mode from **Increment** to **Volume** to automatically place the bones according to the volume of the finger.

9. Then we will create the chains for the other fingers and the thumb and rename them as **Finger[Which finger]Top.L**, **Finger[Which finger]Mid.L**, and **Finger[Which finger]Tip.L**.
10. We will now need to re-parent them to the hand so, when the hand moves, the fingers follow. To do this, we will select the top bone of each finger (the root of each finger chain) and then select the hand bone (so that it is the active selection) while holding *Shift*, pressing *Ctrl + P*, and selecting **Keep Offset**. Keep Offset means that the bones are going to be parented but they will keep their original positions (that is, they are not connected to their parent).
11. We will now change our cursor location to the left thigh of our character and press *Shift + A*. We can then place the tip of this bone to the knee location. Also, check the side view and put this tip a little bit forward. We can then extrude a new bone from the knee to the ankle. Rename these bones as **Thigh.L** and **Bottom Leg.L**.
12. We can then extrude the foot and the toes. Name them as **Foot.L** and **Toes.L**.
13. The leg chain needs to be parented to the hips. This can be done with *Ctrl + P* and selecting **Keep Offset**. Remember to first select the child and then the parent while doing your selection for parenting.
14. Now we can add the bones of the tail. We will create a chain of bones starting from the back of the Rat Cowboy to the tip of the tail where the tips and roots of each bone are placed according to the topology of the mesh. Remember to rename the bones properly from **Tail01** to **Tail07**.
15. Then we will parent **Tail01** to **Hips** by pressing *Ctrl + P* and selecting **Keep Offset** so that the whole tail is attached to the rest of the body.
16. The last bones that we will need to extrude are the neck and the head. The neck starts from the tip of the **Spine01** bone and goes straight up along the Z axis. Then we will extrude the head bone from the neck tip. We will rename them as **Neck** and **Head**.

Now we will have to verify on which axis the bone will rotate. You can display the axes of the bones in the **Armature** tab of the **Properties** editor under the **Display** subpanel. We will need to adjust the roll of each bone (*Ctrl + R* in the **Edit Mode**) to align them along the *x* axis. You can test the rotations by going to the **Pose Mode** (*Ctrl + Tab*) and rotating the bones around their *x* local axis by pressing *R* and then pressing *X* twice. Beware, rolls are very important!

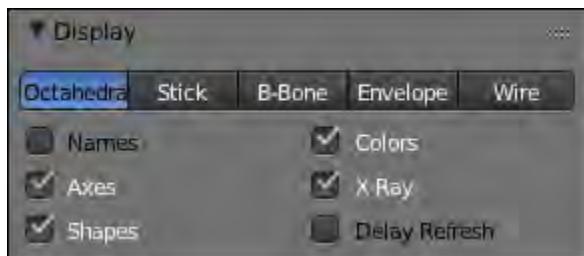


Placement of the deforming bones

Note

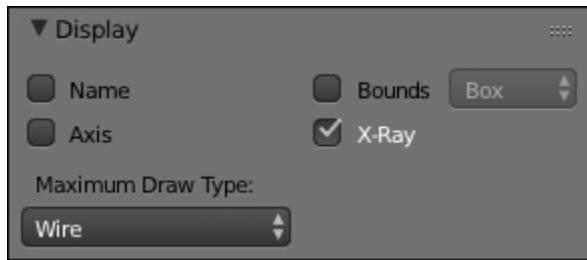
The Display options

You can find different display options for your bones in the **Display** subpanel in the **Object Data** tab of the **Properties** editor. A nice way to display the bones is to activate the **X-Ray** display mode, which allows us to see the bones through the mesh even in **Solid** shading mode. We can also display the axes of orientation and the name of each bone and change its shape.

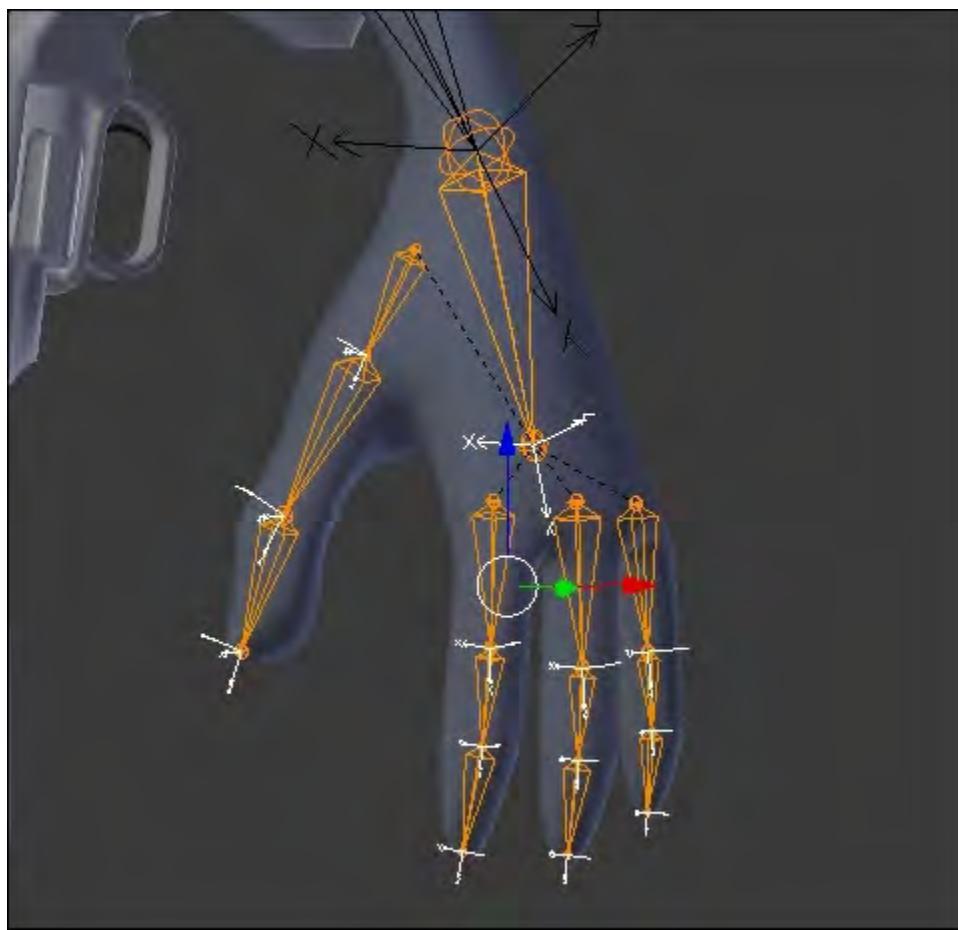


For instance, we can use the **B-Bone** mode that changes the bones to boxes that we can rescale with *Ctrl + Alt + S*. This is a nice way to display bones that are on top of each other. You can also use the

Maximum Draw Type drop-down menu in order to change the shading of your selected object in the **Display** subpanel in the **Object** tab of the **Properties** editor.



The following image will show the placement of the deforming bones of the hand:



Placement of the deforming bones of the hand with a correct roll

The leg and the foot

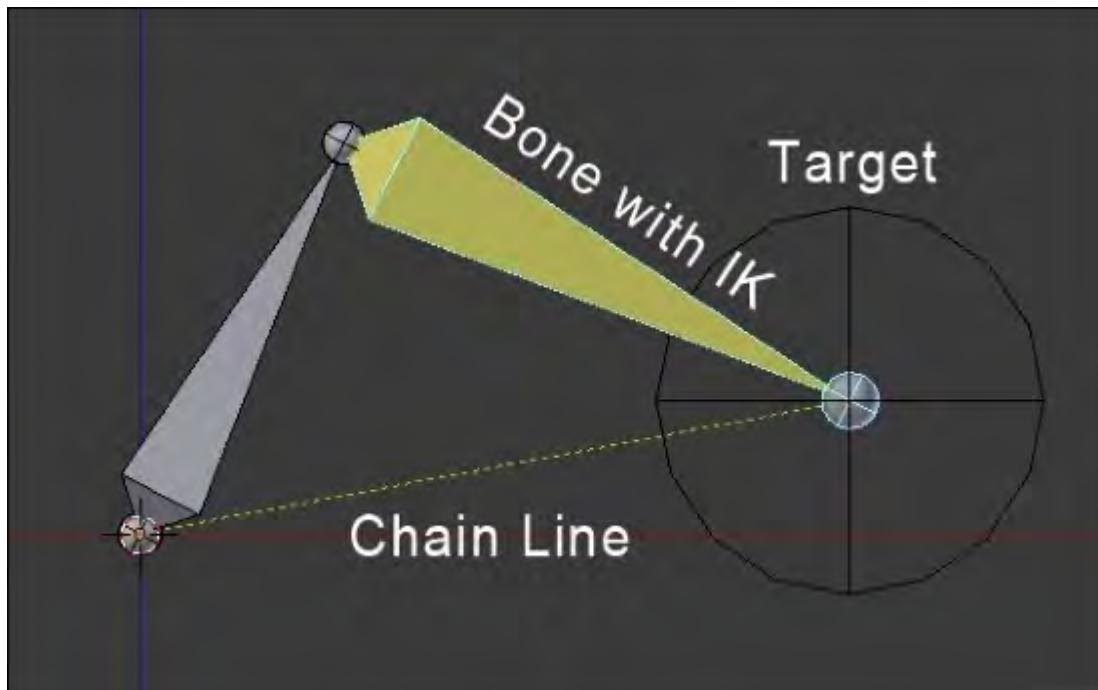
Now that we have all the deforming bones that are needed, we are going to add some bones that will help us to control the leg and the foot in a better way.

1. We will now add a bone that will be a controller for the **Inverse Kinematic (IK)** constraint of the leg. We add this to the ankle and align it with the floor. It's important that this bone is disconnected for now. We rename it as **LegIK.L**.
2. Under the **Bone** tab of the **Properties** editor, we will uncheck the **Deform** checkbox so that our bone does not deform our geometry later.

Note

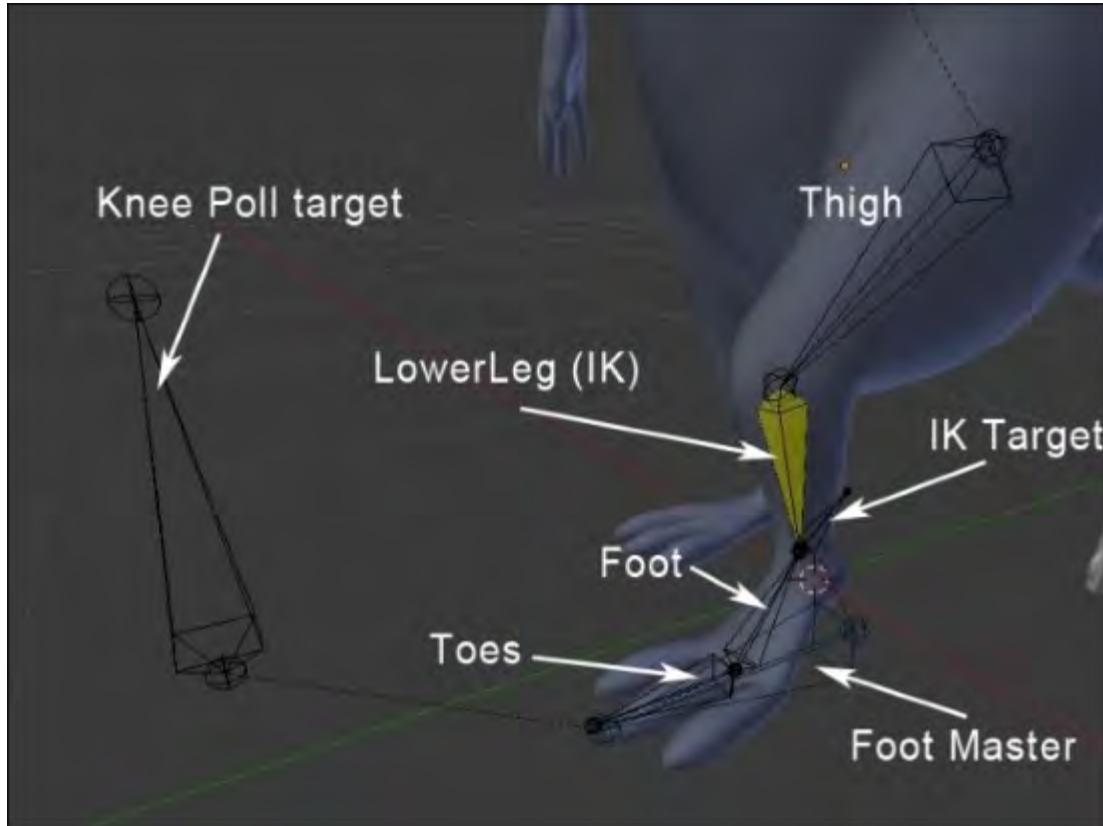
What is an IK constraint?

Usually, when you manipulate bones in the **Pose Mode**, you rotate each one in order to pose your object, and this method is called **FK (Forward Kinematic)**. The role of an **IK constraint** is to let Blender calculate the angle between a minimum of two bones according to a target. To better understand what this does, imagine that your foot is a 3D object and you can move it where you want in space. As you can see, your thigh and lower leg will automatically bend with an appropriate angle and direction. That's the whole point of **IK**!



3. Now we are going to tell Blender that this new bone is the target that will control the IK constraint. To do this, we will first select it in the pose mode (*Ctrl + Tab*) and then select the lower leg bone to make it the active bone. Now we can use the *Shift + I* shortcut to create a new IK constraint. As you can see, the lower leg bone turns yellow.

4. Now we will change the settings of the constraint. The bone Constraints panel is located in the **Properties** editor (a bone with a chain icon) in the **Pose Mode**. If we select the lower leg bone, we can see our IK constraint located here. As we've used the *Shift + I* shortcut, all the fields are already filled. The setting that we will change is **Chain Length**. We set this to two. This will tell Blender to calculate our IK constraint from the bone where the constraint is on the tip to the next bone in the leg chain.
5. We can now go back to the **Edit Mode** (*Tab*) and add a new floating bone in front of the knee location. This bone will be the target of the knee. Rename it as **PoleTargetKnee.L**.
6. Back in the **Pose Mode** (*Ctrl + Tab*), we will set this new bone as the **Pole Target** bone for the IK constraint. We will first select the **Armature** object and then the bone. After this, we will adjust **Pole Angle** to reorient the IK constraint so that the leg points to the knee target. In our case, it's set to **90°**.
7. The next thing to do is to uncheck the stretch option so that the leg can't be longer than it already is.
8. Now that we've rigged the leg, it's time to rig the foot. We are going to make an easy foot rig here. But note that a foot rig can be much more complex with a foot roll. In our case, we will first remove the **Foot.L** parentage. To do this, we go into the **Edit Mode**, select the parent, press *Alt + P*, and select **Clear parent**.
9. Now we want to switch the direction of the bone so that its root is located at the toes. To do that, we will select the bone in the **Edit Mode**, press *W*, and select **Flip Direction**. Remember that a bone rotates around its root, so this will give us the ability to lift the foot up on the character's toes.
10. Now we can connect the IK target to the foot bone. To do this, we will simply select the child (**LegIK.L**), select the parent (**Foot.L**), press *Ctrl + P*, and select **Connected**. So now, when we rotate the foot in the **Pose Mode**, the IK target is going to lift up and the IK constraint will do its job.
11. The last thing that we will need to do is create a master bone that will move all our foot bones. In the **Edit Mode**, we will add a bone that starts from the heel to the toes and rename it as **FootMaster.L**. We will then parent the toes to this with the **Keep Offset** option. Then we will parent the foot to the toes with the **Keep Offset** option. As you can see, if you move the master bone in the **Pose Mode**, all the bones will follow this. We are done with the foot and the leg rig!



The rigging of the foot and the leg

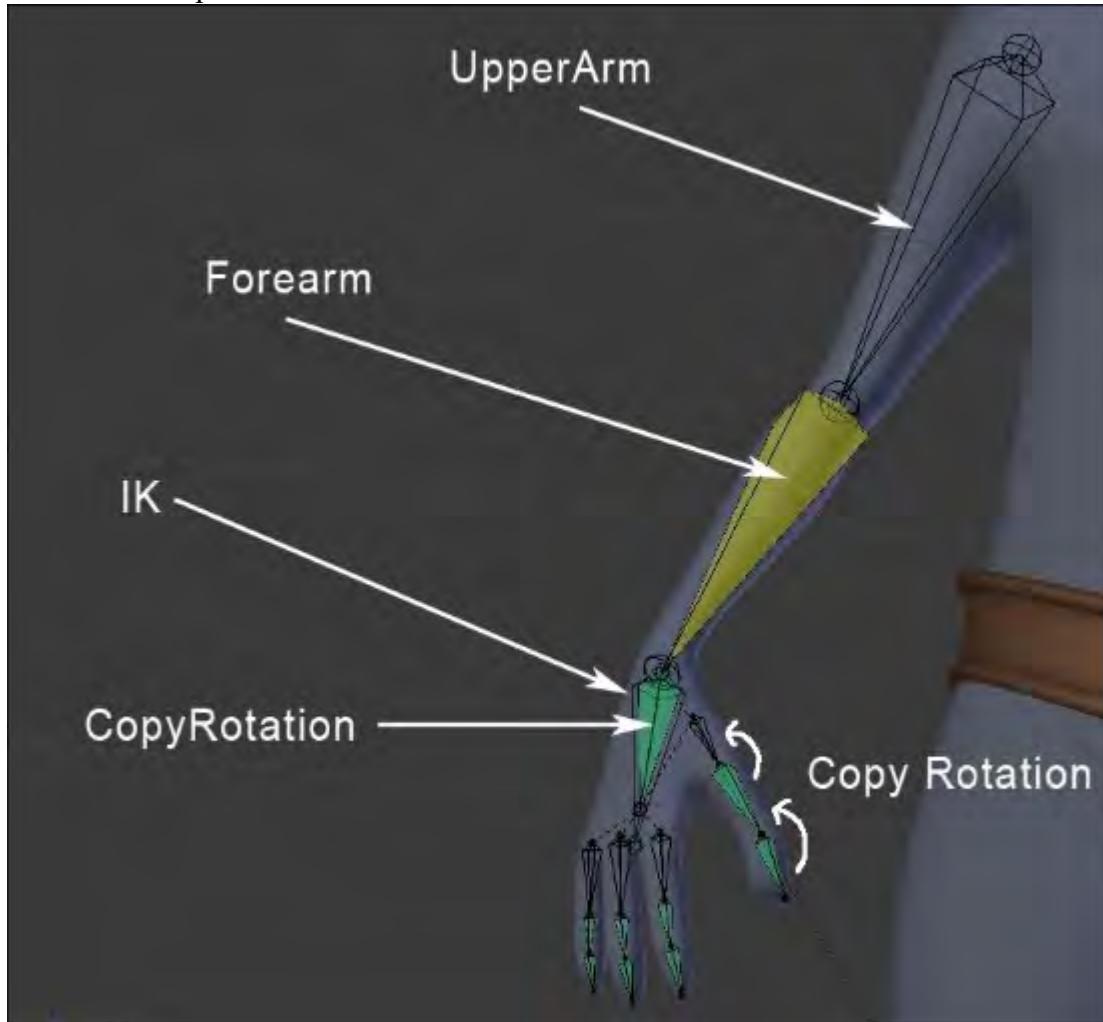
The arm and the hand

The next important part to rig is the arm. In many rigs, you will have a method to switch between FK and IK for the arms. In our case, we are only going to use IK because it will be quite long and boring to teach how to create a proper IK/FK switch with snaps. When animating the arm with IK, you will have to animate arcs by hand, but this will allow you much more control if you don't have an FK/IK switch.

1. We will create a new floating bone that will become our IK target for the arm by duplicating the bone of the hand and clearing its parent. Remember that the target of an IK switch needs to be freely movable! We will rename this bone as **HandIK.L**.
2. Then we will set up our IK constraint by first selecting the target, then the forearm, and then pressing *Shift + I*. Now, we can change the chain length to two as we did for the leg.
3. The next thing to do is add a poll target for the orientation of the elbow. To do this, we will create a floating bone behind the elbow of our left arm, we rename this as **ElbowPollTarget.L**, and we set this as the poll target of the IK constraint. Also, we will change the **Poll Angle** to match the correct orientation of the elbow.
4. Both the target and the IK target need to have the **Deform** option turned off.
5. The animator will only want to manage one bone for the hand. The bone that will deform the hand is not the target as it needs to be connected to the arm, so we will need to find a way to tell the hand-deforming bone to follow the IK target rotation. If this happens, the animator will only

control the target for both arm placement and the hand rotation. To do this, the IK target is going to transfer its rotation to the deform bone. So, we will first select the **HandIK.L** bone, then the **Hand.L** bone, and then press **Ctrl + Shift + C** to open the constraint floating menu and select **CopyRotation**.

6. We are now going to rig the fingers again with a **Copy Rotation** constraint. The motion that we want to achieve is that, when the base of the finger is rotated around the X local axis, the finger curls. To do this, we will indicate to the mid bone of the finger to copy the rotation of its parent (the top bone) and the tips to copy the rotation of its parent too (the mid bone). We will show the process for the index finger and let you do the rest for the others.
7. We will select the **Finger1Top.L** bone, then the **Finger1Mid.L** bone, press **Ctrl + Shift + C**, and select **CopyRotation**. After this, we will select the **Finger1Mid.L** bone, then the **Finger1Tip.L** bone, and create a copy rotation constraint. If we rotate **Finger1Top.L** on the X local axis, we can see that the finger bends.
8. In order to finish the hand, we will disable the **Y** and **Z** local rotations of the mid and tip bones of each finger. To disable a rotation on a particular axis, we will open the right panel of the 3D view (**N**) and change the rotation type from **Quaternion** to **XYZ Euler**. Then we can use the lock icon on a particular axis.



The rigging of the arm and the hand

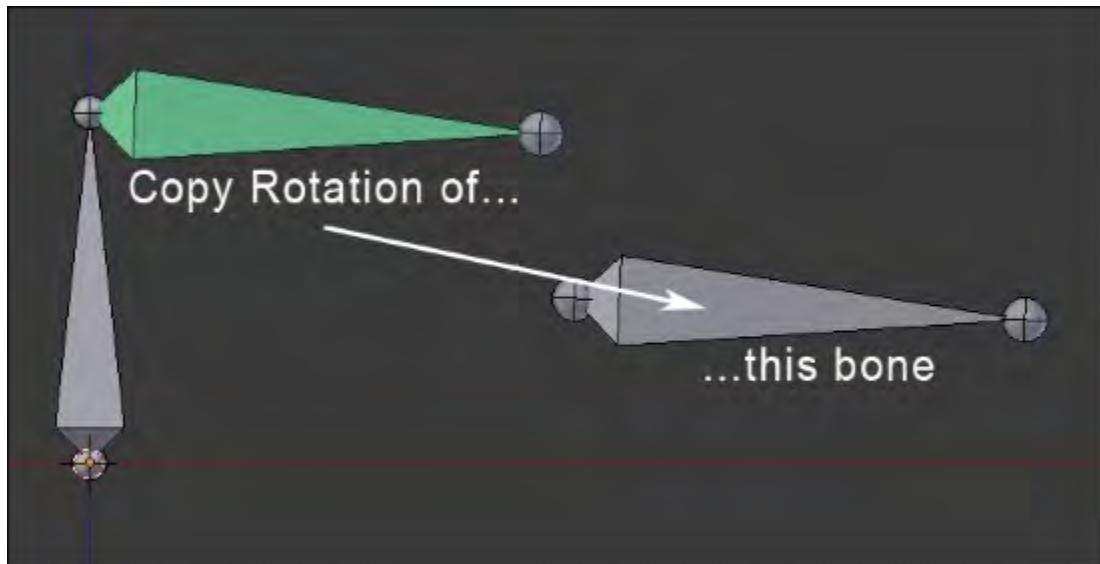
Note

What is a Copy Rotation constraint?

As its name implies, the copy rotation constraint will tell an entity to copy the rotation of another entity. The settings of the constraint allow you to choose in which space the rotation will be. The often used spaces are **World** or **Local**.

The **World** space has its axes aligned with the world (you can see them in the left corner of the 3D view).

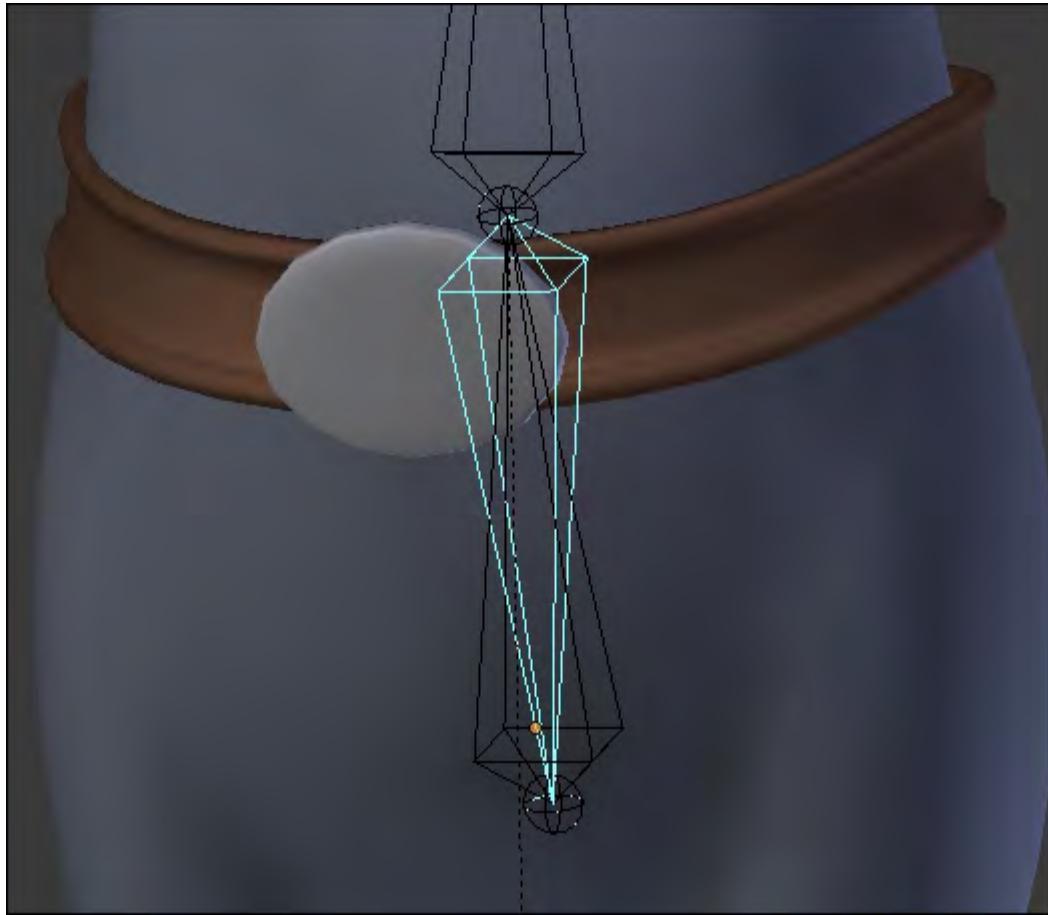
The **Local** space has to do with the orientation of an object. For instance, if an airplane has a certain direction, but when it rotates around its fuselage, this takes into account its orientation.



The hips

Now it's time to do the hips motion so that it is easier to control for animation. You have done a lot of work until here. Have yourself a cookie, you deserve it!

1. To create our hips motion, we will duplicate the hip bone. Also, remember to uncheck the **Deform** option. We will rename it as **HipsReverse**.
2. Now, we are going to flip the direction of this bone so that the hip deforming bone rotates around its tips (because the root of the hip's reverse bones will be here).
3. Now you can test in the **Pose Mode** that, when you rotate the **HipsReverse** bone, the hip's deform bone rotates with it.

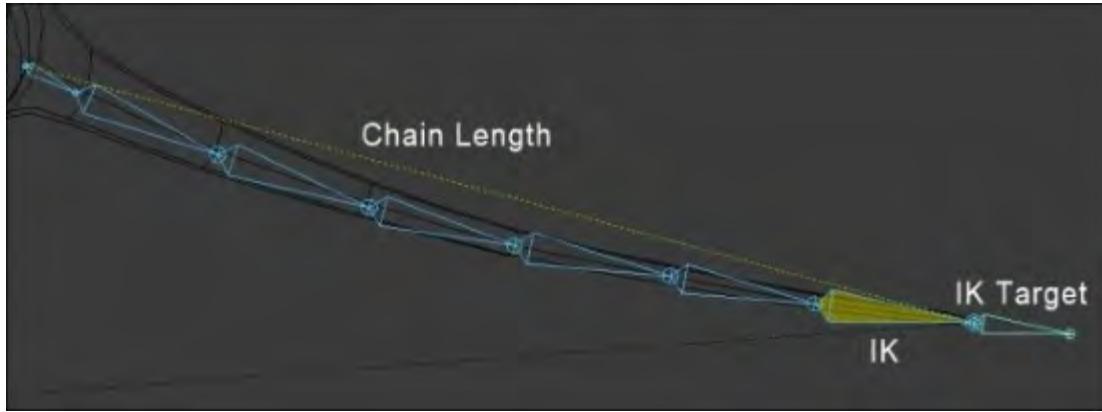


The rigging of the hips with the reversed bone

The tail

A rat without a tail is pretty strange, so we will take some time to rig his tail. The technique that we will show here is quite simple but very effective:

1. In order to rig the tail of the Rat Cowboy, we will need to add a new bone that controls this in the **Edit Mode**. To do this, we will extrude the last **Tail07** bone so that it is placed at the right location, and we will un-parent it with **Alt + P** and select **Clear Parent**. Rename this as **TailIK**.
2. We will now need to create an IK constraint. We will first select the target, then the **Tail07** bone (the last in the chain), and press **Shift + I**.
3. Now in the IK constraint settings, we will change the chain length to **7** (to let the IK constraint solve the angles from the tip to the last bone of the tail chain).
4. We will check the **Rotation** option, too. Now, as you can see, the tail is fully rigged and can be placed and rotated through the tail target:

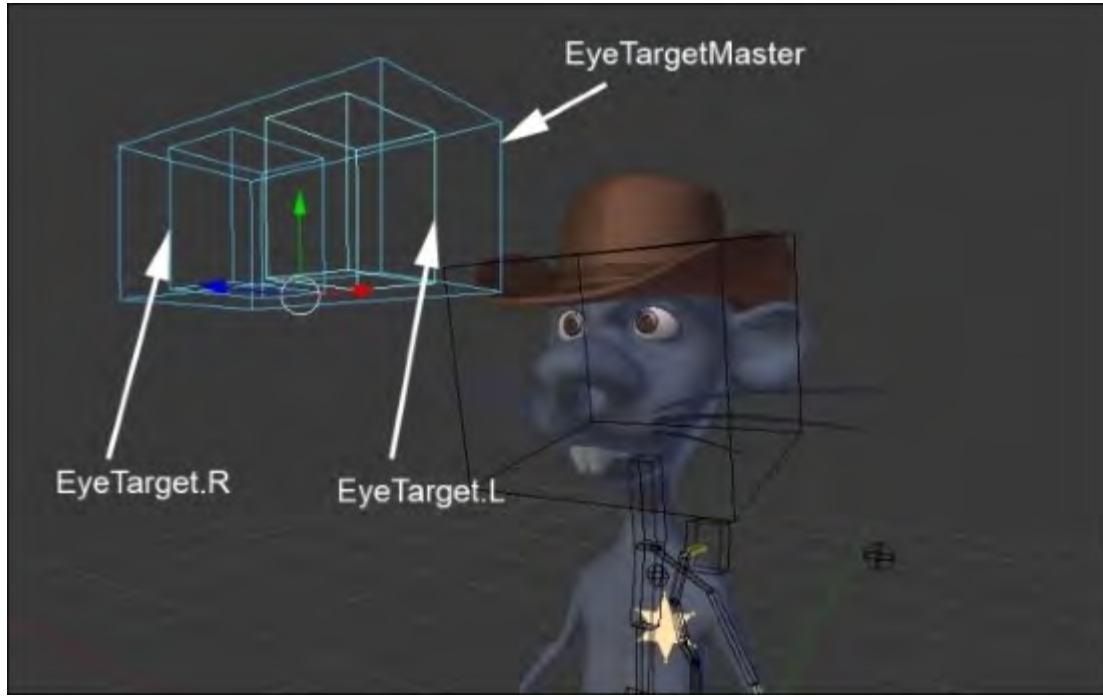


The rigging of the tail with a chain length of 7

The head and the eyes

In order to control the head, we will only manipulate the bone of the head, so we will directly begin the rigging of the eyes. We will ensure that the eyes are two separated objects and they have their pivot points at the center to make for good rotation. In our case, we have two half spheres, so we don't waste performance with hidden geometry:

1. We will select one of the eyes, and in the **Edit Mode** (*Tab*), we will select the outer edge loop. We will press *Shift + S* to open the *Snap* menu, then we will select the **Cursor to Selected** option, and then, in the **Object Mode** (*Tab*), we will press *Ctrl + Alt + Shift + C* and select **Origin to 3D cursor**. The pivot point must be at the right location. Do not hesitate to test this with a free rotation (**R x 2**) in the **Object Mode**. We must repeat the same process for the other eye.
2. In the **Object Mode**, we will select the eyes and the teeth, and then we will parent them to the head bone, but not with the usual method of parenting the bones that we saw earlier. To do this, we will first select the eyes and the teeth, and then the head bone (the armature must be in the **Pose Mode**). We will press *Ctrl + P*, and we will select the **Bone** option.
3. We now want to create a controller bone for each eye. We will select the armature, and in the **Edit Mode** (*Tab*), with the 3D Cursor in the middle of the eye, we will create a bone (*Shift + A*). In the Orthographic (5) left (3) view, we will move the controller bone in the front of the character. We need a small distance between the head and the bone controller. We will repeat the same process for the other eye, and we will rename them as **EyeTarget.L** and **EyeTarget.R**.
4. We will set the eyes to look at their controllers with a **Damped Track** constraint (**Properties | Constraint**). We will start with the left eye. We must select the Armature as **Target** and **EyeTarget.L** as the bone. Now, we must adjust the rotation by tweaking the **Z axis**.

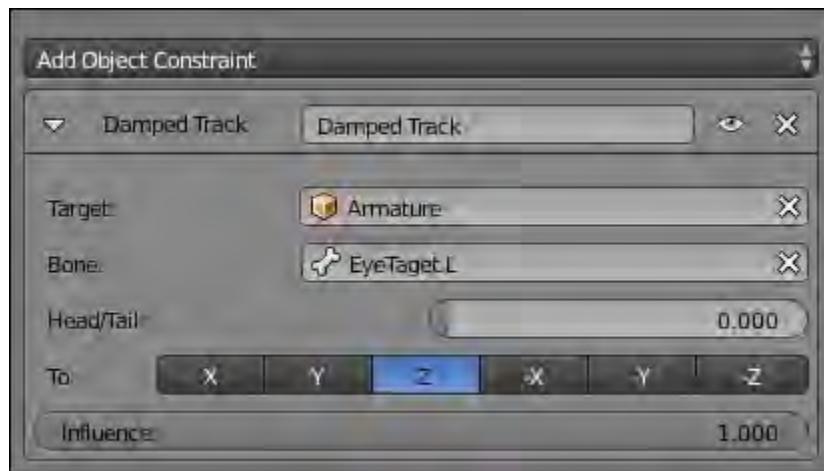


The eyes' controllers

Note

The Damped Track constraint

This allows us to constrain a 3D object to always point towards a target on an axis. The 3D object doesn't move, it just rotates on its pivot point depending on the location of the object.



Both the eyes must now follow the **EyeTargetMaster** movements. Do a test by pressing *G*.

Mirroring the rig

In order to save time, as in the modeling or the sculpting process, it is often very useful to work with symmetry. There are three ways to do this in an armature.

The first method consists of checking the **X-Axis Mirror** option in the **Armature Options** tab in the left panel of the 3D Viewport (*T*) that allows us to directly create the bones in a symmetry. We must extrude the bones by pressing *E* while also pressing *Shift*. This solution doesn't copy the constraints.

The second method is efficient even if it requires some manipulations to get a perfect mirror. Until then, we will place the bones of the arms and legs on the left-hand side with all the constraints that we need and the appropriate names:

1. We will select the armature in the **Edit Mode**, and we will align the 3D cursor at the center (*Shift + S* and select **Cursor to Center**).
2. In the **Header**, we will put the **Pivot Point** options on 3D Cursor.
3. Then, we will select every bone of the arm and the leg on the left-hand side.
4. We will duplicate them (*Shift + D*), and we will mirror them by pressing *S + X + I* on the numeric keyboard. Then we will press *Enter*.
5. In the **Edit Mode**, we will select the bones of the arm and the leg on the right-hand side and flip the names (**Armature | Flip Names**). This renames all the bones of the left-hand side with the **.R** termination.



Mirroring the bones with their constraints from the left to the right side

The third method is very interesting. It has been available since Blender 2.75.

Once we have placed and named all the bones with the **.L** termination and have all the constraints, we will select them in the **Edit Mode**, and then we will press **W** and select **Symmetrize**. This will automatically rename the bones of the right-hand side and the constraints will be copied.

Let's now focus on the gun for a rig.

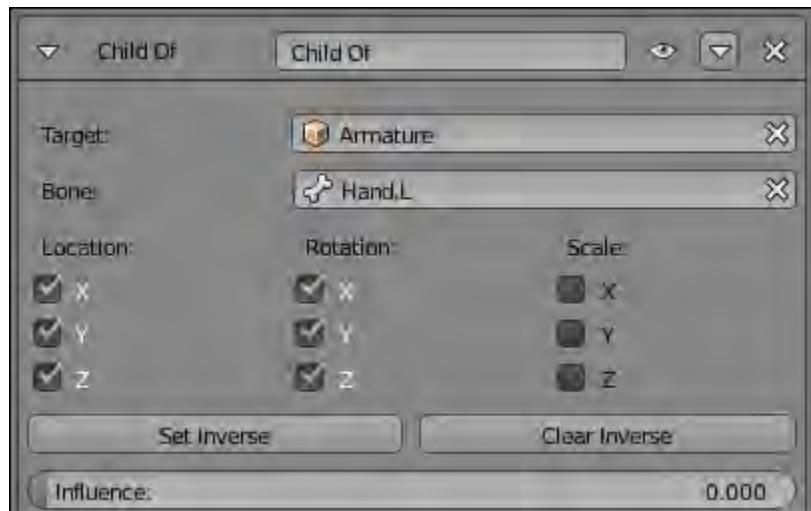
Rigging the gun

In order to easily animate the gun, we will need it to be able to follow the hand of our character when the Rat Cowboy uses it, and the gun must be able to follow the holster all the time. To do this, we will use a bone and a **Child Of** constraint.

Note

The Child Of constraint

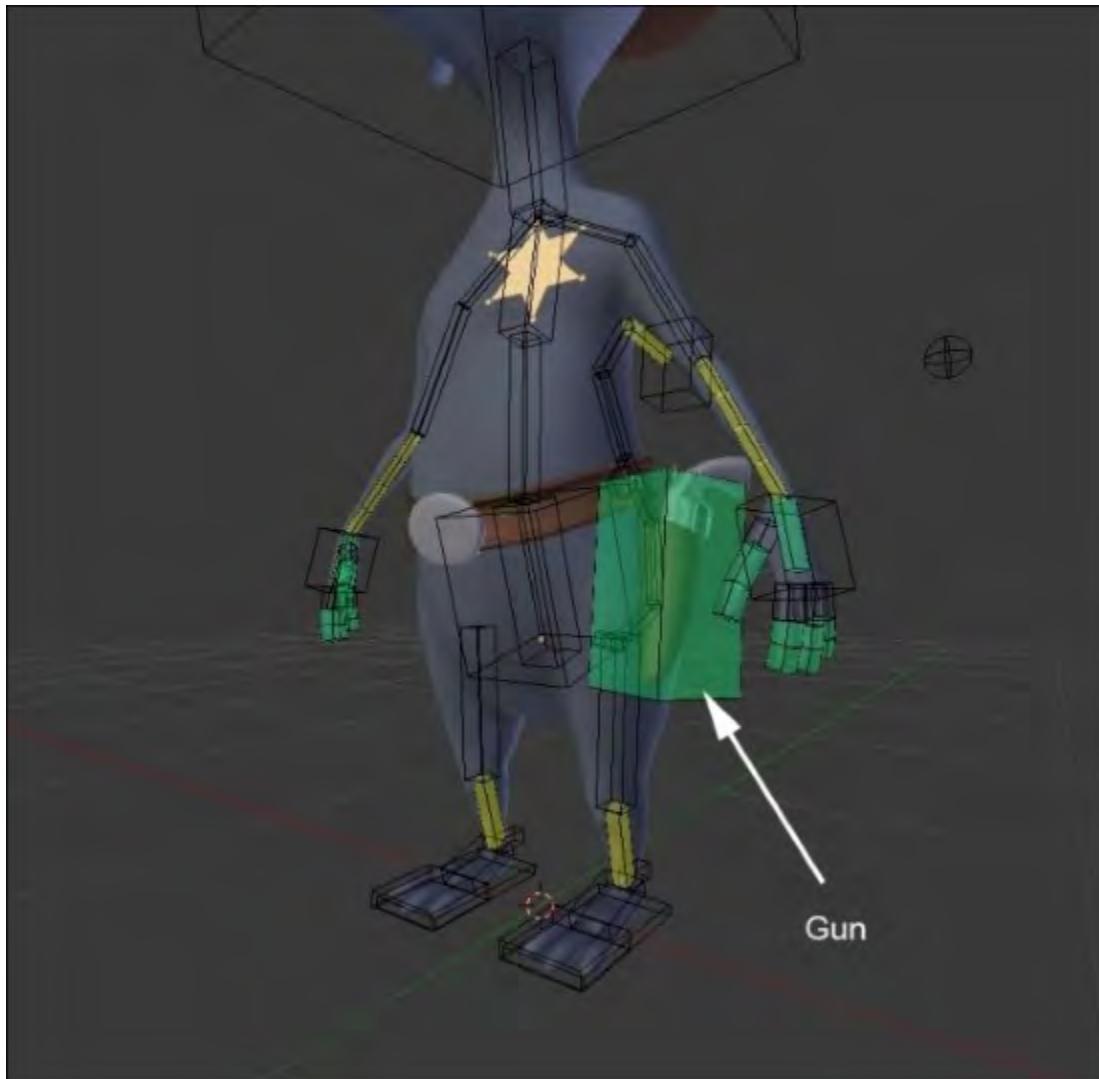
This constraint allows us to make an object a parent of another object by weighting their influences. This allows the animator to animate its influence in order to change its parent. This is much better than a classical parentation. This is very useful to make an object follow different objects one after another like a character driving with one hand on the steering wheel and the other hand on the speed box. You can also combine multiple children of the constraints.



1. We will begin by taking care to apply the rotation of the gun (press *Ctrl + A* and select **Scale and Rotate**).
2. We will select our Armature, and in the **Edit Mode**, we will place the 3D Cursor at the location of the hammer; then we will create a bone (*Shift + A*) that follows the length of the gun. We will rename this new bone as **Gun**.
3. We will make the gun the parent of the bone by first selecting the gun in the **Object Mode**, then the **Gun** bone in the **Pose Mode**, and then we will press *Ctrl + P* and select **Bone**.
4. We will then modify the appearance of the bone in B-Bone (**Properties | Object Data | Display | B-Bone**). We will scale it to make it easier to rig (*Ctrl + Alt + S*).
5. We will check the **X-Ray** option.
6. We will move the gun in the holster by moving and rotating the Gun bone. You can also directly rotate the gun a little bit in order to get the best position.
7. We will add two **Child Of** constraints to the Gun bone. We will uncheck the scale option of both the constraints.
8. We will decrease the influence to **0** on both the **Child Of** constraints.

9. With the first **Child Of** constraint, we will put our Armature as Target with the **Hand.L** bone in the bone option. We will press the **Set Inverse** button.
10. For the second **Child Of** constraint, we will put the **Holster** object as Target, and we will press the **Set Inverse** button again.

Now, when we place the left hand with **HandIK.L** near the gun and put the influence of the first **Child Of** constraint at **1.000** (the influence of the second Child Of constraint must still be at **0**), the gun joins the **HandIK.L** bone and follows it. In order to reposition the gun in the holster, it must be very near to the holster. The influence of the first Child Of constraint must be at **0**, and the influence of the second one must be at **1.0** (reversed influences).

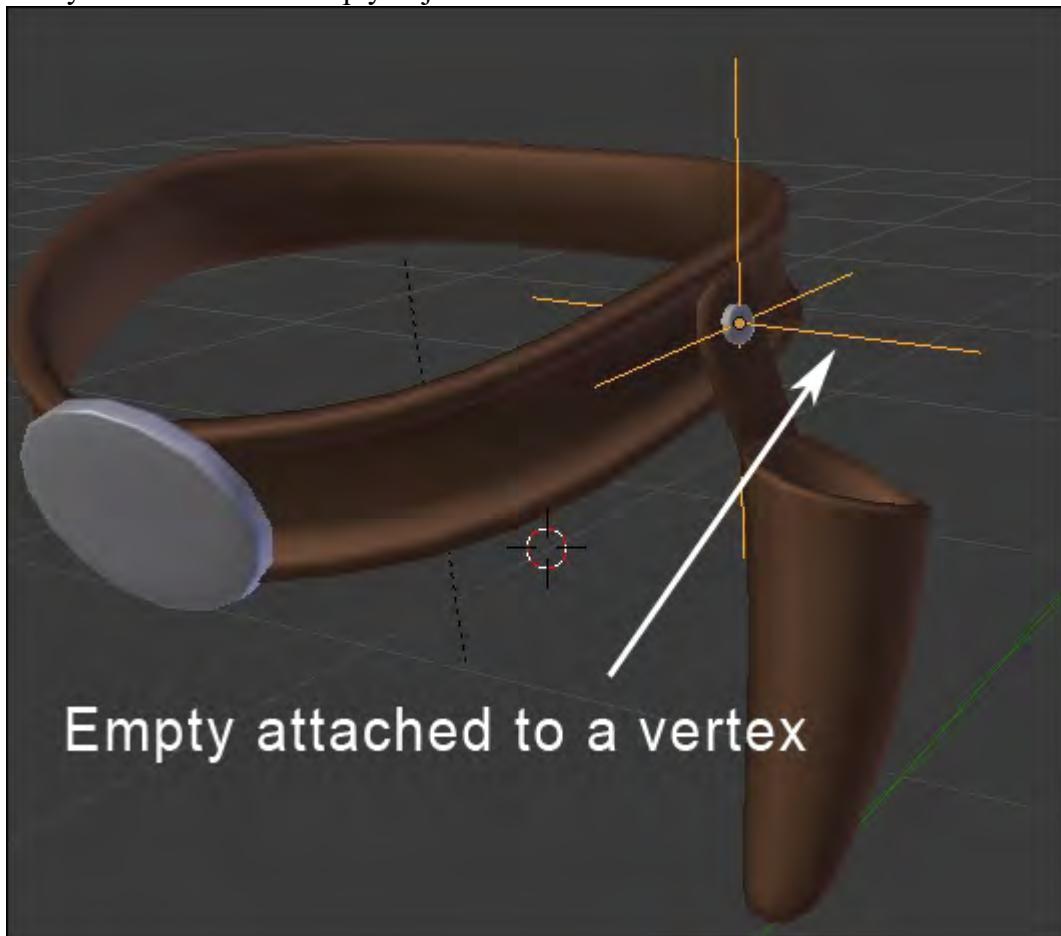


The gun bone

Rigging the holster

Now we are going to rig the holster. As you can see, it is a separate object. We will need to pin it to the belt. In this section, we won't use bones, but they are still a part of the rigging process.

1. First, what we are going to do here is create an empty object that will be the parent of the holster. To create an empty object, press *Shift + A* and select **Empty | Plain Axis**.
2. Now we will select the empty object, and while holding *Shift*, we will select the belt. We can now enter the **Edit Mode** of the belt and choose a vertex near the pin of the holster. We can press *Ctrl + P* and choose **Make Vertex Parent**. Now when we move the vertex, it moves the empty object!
3. The last thing that we need to do is make the holster the parent of the empty object, and the trick is done! Of course, we could have made the holster object a parent of the vertex directly, but it's always nice to have an empty object in between in this kind of situation.

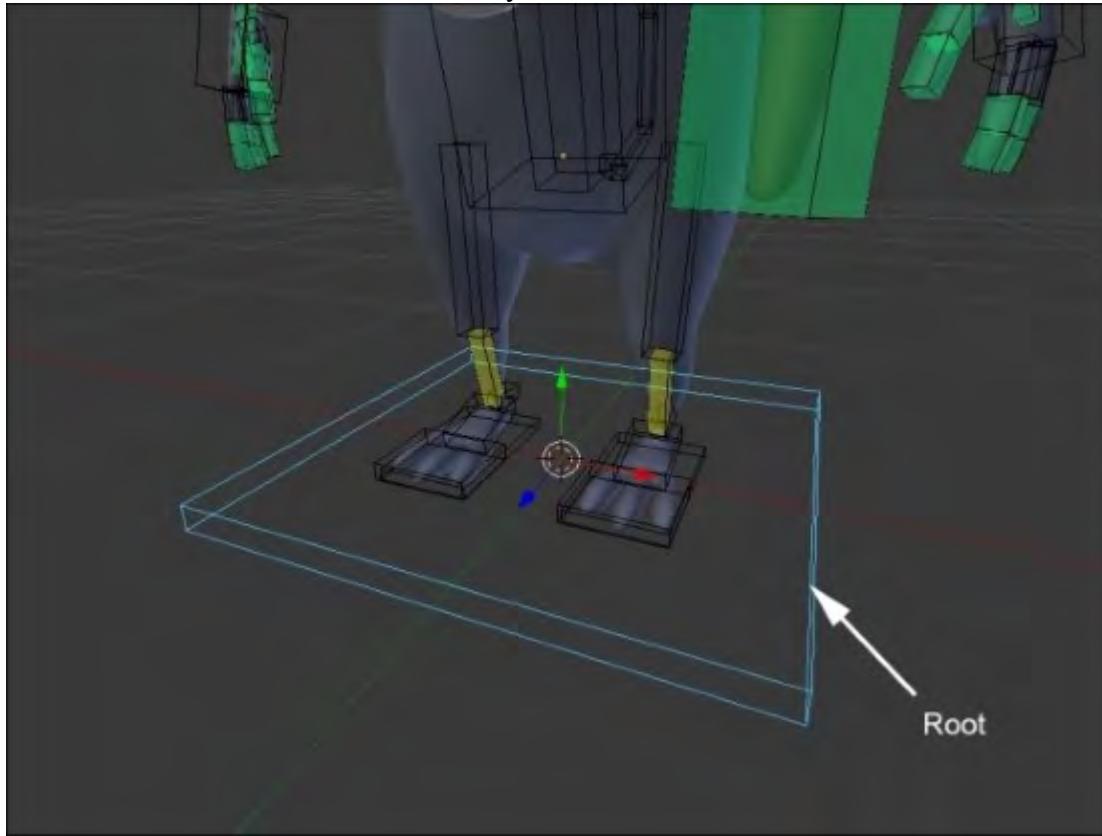


The rigging of the holster

Adding a root bone

The root bone is also called the master bone; it is the bone that will control the entire skeleton and is the top parent. With this, it is very convenient to place our character and animate it anywhere.

1. We will select the **Armature**, switch in the **Edit Mode (Tab)**, place the Pivot Point at the center of the world (press *Shift + S* and select **Cursor to Center**), and then we will add a bone (*Shift + A*).
2. We will make this bone bigger (*Ctrl + Alt + S* in the **B-Bone** mode) because it represents the central control element of the **Armature**. We will flatten it by selecting and moving the tip of the bone. We will rename it as **Root**.
3. In the **Edit Mode**, we will select the IK bone controllers of the **hands, feet, tail, EyeTargetMaster** that controls the **Eyes, Hips, and Pole Targets** at the location of the knees and the elbows; finally, we will select the master bone (so that it is the active selection).
4. We will make them parents (press *Ctrl + P* and select **Keep Offset**).
5. Remember to uncheck the **Deform** option (**Properties | Bone | Deform**).
6. You can see all the bones follow when you move the **Root Bone**.



The root bone at the center of the world

Skinning

Skinning is a very important step in the setup of a character for animation that will allow us to deform a mesh parented to a rig. It should be noted that the term skinning is not directly used in Blender. In Blender, you will generally find the term "Weight" designating the influence that a bone has on the geometry. It is often a long and delicate step, but fortunately, Blender allows us to perform an automatic skinning that is already very clean and quick. This is one of the most efficient skinning algorithms.

Before skinning our character, just as a reminder, we must determine which bones will deform the mesh and which not. This will be done as follows:

1. We will verify whether the **Deform** option is unchecked for all deforming bones (**Properties | Bone | Deform**).
2. In the **Object Mode**, we will select the mesh of the **Rat Cowboy**, then the Armature, and we will make them child and parent (press *Ctrl + P* and select **With Automatic Weight**).

A nice skinning has been done for us. You can make a few rotations on the rig in the **Pose Mode** to visualize the result.

However, we must do a few adjustments to improve this. Let's dive into the tools that we have at our disposal.

The Weight Paint tools

If we select our mesh and observe the **Data** menu of the **Properties** panel, we can see **Vertex Groups** that matches the bones of the Armature. This menu, in **Weight Paint** mode, allows us to select and view the influence of each bone. The **Weight Paint** mode could be activated directly on the object. If the armature is in the **Pose Mode**, it is possible to select the bones by a RMB click as well while we are in the **Weight Paint** mode of the object.

In the **Weight Paint** mode, we can view the influence that each bone has on the geometry with a color play. Blue means a 0% influence, a greenish-blue color means 25%, green means 50%, yellow means 75%, orange means 85%, and red means 100%.

In order to modify the influence, we have several brushes that can be used exactly in the same manner as **Texture Paint** and **Sculpt Mode** in the left panel of the 3D viewport. These brushes allow us to paint bones influence directly on the mesh. In our case, we will use only three brushes. The brushes that will serve us for sure will be the **Add**, **Subtract**, and **Blur** brushes.

- The Add brush allows us to increase the weights
- The Subtract brush allows us to reduce the weights
- The Blur brush allows us to soften and mix the weights

The options of the brushes are exactly the same as with **Texture Paint** and **Sculpt Mode**. We can change the radius of the brush by pressing *F* and moving the mouse. Also, we can change the strength of the brush that will completely modify the impact of the brush. You can change the curve, too.

We can paint the weight in symmetry. To do this, the mesh must be perfectly symmetrical. We will check the **X-Mirror** option in the **Option** tab of **Left Panel (T)**.

There are also a few useful options in the **Weight Tools** menu. For example, **Mirror**, to make a symmetric skinning and **Invert**, to invert the influence of our bones.

Note

For more information, you can have a look at the official Blender manual at this address:
https://www.blender.org/manual/modeling/meshes/vertex_groups/weight_paint_tools.html.



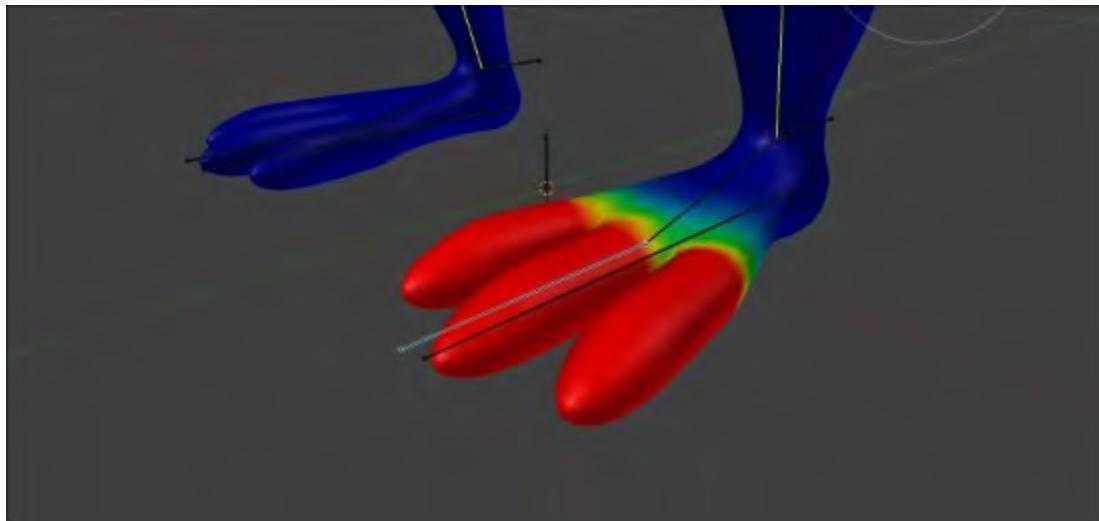
The weight paint of the Rat Cowboy (here the head influence is shown)

Manually assigning weight to vertices

Weight Paint is a very useful technique, but sometimes it is not very accurate. It can be useful to assign weight precisely on some geometry components.

To do this, we must be in the **Edit Mode** and select the vertices to which we want to assign a weight. Then we can go in the **Vertex Group** menu (**Properties | Data | Vertex Groups**). There will be a **Weight bar** from which we must select the desired weight and click on **Assign**. If you don't have any group, you can click on the + icon.

Another nice feature is located in the **Right Panel (N)**. There the **Vertex Weight** menu allows us to visualize the bones that influence the selected vertex and adjust the vertex directly. We will notice that the total weights assigned to a vertex group may exceed **1.000**. In fact, Blender will add the total of influences and make an average.



Correcting the weight paint of the toes.

Correcting the foot deformation

If you have used the **With Automatic Weight** option, you should have a very few things to change. The arms, legs, and head deformations should be quite good.

However, the toes aren't working well because there isn't a bone for each toe. So we will fix this as follows:

1. We will check the **X-Mirror** option, and we will select **Toe bone**.

2. Now, we can paint with **Add Brush** to add some weight to the toes until we get a red color (100%).
3. Then, we will need to remove the influence that the foot bone has on the toes. To do this, we will use the subtract brush. Now the foot shouldn't affect the toes.

Note

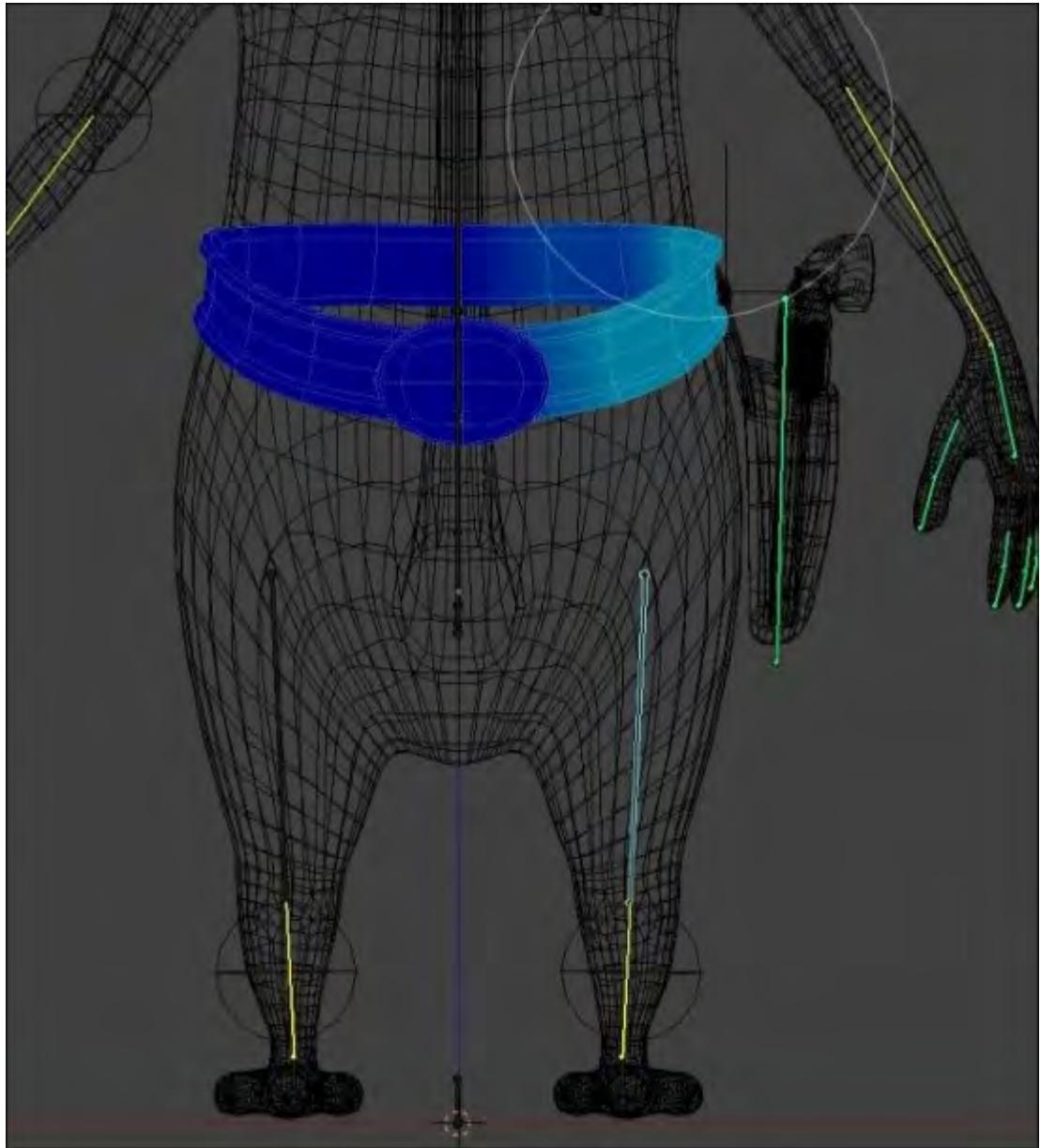
Stick Display for Weight Paint

To easily edit a skinning, it is advised to change the appearance of the bones to **Stick Mode** with the **X-Ray activated** option. You will get better visibility.

Correcting the belt deformation

Let's do the skinning of the belt. It is not a particularly easy element to skin because of its thickness and possible interpenetration with the body, but do not be discouraged. In this case, we will manually assign the weight to the vertices. This is shown in the following steps:

1. We will start by selecting the belt, then select the Armature, and we will make them child and parent (*Ctrl + P* and select **With Automatic Weight**).
2. In the **Vertex Groups** menu (**Properties** | **Data** | **Vertex Groups**), we will remove all the vertex groups except these: **Hips**, **Thigh.L**, and **Thigh.R** with the **-** button.
3. First, we will assign a weight of **1.000** to the **Hips** vertex group, so the hips bone deforms all the belt.
4. Then, we will select the vertices on the right half of the belt in the **Edit Mode**, and we will assign a weight of **0.2** to the **Thigh.R** group.
5. We will do the same thing for the vertices on the left half of the belt but with **Thigh.L**.
6. Then we will check the rotations of the **Belt** bone to verify some of the potential problems of transition. We will adjust the weight of a few vertices. The goal here is to create a gradient of weight at the center according to each thigh bone so that the thigh "attracts" some part of the belt in a smooth manner.



The weight of the left-hand side of the belt

Custom shapes

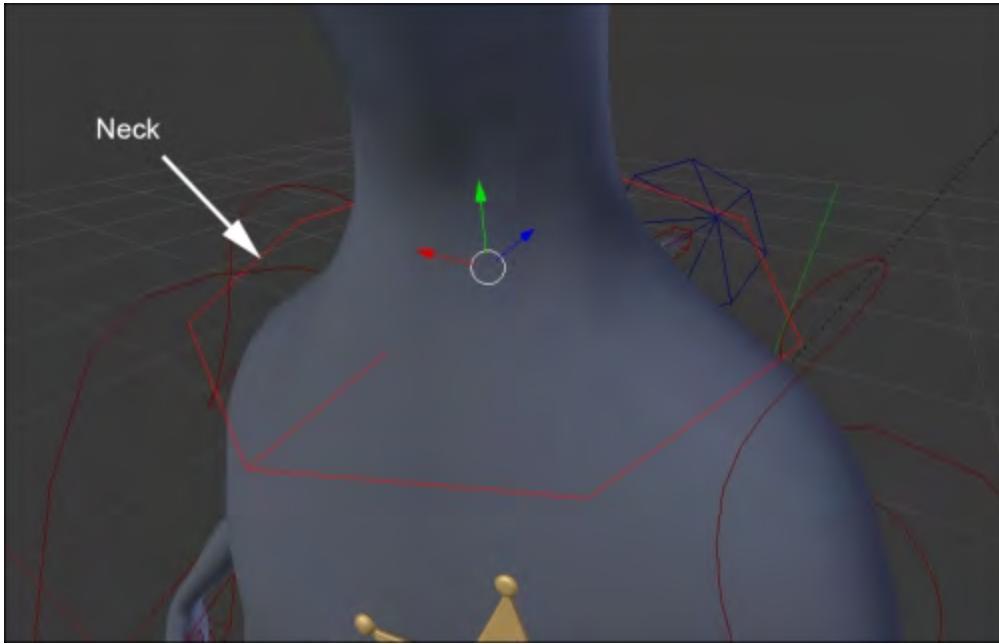
Now that our rigging is almost finished, we can opt for custom shapes. These are 3D objects that can replace the usual look of bones. The purpose is primarily aesthetic, but functional as well. We can make bone shapes that will go around the mesh and allow us to uncheck the X-ray option. Also, it will allow us to manipulate bones more easily.

We can hide the bones that do not require manipulation, such as the arms that are in fact only controlled by **HandIK.L** and **HandIK.R**.

To implement a custom shape, we must first to create a form, usually from a circle. We are going to make the custom shape of the **Neck** bone together and let you do the rest as this is very repetitive:

1. We will add a circle (eight sides is enough) to the scene (press *Shift + A* and select **Mesh | Circle**).
2. In the **Edit Mode**, we will select the opposite vertices on the Y axis, and we will connect them (*J*). This connection in the custom shape allows us to visualize the orientation of the bone better.
3. We will rename this object as **SHAPE_Circle_01**.
4. Now we will select the neck bone and **SHAPE_Circle_01** in the **Custom Shape** option (**Properties | Bone | Display | Custom Shape**).
5. There is often a problem with the axis of rotation and the scale. These must be adjusted in the **Edit Mode**. We can work on **SHAPE_Circle_01** again in the **Object Mode** without modifying the custom shape applied to the neck bone.
6. Once the custom shape is adjusted, we don't have to see the **SHAPE_Circle_01** object, so we move it to another layer (*M*). We will choose the last layer to the right in our case. This layer is usually used as a garbage layer where we put unwanted objects.

We redo the same process for almost every controller bone. Some shapes are very often used, such as glasses for the eyes, but try to find custom shapes that fit your needs. They must be explicit and made of very few vertices.



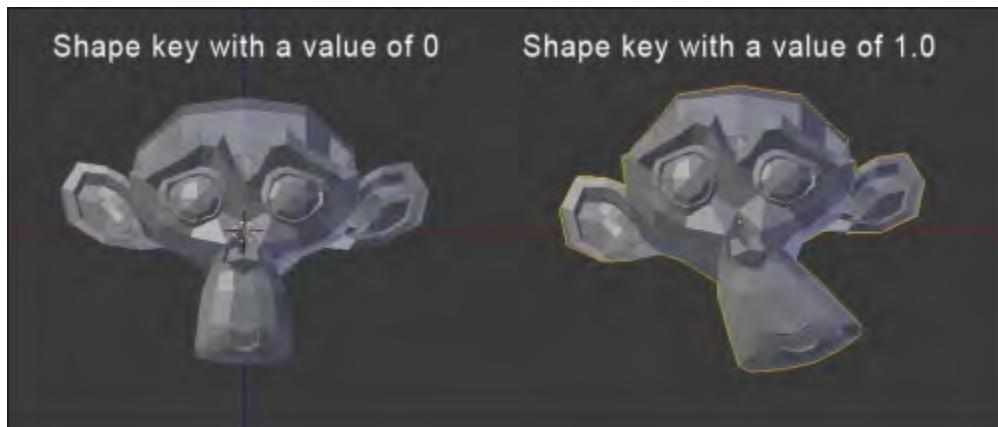
The custom shape of the neck bone

The shape keys

In the following sections, we are going to learn about shape keys and drivers. This will help us to create some very basic facial controls that we will use in the next chapter. Facial rigging is a long process, so we are not going to create a fully functioning facial rig here, but you will have all the tools needed to create your own if you want to.

What is a shape key?

A shape key is a method to store a change of geometry in a mesh. For instance, you can have a sphere object, add a shape key, move your vertices so that the sphere looks like a cube, and the shape key will store the changes for you. A shape key is controlled by a slider. The value of the slider corresponds to the distance each vertex has to move to get to the stored positions. As you can imagine, shape keys are very useful for facial rigging as they allow us to create different expressions and turn them on or off.



Example of a shape key with Suzanne

Creating basic shapes

In this section, we will create five basic shape keys. They are eye blink left and right, smile left and right, and frown. This will be done as follows:

1. First, we will need to select the object that will receive the shape keys. Then we will go to the **Object Data** tab of the **Properties** editor, and we will open the **Shape Keys** subpanel.
2. When creating the shape keys, we always need to have a reference key that will store the default position of each vertex. To do this, we will click on the + button. It is called **Basis** by default.
3. Now we can add a new shape key with the + icon and name it as **EyeBlink.L**. As you can see in the following, there is a **value** slider. At **0**, the shape key is turned off, so change the value to **1.0** in order to view your changes in the **Edit Mode**.
4. Now in the **Edit Mode**, we can change the left eye geometry to close the eye. You can use the **Connected Proportional Editing** tool (**Alt + O**) in order to smoothly move each eyelid to the

center of the eye. Beware, you only want to move the eyelid's geometry, otherwise any other changes done will be part of the shape key.

5. Next, we will mirror the shape key without tweaking our geometry on the right side again. We can do this because our mesh is perfectly symmetrical. We are going to create a new shape key based on the one that is currently active in the shape key stack. The value of **EyeBlink.L** is still at 1.0, so we can click on **black arrow** under the – icon and choose **New Shape From Mix**. Now, we have a new shape key with the same information as **EyeBlink.L**, but this not what we want. We will need to mirror this to the other side. To do this, we will again click on **black arrow**, and we will press **Mirror Shape Key**. We can also rename this as **EyeBlink.R**. Now if you change the value of **EyeBlink.R**, you can see that the right eye of the Rat Cowboy is closing perfectly. What a huge time-saver!
6. The next two keys, **Smile.L** and **Smile.R**, are created using the same method. What you need to do is create a nice smile on one side and mirror it using **New Shape From Mix** and **Mirror Shape Key**.
7. The last key to be created is a frown. To do this, we are just going to create a new key with the + icon and move the geometry between the eyebrows. There are tons of other things to know about shape keys, but for now this is all we are going to need.



Our facial shape keys

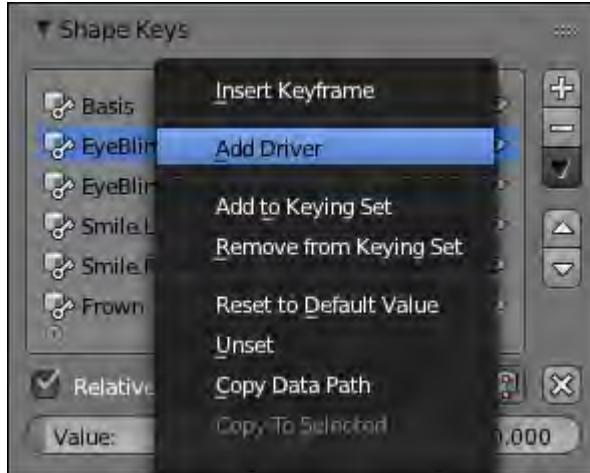
Driving a shape key

You might be thinking that animating directly shape key from the sliders is not very practical, and you are right! This is why we are going to use drivers. It is a method to indicate to an entity (object, bones, and so on) to control another value. In our case, we are going to tell Blender that the sliders of our shape keys are going to be controlled according to the transformations of the bones located on the face. This will give an impression that we are directly manipulating the head of our Rat Cowboy.

1. The first thing to do is add three new bones to our **Armature** object in the **Edit Mode**. The first one will be located between the two eyebrows, and this will control the frown. The others are going to be near the mouth corner and will be symmetrical. They are called **Frown**, **Smile.L**,

and **Smile.R** respectively. In order to control the eyes, we are going to use the bones we've used previously as targets.

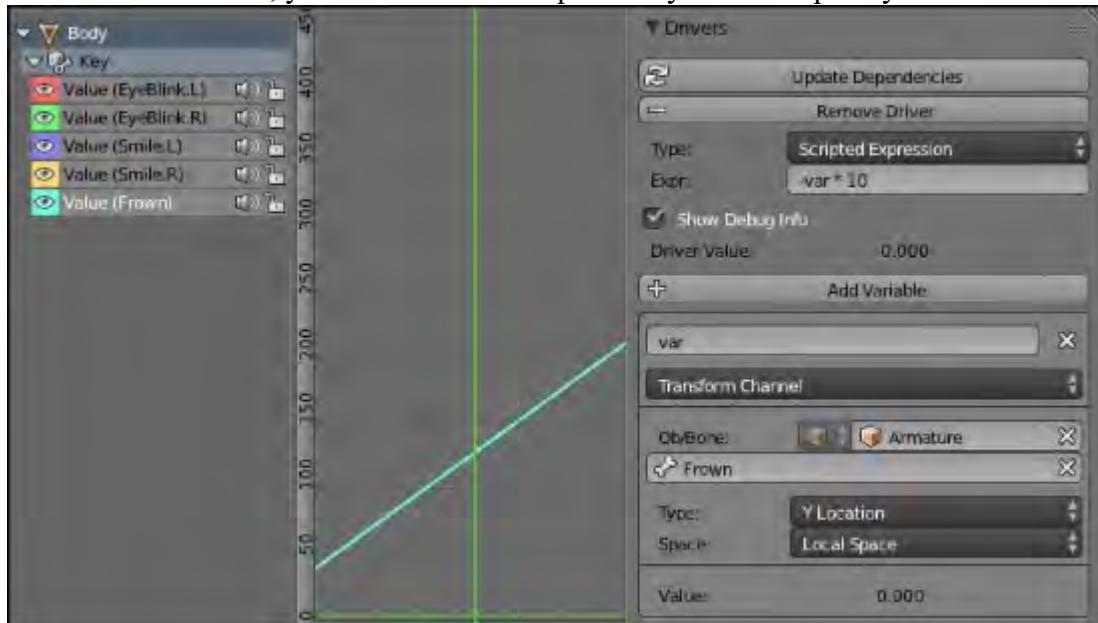
2. Now we will add a driver to the **EyeBlink.L** shape key by right-clicking on the value slider. Note that the value slider is now inaccessible, so it will be driven by another entity.



Adding a driver to a shape key

3. Now split your view in a new editor of the **Graph Editor** type. Switch the mode from **F-Curves** to **Drivers** in the Graph Editor's header. As you can see, if you still have the rat mesh selected, you will have a new key on the left-hand side of **Graph Editor**. We can click on the white arrow on the left-hand side to unfold the group of keys, and we can select the **EyeBlink.L** one.
4. Now we will open the right panel of **Graph Editor (N)**, and under the **Drivers** subpanel, we can change the different settings. First, if you have an error, you can go to the user preferences under the **File** tab and check **Auto Run Python Scripts**. Drivers work with Python internally, so you must accept that it runs scripts. But don't be afraid, we are not going to code here!
5. The first thing that we will change in our settings is the **Ob/Bone** field. We are going to choose the **EyeTarget.L** bone. So first choose the **Armature** object and then the bone. This will tell Blender that the target will be the bone that affects the driver.
6. Now we are going to set **Type** to **Y Scale** and **Space** to **Local**. This means that, when we scale our target, the value of the variable called **var** in the upper field will take the value of **Y Local scale** of the bone.
7. Now you need to imagine that the field called **Expr** is directly linked to the value slider of the shape key. You can even test this. If you enter 0, the eye will open, and if you enter 1, the eye will close. So now what we can do is replace this field with the **var** variable that holds the value of the y local scale. You can now see that the Rat Cowboy's eye is closed. But if you scale it, it will open. We are close to the required result.
8. The last thing to do is change the expression so that it is open by default. We know that the **Y local scale** of our bone is 1 by default and that the "open state" of the eye corresponds to the **0** value of the shape key slider. So we can add the **1-var** expression to the **Expr** field. If the Y local scale of the bone is 1, we will have **1-1 = 0** and the slider value will be 0, so the eye will be open. If the **Y local scale** of the bone is 0, we will have **1-0 = 1** and the slider value will be 1, so the eye will be closed. Done!

9. We can replicate the same process with the other eye. In order to save time when creating the driver, you can simply right-click on the **EyeBlink.L** value slider, press **Copy Driver**, right-click on the **EyeBlink.R**, and choose **Paste Driver**. At this point, you just need to change which bone is controlling the key in the driver settings.
10. For the smile driver, we will do a similar thing. But instead of feeding the variable with the Y local scale of the eye target bone, we are going to use the local Y location of the **Smile.L** bone. If you see a need to move the bone a lot in order to change the shape key, you can simply multiply **var** time a scalar in the **Expr** field. In our case, the expression will be **var * 5**.
11. We can do the same thing for the other **Smile.R** bone and for the frown. The expression of **Frown** is **-var * 10** in our case. As you can see, we have negated the **var** variable because we want to move the **Frown** bone down in order to control the shape key.
12. As a bonus, we are going to lift the hat with the frown. To do this, we will copy our Frown driver to the X rotation of the hat object in the right pane of the 3D view (*N*). We will adjust the expression. In our case, the expression is simply **-var**.
13. We will need to add a **Limit Location** constraint to the Frown bone as well. This will provide the animator with the ability to lift the hat high or low by locking the bone between a minimum and a maximum value on the Y axis in local space. To do this, we will change the settings of the constraint. We will check **Minimum Y** and set the value to **-0.115**, and then check **Maximum Y** and set its value to **0**. Also, we can provide moves on the X and Z location by checking the other check boxes and setting their value to **0**. Then we will need to select the **Local Space**.
14. Now, we have minimum control over the Rat Cowboy's face in order to start animating our character. We advise you to go further yourself in order to gain experience with shape keys and drivers. For instance, you could add cheek puff or eyebrow shape keys.



The setup of the Frown driver in Graph Editor

Summary

In this chapter, you've learned how to create a simple bipedal rig. All the knowledge that you've acquired could be used to push this further. For instance, you can add an FK/IK switch slider for the arms and the legs, add more facial controls, or create a more complex foot roll. If you want to see or use a more complex rig, you can check the rigify add-on (integrated in Blender by default). However, this rig will be quite interesting for animation. Talking about animation, let's start using our rig in the next chapter!

Chapter 8. Rat Cowboy – Animate a Full Sequence

This chapter will be devoted to the animation of a full sequence. We will begin our journey by discovering the 12 animation principles. Then, we will learn more about the preproduction stage that is all the things that we need in order to prepare the animation such as the writing of a script and the creation of a storyboard. After this, we will learn some important tools in order to animate in Blender such as the Timeline, Dope Sheet, Graph editor, and NLA. Next, we will create a layout that is a rough 3D visualization of the sequence without animation. After we've done all this, it will be time to start animating our shots. We will first learn how to animate a walk and use the NLA in order to mix actions together. We will then animate a close shot and a gunshot inspired by old western movies. The graph editor will be used extensively in order to animate a trap. Finally, we will learn how we can render a playblast of our shots. So let's dive into the wonderful world of animation where things start to move!

In this chapter, we will cover the following topics:

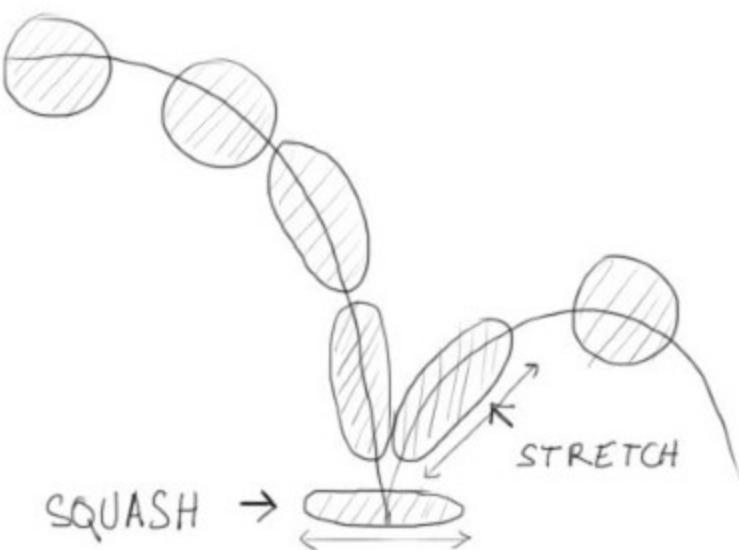
- Learning the principles of animation
- Preparing our animation with a script, storyboard, and layout
- Using the Blender animation tools
- Animate different shots
- Render a playblast

Principles of animation

In order to start to animate with Blender in the best way, it is important to understand some basic principles defined in the 80's by Ollie Johnston and Frank Thomas. These principles are inherited from the 2D animation art called "traditional animation". Animation involves recreating the illusion of motion by a sequence of images. Most of these principles also work for 3D animation. Here, they have been developed for a cartoon style, quite far from realistic movements. So, we don't have to apply them to any situation, but they still contain the secrets of animation.

Squash and Stretch

This is one of the common principles that applies to cartoon-style animation. The goal is to over-exaggerate the effect of inertia and elasticity on a particular object. From a 2D perspective, it's quite hard to manage because the object doesn't need to lose its volume, so we need to judge the shape by eye, but in 3D this is just a matter of a good rig.



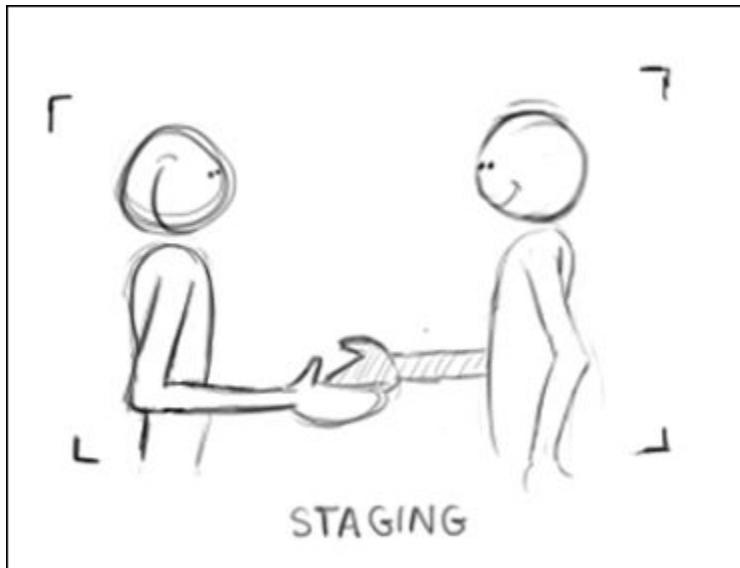
Anticipation

The principle of Anticipation describes the anticipation before an action. For example, a character ready to jump is going to bend the knees, the back, and the arms before the actual jump. It is important in order to add realism to the action you want to depict through your animation. To better understand this principle, try punching with your hand, you'll see your hand going back first.



Staging

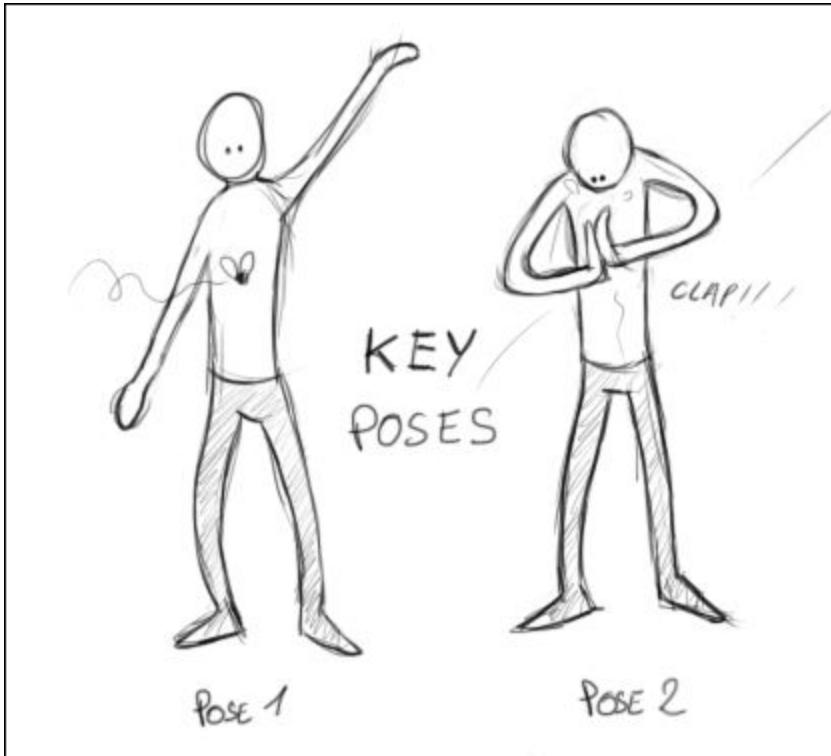
This principle is applied in cinema and theater to retain the attention of the spectator on specific elements and remove every useless detail from their view. This is useful to communicate a perfectly clear idea. This may include many areas of animation such as lighting, acting, or camera positions.



Straight Ahead Action and Pose to Pose

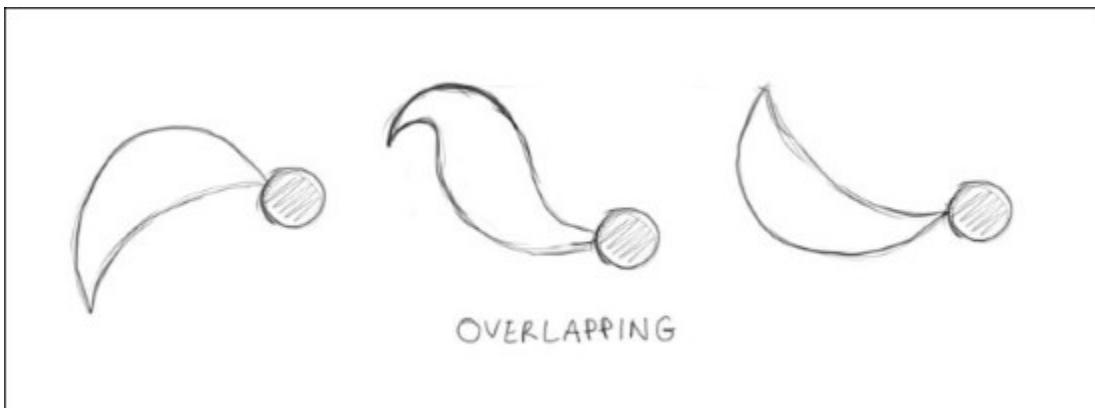
These are two different approaches while animating. The **Straight Ahead Action** method consists of making an animation gradually frame by frame from the beginning to the end. With 2D animation, this is especially useful to create special effects such as fire and water, and this allows improvisation.

The **Pose to Pose** method allows us a much better control of the timing and is easier to manage. This is often much more efficient when we animate characters. For this method, we start by adding the **Key Poses**. They are the main keys that indicate the action. Then comes the **Extremes** that we add to the extremities of motions that exaggerate the action between two key poses. Then, there are the **Breakdowns** that are the main intermediate keys of the action between the extremes. These add more fluidity to the action. A mix of the both methods is often used.



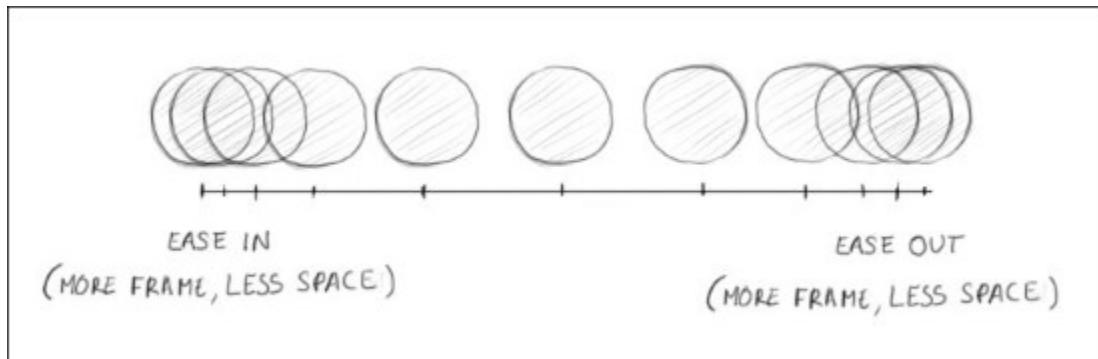
Follow Through and Overlapping Action

Follow Through and Overlapping Action techniques being very close consist of giving some inertia to an animated object and add a better sense of realism. **Follow Through** consists of continuing the movement of a part of an object after it has stopped moving. This can be applied to a tail, for instance. **Overlapping Action** consists of creating an offset between a movement of an object and a part of this object. For example, long hair moves at a different speed than the head.



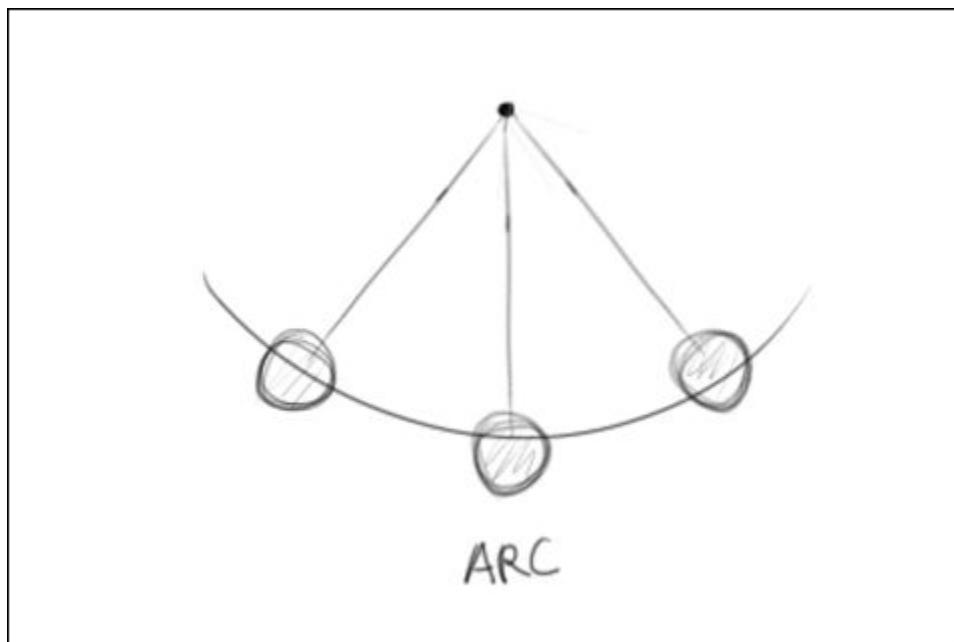
Slow In and Slow Out

These are two effects that consist of attenuating the speed of a moving object around the extreme keys. A Slow In effect is a deceleration at the beginning of an action, and a Slow Out effect is a deceleration at the end of an action. In Blender, these effects can be seen in the form of Bezier curves and are controlled by their handles in the Graph Editor.



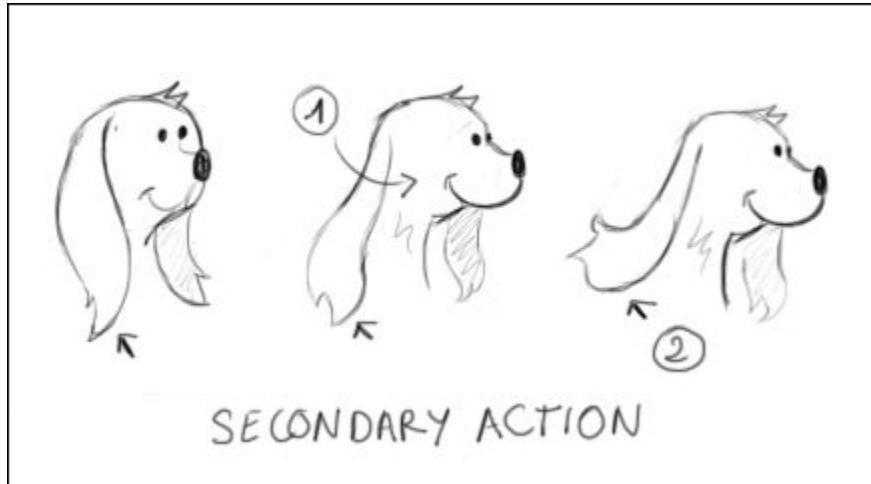
Arcs

This is a principle that consists of creating movements that follow an arc trajectory. Almost every motion follows this principle. For instance, when a character throws a ball, his or her hand follows an arc trajectory. If you take a pendulum and fix the ball, you will see that it follows an arc.



Secondary Action

The Secondary Action principle is about how every little action adds dimension to the main action. For example, in the case of a facial expression, a blink of the eyes can add more expression to the character. Another example could be the ears of a dog that hangs when he turns his head.



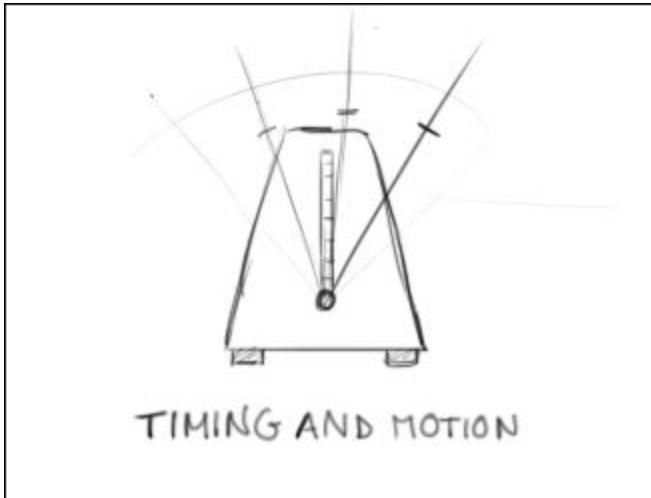
Timing

The number of frames between the beginning and end of an action directly affects the speed of the action. Timing is related to a physical consistency respecting certain laws of physics. In order to animate a character, timing defines its personality and emotions. For example, if a character is sad, timing is going to be definitely slower.

Note

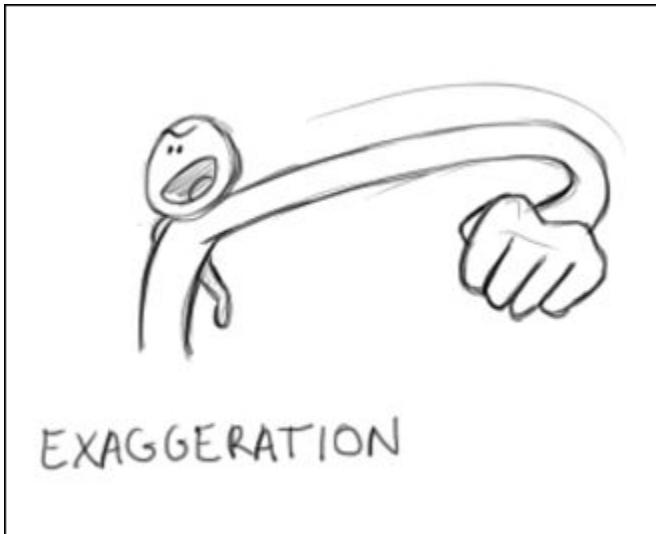
24 images per second

The frame rate in cinema is 24 images per second, and this is enough for the eye to have an impression of fluidity. This frame rate is used in the basic parameters of Blender.



Exaggeration

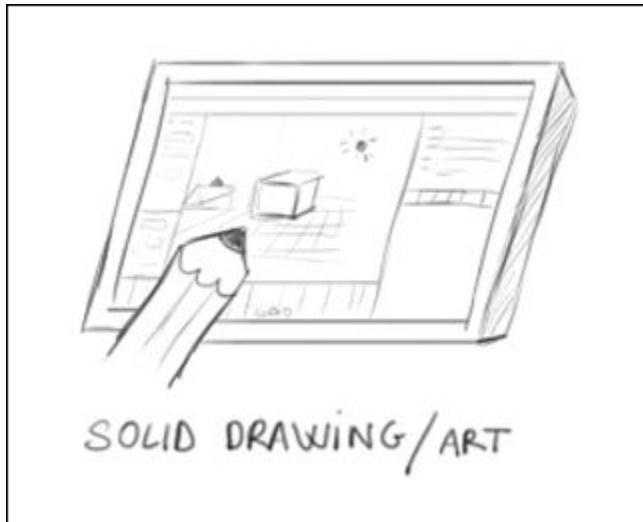
The exaggeration principle is used when you want to accentuate the action that you are transmitting to the viewer. The action can be a pose, a gesture, or an expression. Sometimes it is better to slightly get away from realistic animation to get a better impact. For instance, if you want to accentuate a punch, you can exaggerate the proportions of the arm of a character.



Solid drawing

This principle mostly applies to 2D animation. It consists of ensuring certain realism in drawing regarding volume, weight, and balance. This advocates paying attention to volume and perspective in order to avoid a flat render. A character must be possibly seen from any angle of view. In 3D animation,

it is a bit different. Even for a 3D animator, drawing can be seen as a strength, because it allows to quickly put the poses you have in mind on paper, but this is not going to be your main tool.



Appeal

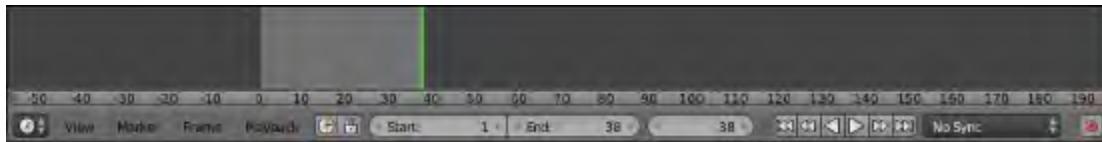
This principle is mainly about the interest and appearance of the characters you will animate. It means making a dynamic design through shapes, colors, proportions, gestures, and personality. It can be a hero or a villain, whatever.

Animation tools in Blender

Now that you understand a little bit more about what animation involves, we are going to dive into the different tools that we are going to use when animating our sequence. In order to test these tools, we are going to use the default cube of a new .blend file.

The timeline

The timeline editor gives us a lot of information about the animations of our file. The timeline represents each frame as you can see in its lower part. You can navigate to any frame by dragging the green bar. Using the Left and Right arrows, we will move one frame at a time. Using *Shift* and Up or Down arrows, we will move by an increment of ten frames. In the header we have two sliders, Start and End, which respectively represent at which frame the animation starts and ends. As you can see, we have dark grey parts in the timeline that visually shows this range. If you want, you can set the start and end frame by placing the timeline bar where they need to be and by pressing *S* and *E* respectively. You can quickly move the timeline bar to the start and end frame by pressing *Shift* and left or right mouse buttons. You can also zoom in to the frame with the Mouse Wheel. You can use the well-known play, rewind, and pause buttons in order to play the animation. The shortcut to play the animation is *Alt + A*.



The timeline

What is a keyframe?

Now it's time to learn how to set keyframes. But wait a minute, what is a keyframe? It is used to store the state of an object (or any other animatable thing) at a certain frame. Let's add a keyframe to our default cube:

1. We will open a fresh new blend file, and we will select the default cube.
2. We will then place the timeline bar at frame 0 and press *I* to open the **Insert Keyframe Menu**. We can now choose between many options, but we are going to choose the **LocRotScale** one because we are going to keyframe the location, rotation, and scale of our object.
3. Now we will go to frame 20 (remember the *Shift + Up arrow* shortcut of the timeline), and we will move, scale, and rotate our cube. Now we can add a new key to store the current state of the cube, press *I* and select **LocRotScale**.
4. If you move the timeline bar or play the animation, you can see that Blender automatically interpolates the motion for you! You can clearly see that you have two keyframes represented in yellow. Congratulations, this is your first animation in Blender.

Note

The AutoKey option

In the timeline header, we have a nice little red circle button that allows us to automatically create a keyframe as soon as we change the location, rotation, or scale of our selected object. Thus we don't have to call the Insert Keyframe menu every time.

The Dope Sheet

The timeline is great but is not very useful when we want to manipulate our keys. To do so, we can use a more robust editor called **Dope Sheet** (as always you can split your view and select the editor type that you want on the left-hand side of the header). On the left-hand side, we can see every object that has keys on it. In the header, we have many important options such as the **Show only selected** button (a mouse pointer icon) that tells Blender to only show the keyframe of the selected objects. The keys are represented by a diamond shape and can be selected with by right-clicking on them or by using the *B* key for the box select tool. Of course, you can select all the keys by pressing *A*. You can move the keys with *G*. You can also scale a group of keys with *S*. In this case, the timeline bar will be the pivot point. The **Dope Sheet Summary** row enables us to select every key that is below the corresponding key.

For instance, if we select the dope sheet summary key at frame 0 by right-clicking on it, we will automatically select every key on frame 0.



The Dope Sheet editor

The Graph editor

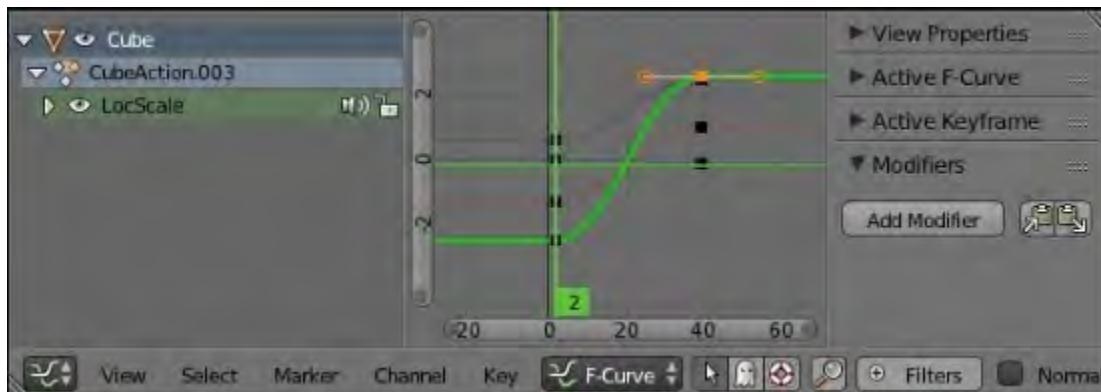
Now we will learn a little bit more about curves with the **Graph** editor. This is a really important editor to learn in order to become a good animator. It mainly allows us to control the interpolation between the keys. It looks like graph paper with the *y* axis corresponding to the value of the key and the *x* axis to the time. We can select the keys in the same way as the Dope Sheet. As you can see on the left-hand side, we still have the objects on which we added the keyframes. We can also see that we have the type of the keyframe that we've placed; **LocRotScale** in the case of our cube. If we open this with the white arrow,

we can see each transformation and its corresponding curve on the right-hand side. Each frame has a handle type that you can change with the **V** key. It looks a little bit like the curve handles that we saw in the Haunted House project. We can set them as **Vector** to have a linear interpolation, for instance. We can also change the interpolation type of each frame by pressing the **T** key. This is a very useful menu that we are going to use in order to quickly animate the bounce of our trap. If you press the **N** key, you will see a bunch of options. You can even see modifiers that allow us to add procedural effect to our curves, for instance, noise. Note that you can use the **Ctrl + MMB** shortcut to squash the graph.

Note

Default interpolation setting

In **User preferences**, in the **Editing** tab, we have the ability to choose the default type of interpolation that we want between two keys. Many animators like to set it as constant so they don't have any interpolation. When the animation plays, it looks like stop motion. After they have the right poses, they select all their keys in the graph editor, and they press **V** to set the Bezier interpolation type back in order to polish the animation.

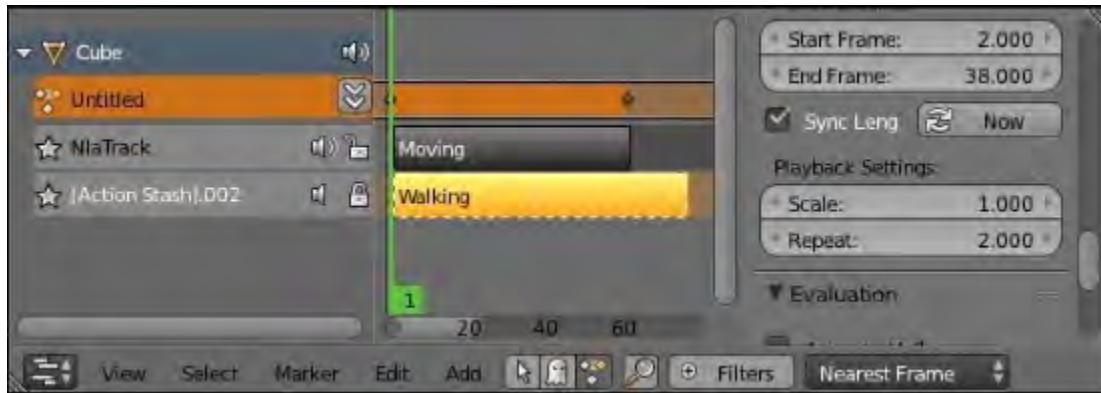


The Graph editor

The Non-Linear Action editor

The **NLA** or **Non-Linear Action** editor is a huge time-saver tool! It allows us to edit actions. Similar to doing video editing, you will edit tracks on which there are actions. You can see an action like a box of keyframes that represent a certain motion of an object. Actions are located in the **Dope Sheet** editor. In its header, we can see a drop-down menu. We can change it from **Dope Sheet** to **Action Editor**. Near this drop-down menu, we have a text field with the name of our action. We can create multiple actions by duplicating them with the **+** button. Each action has its own set of keyframes. So, for instance, you could animate the walk of a character in a specific action, its displacement in another action, and mix them back together in the NLA editor. Back to the NLA editor, we can press **Shift + A** in order to add a new action to a specific track. Tracks are represented on the left-hand side. We can add new tracks by going to the **Add** menu in the header and by choosing the **Add Tracks** option. We can open a hidden panel by pressing **N**. In this panel, you will have many options concerning the selected action.

For instance, we can repeat the selected action by changing the **Repeat** value under the **Action Clip** subpanel. We can also move actions by pressing **G**. This also allows us to move an action from one track to another.



The NLA editor

Preparation of the animation

Before starting the creation of the sequence, it is important to plan what we are going to do.

Writing a short script

We start by organizing our ideas with some brief writing work. We must describe the scene to be animated shot by shot. We can be creative at this moment of the process and imagine any kind of place and situation.

In the first part, we will put some useful information such as the title, exposure (for instance, Out-Day in order to indicate that the action happens outdoors during the day), and the number of the sequence. In our case, there is only one sequence, so we call it **Sequence 1**. This kind of information is usual in a movie script.

The action of the sequence is in the desert. In a very warm and dangerous place, our rat sees a trap after a very long walk. This trap seems to be there for him.

We will follow a rather traditional script structure. There is an initial situation, a disruptive element that happens, and then the fall. In the case of a short animated film of one minute, there is no time to introduce and develop the characters and an enigma. We must go straight to the point.

For our short film, we will do a staging composed of different camera shots, which will require an editing step later. But we must conceive it now. Maybe you know that cinema has a visual grammar that is expressed by editing. It allows us to make sense of the different shots. It is something that you can learn, and there are certain rules to understand. Going deep in this area can only be a huge advantage for your 3D projects.

In order to write a script, we must describe our shots. There are different types of information. These are the field sizes of a shot:

- **The extreme long shot:** This is used for panoramas.
- **The long shot (or establishing shot):** This allows us to introduce a situation.
- **The full shot:** This frames the characters entirely. It is great for large movements.
- **The medium shot:** This frames the chest and the head of the character.
- **The American shot (or ¾ shot):** This frames a character from the thighs to the head. It is close enough to a medium shot.
- **The close-up shot:** This frames the face of the character, and it allows us to perceive emotions better.
- **The Italian shot (or extreme close up):** This frames the eyes of the character.

There are also the angles of a shot:

- **The low angle shot:** The camera is low and the frame upwards. This will enhance the character.
- **The high angle shot:** The camera is high and the frame downward.
- **The aerial shot:** This frames the scenery viewed from the sky.

The camera can also rotate with a pan or be mobile. We often speak of a **tracking shot**. In cinema, the camera is often mounted on a camera dolly or a steady cam for a perfect smooth shot. There are other types of shots and framing, but these are the main types you should know in order to express yourself.

Title : Rat Coybow

Outdoor Day

Sequence 1 :

Shot 01 : Long shot of the background. The character moves forward and the camera makes a lateral tracking shot. The character stops walking in front of the camera. He frowns and seems to observe something away.

Shot 02: Medium shot of the cheese placed on a trap.

Shot 03 : Full shot on the side. We see the position of the rat and the trap.

Shot 04 : Close up of the eyes of the rat.

Shot 05 : Close up of the cheese. slight zoom.

Shot 06 : Close up of the hand preparing to take the gun located above the holster.

Shot 07 : Close up of the cheese with the trap.

Shot 08 : Italian shot on the side. The rat waits a moment and shots with his gun.

Shot 09 : Full shot. The cheese is projected behind the trap that closes. Tracking shot focus on the rolling cheese.

Shot 11 : Close up of the eyes of the rat. He is smiling.

The storyboard

Making a storyboard

After this first reflection of writing the script, we can start making a storyboard. It is a technical document that the areas of animation films have been using since the 30's. A storyboard allows us to describe the action with drawings, but it also goes further than the text in the design of the shots. In the case of teamwork, it is a very useful tool to communicate the work, and it gives a comprehensive view of a project.

Seeing that the storyboard allows us to save a huge amount of time and money, it is a practice that has gradually extended to the field of cinema (classical movies, but mostly special effects movies), theater, clips, and commercials. Even if we are very far from making a blockbuster, and have no team to communicate our work to, a storyboard is a very important step to make a good animated short film.

Don't worry if you are not very gifted in drawing. Many storyboards are very simple and schematic. The most important thing is to clarify your ideas of staging. It must be easy to understand with the indications of stage direction such as camera and character motion.

For our storyboard, we draw the different shots referring to the script that we have done previously. Continuity is from left to right like a comic. To describe a shot, we can make several drawings. For shot 1, three drawings are used to describe the movement of the camera and the character. In order to avoid getting lost, we mark the number of the matching shot at the bottom left of each thumbnail.



Storyboard

Finding the final camera placements and the timing through a layout

The layout is an animated version of the storyboard. It is sometimes called an "Animatic". We don't need to animate our character at this stage. We simply need to visualize the shots that we imagined previously with the script and the storyboard. We can then verify if this works and get a better idea of the time we need for each shot.

The process is as follows:

1. We will import the character in Blender by simply copying it from the `RayCharacter.blend` file to a new file.
2. We will save the scene as `RatLayout.blend`.
3. We will start outlining very simple scenery with a few low poly 3D models. We will use a plane for the floor and an extruded cube for the mountains beyond.
4. We will model a simple cactus and duplicate it pretty much everywhere on the floor.
5. We will also make a simple model of the trap and the cheese. We only need to get the basic shape.
6. To be more comfortable with this, we will organize the environment by dividing it in several parts. It is much better to display the **Dope Sheet**, **Graph** Editor, and camera view in a little window (0 of the numeric keyboard).
7. For each shot, we will create a new scene by clicking on the + button in the main menu bar, and we will select the **Full Copy** option.
8. The placement of our character, the trap with the cheese and making a few tests with the camera are still remaining.

This is the step where we can still make a few changes and test movements and timings.



Screenshot of the layout shot 03 with the rough modeling

Animation references

While animating, it is important to have as many references as you can so that you can have the perfect shot. Many computer animators have a folder with videos of themselves acting the shot. Recording yourself is one of the best ways to understand the gesture of a character. This way you will be able to catch many unconscious movements that you do when you look at the video. It is also a way to improvise different acts. Other reference materials such as character poses or animation cycle images are very interesting. Apart from this, paper and pencil are often useful in order to grasp some poses that you have in mind. You don't need to draw in detail, as a simple stick figure will suffice in order to put the poses ideas on paper.

Organization

Before starting to animate our shots, we will introduce to you how to organize yourself for the whole sequence. The different assets have been created in different .blend files. What's neat about this is that we are going to link them all in one final file for each shot. The benefit of this is that if we want to change the look of one of the assets, we can do it in the original file and it will replicate in the master file. For the rig of the rat cowboy, we are going to create a proxy.

In our case, we have ten shots, so we will create one .blend file for each of them. All these files will be placed in a Scene folder and will reference files that are placed one folder up in the folder hierarchy. Let's create our files:

1. We will first create a new blank file in Blender and save it as 01.blend in a new folder named Scene. Note that you can create a new folder in the file browser with the *I* key.
2. Now let's open the terrain file and select everything that needs to be linked in the shot file. We are only selecting the mesh type objects here. We are going to group our selection with *Ctrl + G* and rename the group as Terrain in the last tool option subpanel.
3. We will then repeat the same process with the other asset files. For the cactus file, you can create one group for each cactus.
4. Now the different groups are ready to be linked in the shot file. In the 01.blend file, select the **Link** option in the **File** menu or press *Ctrl + Alt + O*. We can now click on the **Terrain** file and navigate to the **Group** folder in order to select the **Terrain** group that we've created within this file. We can validate by pressing **Link from Library**.

Note

The Link and Append file structures

The structure of a .blend file is composed of different sections that represent the file. Each section is related to the entity it contains. For instance, in the **Group** section, you will find every group that has been created in the file, and in the **Nodes** section, you will find every node that has been created in the file, and so on. This file format is quite nice because it's open and very well organized.

One very cool feature of Blender is the ability to mix files or parts of files together by linking or appending them. With the **Link** option, you keep a relation with the original file, so any modification will be replicated. The **Append** method creates a pure copy of what you want to mix.



The structure of a blend file

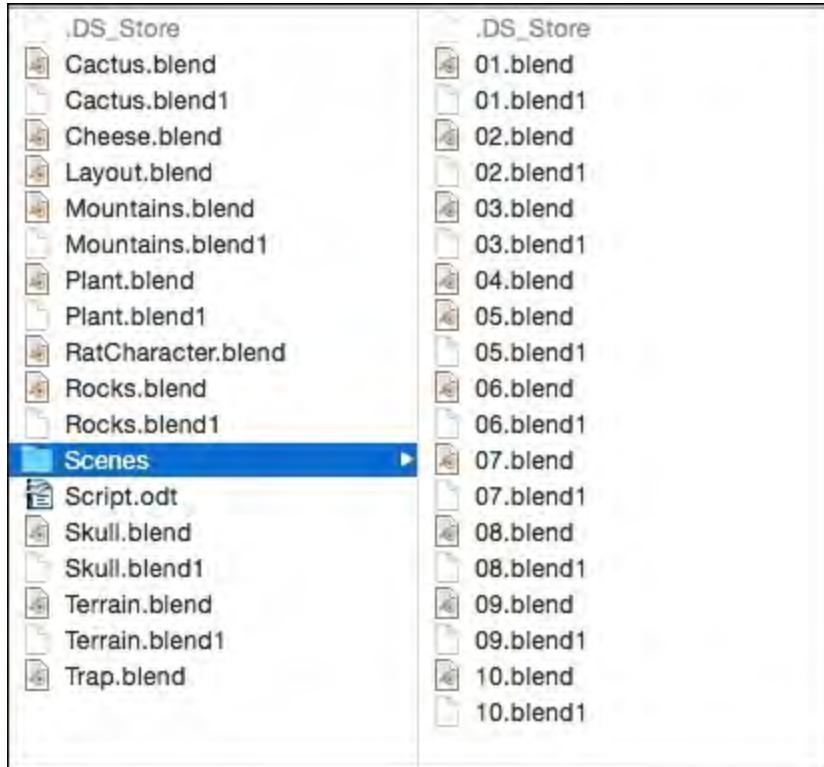
5. The link should be done. You can test whether it works by saving the **01** file then tweaking the **Terrain** file, and going back to the **01** file to see whether the changes appear. Now we can repeat the same process with the other assets that need to be linked in the shot file. You can easily nest files by linking groups to files that are linked themselves as a group in another file.
6. Cactus, bones, and bush are linked in the terrain file and they are part of the **Terrain** group (remember to add them to this group, as it will be linked in each shot file). The terrain is linked in each shot file. The **Cheese** group is linked to the **Trap** file, and the **Trap** group is linked to the **02**, **03**, **05**, **07**, and **08** files. We will not link the trap in the terrain file as it won't be needed for each shot. For the rat character, we will simply create a group with the **Armature** and **Mesh** object that we will link to the **01**, **03**, **04**, **06**, **08**, and **10** files.
7. In each file, we need to create a proxy for the rig of the rat. To do so, we will select the rig, and we will press **Ctrl + Alt + P** and click on the **Armature** object.

Note

Proxy

You may have already seen that when you use the link option, you can't do any modifications in the linked file. This is a security guard, so you only manage your art in one file. But in the case of a rigged character, this could be embarrassing. That's why we create a local access of the rig called a **Proxy** in the linked file with **Ctrl + Alt + P**.

You can have a look at the structure of our project as follows:



The architecture of our project

Animating the scene

Now that we have a story to tell, let's start to animate each shot using the tools that we saw previously.

The walk cycle

We are now going to learn how to create a walk cycle for the first shot. Why a cycle? This is because we are simply going to repeat the walk actions automatically later in order to save time. There are different types of walk that can express the actual feeling of the character. In our case, we are going to animate a cowboy walk, so this means our character will need a certain assurance. In order to be efficient, we are first going to "key" the three main poses of a walk as follows:

1. We will first open the 01.blend file and focus our view on the left-hand side view of the character. We will need to be sure that the **Auto Key** button is turned on, so any translation (grab, rotate, or scale) will create or override a key where the bar is located in the timeline. This button is located in the header of the **Timeline** editor and looks like a recording button.
2. Before creating our poses, we will have to select our armature in pose mode. We can then hide the **Master** bone by selecting it and pressing *H* because our walk will be animated onsite. Also, be sure that your timeline bar is at frame 0.
3. The first pose that we are going to create is usually called the "Contact pose" because both feet are touching the ground. From the side view, we will select the left foot bone, and we will move it in front of the character. We will then place the other foot bone in the opposite location. As you can see, because the legs are too far apart, the character isn't able to touch the ground (represented here by the Y world axis). So we have to select all the visible bones and move them down until the character is on the ground.
4. Now we will lift the front leg toes up a little bit on the local *x* axis (press *R* and then *X* twice).
5. We will then pose the back foot so that the rat stands on its toes.
6. From a front view, we will orient both legs outwards a little bit.
7. The arms need to follow the direction of their opposite leg. To move the arm, we will use the **HandIK** bone. We can also slightly bend the arms and rotate it, and then break the hand rotation a little bit.
8. From a front view, we can slightly put the right arm inward. The left arm will not rotate as much inward because of the holster.
9. In order to polish this pose, we will rotate the **Hips** bone down in the direction of the right leg and the **Spine01** and the **Neck** bone in the opposite direction of the hips. We will also rotate the head down so that the character looks at the ground. We will also pose the fingers so that the index finger is straighter than the others. We will also rotate the tail outward.
10. At this point it's a good idea to open a **Dope Sheet** editor. As you can see, we have small diamonds that represent the keys on each bone (described in the list on the left-hand side). For the main poses, it's always a good idea to have a key placed on every bone. To do so, we will select them all in the 3D view, and we will press *I* and choose the **LocRotScale** option.
11. In order to have a perfect cycle motion, the first and the last key of the walk needs to be the same. So in the dope sheet, we will select all the keys of our first pose by right-clicking on the **Dope Sheet Summary** corresponding to the key, and we will press *Shift + D* to duplicate this. We can move this to frame 24. You can clearly see that the keys are the same because of the dark lines that link them.

- Now, we will copy this pose right between these two keys but in a mirror. To do so, we will select every bone in the 3D viewport, and we will store the pose with *Ctrl + C*. At frame 12, we can press *Ctrl + Shift + V* in order to copy the pose in the mirror, thanks to our bone naming convention with the **.L** and **.R** suffixes. If we scrub the timeline bar, we can see a preview of our walk. But there are still many things missing.

Congratulation, you have made the contact poses of the walk cycle! Now we will create the "Passing pose". This is done as follows:

- To create our passing pose, we will use the same process as before. We will first create our pose between the 1 key and the 12 key, and we will mirror it between the 12 and 24 keys.
- So we will place our timeline bar at frame 6. We will start creating our pose from the side view. The leg that was in the front of the character will be straight. So we will select the left foot controller, and we will align it horizontally with the rest of the body. As you can see, the leg can't be straight if the hips aren't moved up. We move the hips up. This is the key point of any walk: *the body always moves up and down*.
- The other foot is bent. It is also placed slightly behind the straight leg, and the toes point downward.
- At this point, the arms are almost straight and aligned with the rest of the body.
- From the front view, **Hips**, **Spine01**, and **Neck** need to be aligned with the ground.
- We can slightly rotate the head on the *z* axis to the left of the rat.
- Now we ensure that we have keys on every bone. After this, we will copy our pose in the mirror on frame 18 with the same method seen previously.
- A nice trick you can do is to change the visual look of your keys in the Dope Sheet by selecting them and by pressing *R*.

We now have the essential poses of a walk. The rest of our work will consist of exaggerating the motion by adding a "Down" and "Up" pose. This is done as follows:

- The "Down" pose will be placed before the "Passing" pose. To create this, we will clear the toe rotation of the front foot.

Then we will exaggerate the pose by slightly moving the pelvis down. The whole goal of this pose is to feel the weight of body on the ground.

- The "Up" pose is placed after the "Passing" pose. This consists of lifting the character on his straight leg and toes.
- After we have finished these poses, we can duplicate them in the mirror by following the same order according to the "Passing" pose.



Walk cycle poses

The walk cycles is completed now! We can view it by scrubbing on the timeline or simply by changing the end frame to the 23 frame in the timeline header and by playing the animation with *Alt + A*.



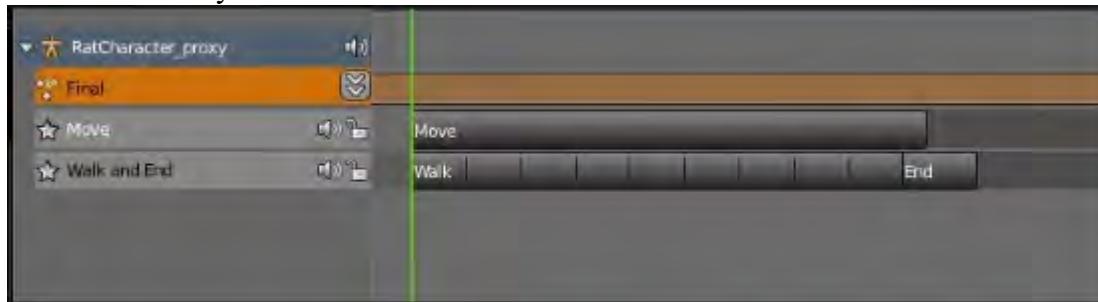
The Dope Sheet for our walk cycle

Mixing actions

We are now going to create three new actions that we will blend in with the NLA editor. One will be in charge of moving the character from its current location to the front of the camera. One will represent the character lifting his head up, and the last one will contain a mix of the others. This is done as follows:

1. The first thing to do is to copy the camera from the Layout blend file. To do so, we will simply copy the camera of the first scene in the corresponding file with *Ctrl + C*, and we will paste it

- with *Ctrl + V* in the **01.blend** file. If you have any other cameras in the scene, remove them and be sure that the copied one is the active one by selecting it and pressing *Ctrl + 0* numpad key.
2. In order to create our new actions, we will need to open the **Action Editor** from the Dope Sheet editor. As you can see, our walk cycle has been already placed on the default action. By the way, we can rename it **Walk**.
 3. We can now click on the **+** button in order to create a new action. This new action will be a copy of the walk action, so we can delete all the present keys and rename the action **Move**.
 4. Now we can start to animate the character displacement on this action. To do this, we will simply use the **Master** bone of our rig. In our case, we have added two keys at frame 0 and 225.
 5. The problem now is that the rat seems to accelerate at the beginning and decelerate at the end. To solve this, we are going to use the graph editor and change the shape of the Y location curve. So in the graph editor, we select the keys for the Y location, and we press *V* and **Vector**. Now, as you can see, the curve is linear. Remember to save your file!
 6. Now we are going to mix the walk and the displacement together. To do this, we will create a new action that contains all the others. We rename it as **Final** and remove all the present keys.
 7. In the NLA editor, we will ensure that we are editing on this action. We will then create a new track (**Add | Add tracks**) and press *Shift + A* to add the **Move** action to it.
 8. We can then press *Shift + A* in order to add the **Walk** action under the **Move** track. Now we just need to repeat our walk cycle with the **Repeat** option located in the right menu (*N*) under **Animation Clip** (be sure that the walk action is selected in your track). The important thing here is that the cycle stops just before the **Move** action because we are going to animate the head at the end of the walk and the head in a new action.
 9. We now need to copy the pose of the character when the walk cycle stops, so we can start with this pose in a new action. We will use *Ctrl + C* with all the bones selected.
 10. Now we are going to create the next action. To do this, we will click on the **+** button again, delete the keys, and rename the action **End**.
 11. We can now paste our pose (*Ctrl + V*) on the same frame where we copied it in the NLA in the **End** action. This is because of the camera motion.
 12. Now we need to complete a half walk and animate the character's head. In our case the animation starts at frame 216 and ends at frame 248. We ensure the character stands on his feet correctly, and we can animate the head pointing towards the camera. We can also add an eye blink in the middle. We will also rotate the head to the left of the camera.
 13. After finishing the animation of the action, we will need to reposition our keys so that they start at frame 0. In the **Action Editor**, we will press *A* in order to select all the keys, and with *G*, we will drag them until the first frame of our animation is on frame 0.
 14. Now we can go back to the **Final** action and open the NLA editor. We will then place our **End** when the walk cycle ends.



*The NLA with our three actions mixed together in the **Final** action.*

15. That's all! We have now blended our three actions with the NLA. Note that you can also rename your tracks on the left-hand side of the NLA by clicking on their default name.



*One frame of the **End** action*

Animation of a close shot

The close up of the face that we are going to do is directly inspired by Italian westerns of the 60's. A close up allows us to give a tension focusing on the eyes of our character. This is done as follows:

1. We will start by opening the **04.blend** scene.
2. We will need to place our character at the same place as in the **01.blend** scene, so we will also open the **01.blend** scene.
3. In the **Right Panel (N)** of scene **01**, we must copy the location information of the **Master** bone (Root) on the three axes, X, Y, and Z. In our case, **X: -1.19863**, **Y: 0,0**, and **Z: 12.29828**, and we will paste them on the location parameters of the **Master** bone in scene **04**. Our character is now in the right place.
4. We need to be sure that the **Auto Key** button is turned on, and we add the first key at frame 1 on the camera and all the bones of the character.

This animation will be in 50 frames.

1. We also need to place the camera in front of the face of the character.
2. We will move to frame 50 of the timeline, and we will slightly move the camera on the local **z** axis.

3. We can see a slow in and slow out effect of the camera, the keys are in Bezier interpolation mode. In order to change the interpolation mode, we must open the **Graph Editor**. We will select the keys of the camera and press V and select **Vector**. Now the camera is moving in a linear way.

Let's animate our character. He is watching the trap, so he doesn't move a lot. The animation is done as follows:

1. We begin by animating the head. We will move the cursor of the **Timeline** at frame 50, and we make a little rotation on the local z axis, a new key frame is added.
2. The Frown bone stays down to keep a serious look.
3. We can now animate the eyes. At frame 50, we will move down the **EyeTargetMaster** a little bit. He is still looking straight toward the camera.
4. We will now add a blink of the eyes to give more realism.
5. We will select the both **EyeTarget.L** and **EyeTarget.R**, and we will add a key at frames 16 and 23 (press I and select **Scale**). We will move then at frame 19, and we will scale the bone controllers to close his eyes.



The close shot

This very short animation for shot 4 is finished. Shot 10 is almost the same but with a smile at the end. Let's start the animation of the gunshot!

Animation of the gunshot

The animation of shot 8 this time is a bit more complex. This will be done as follows:

1. We will start by opening the **08.blend** scene.
2. We will place our character at the same location as the **01.blend** file.
3. With the **Auto Key** button turned on, we will start placing the camera on the left-hand side of the character.

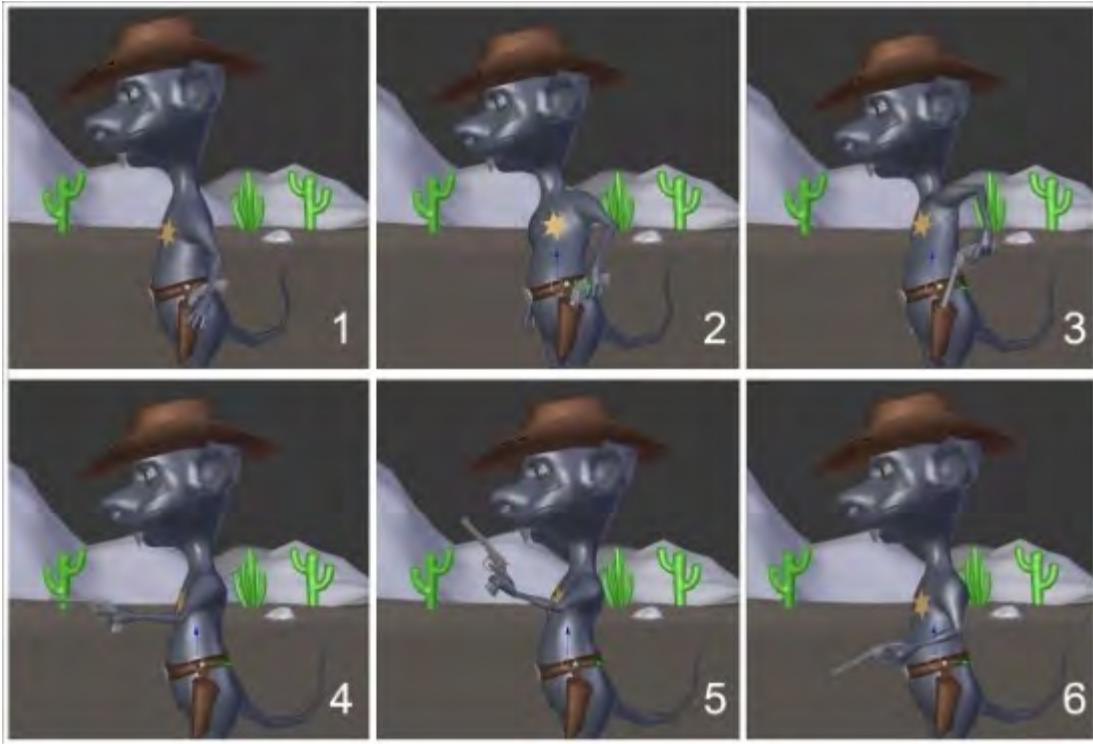
It is an animation in 30 frames.

1. We will put the camera on the left-hand side of the character, and then we will move it slightly to the left along the y axis to frame 30. We will change the keys of the camera in the **Vector** mode (V) in the **Graph Editor**.
2. We can now animate our character. We will start animating the hand. The hand must go straight to the butt of the gun to hold it. The position and the inclination of the hand are very important. It doesn't matter if there is a little interpenetration of the thumb. The animation is fast, and the point of view of the camera can hide small mistakes. The forefinger must be close to the trigger when he holds the gun.



The gun shot

3. To move the fingers, we must select the top bones and rotate them on the local *x* axis (press *R* and then press *X* twice). The **Copy Rotation** constraints will make the rest.
4. When the gun is caught, we must change the influence of the **Child Of** constraint of the gun bone. The influence of child of controlled by the holster is now at **0.000**, and the influence of child of controlled by the left hand is now at **1.000**. The gun is now following the hand. We can keep animating the left hand.
5. We will position the hand so that the gun gives the impression that it will shoot ahead. There is the recoil of the gun, so the hand makes a sudden rotational movement upwards.



Animation of the gun shot

When the animation of the left hand and the gun is done, the largest part of the animation of the sequence will be done too, but we still need to polish this shot a little bit. The rest of the body must be animated and can't stay inert. This will be done as follows:

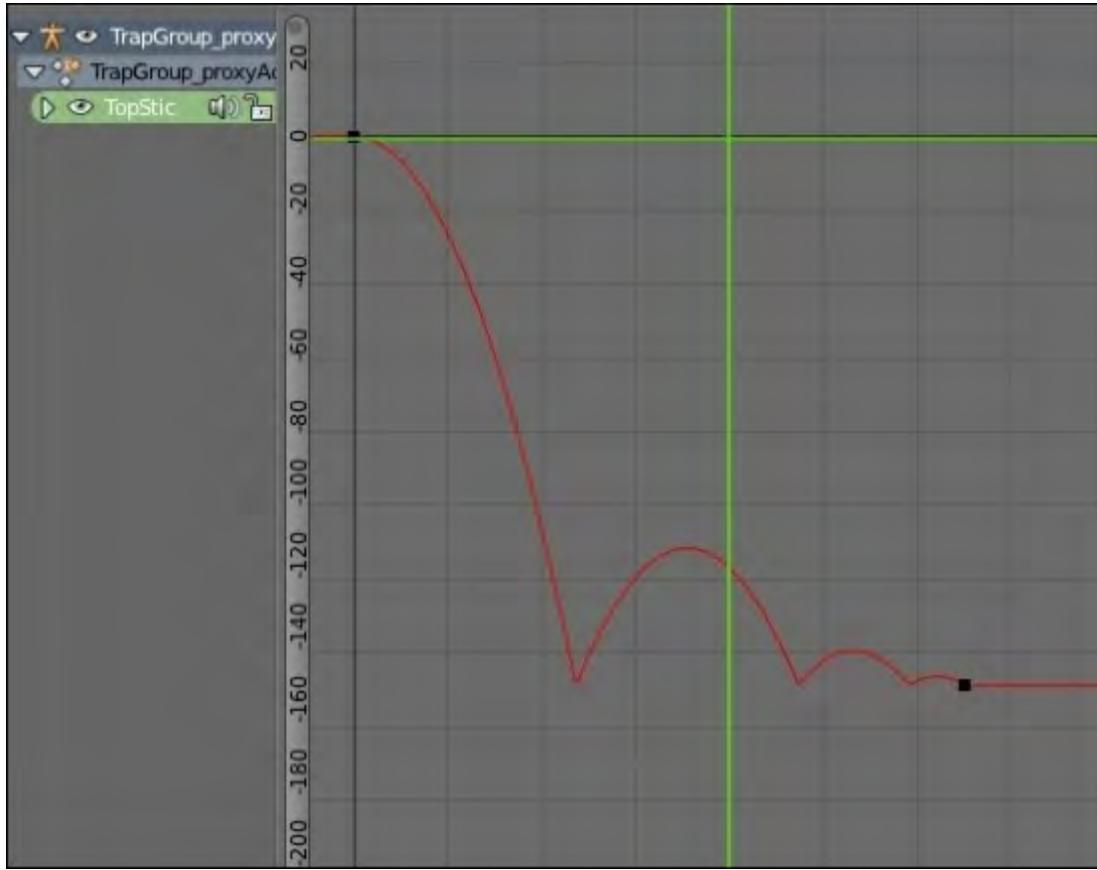
1. We need to make a rotation of the **Spine01** bone on the *z* and *y* local axis when the left hand catches the butt of the gun. Likewise, the left shoulder must rotate towards the top (refer to **2** and **3** in the preceding screenshot) at this moment.
2. As the rotation of the spine01 bone the head is inclined to the right, we will rotate it a little bit to adjust it. For the recoil of the gun, we will make a little rotation of the head.
3. The **HipsReverse** bone also makes a rotation on the local *z* and *y* axes to gives a more realistic feeling.
4. The right hand and the tail also make a little arc. In this case, they are quite important details.

So, the animation of the gunshot is now complete. Let's talk about the animation of the trap.

Animation of the trap

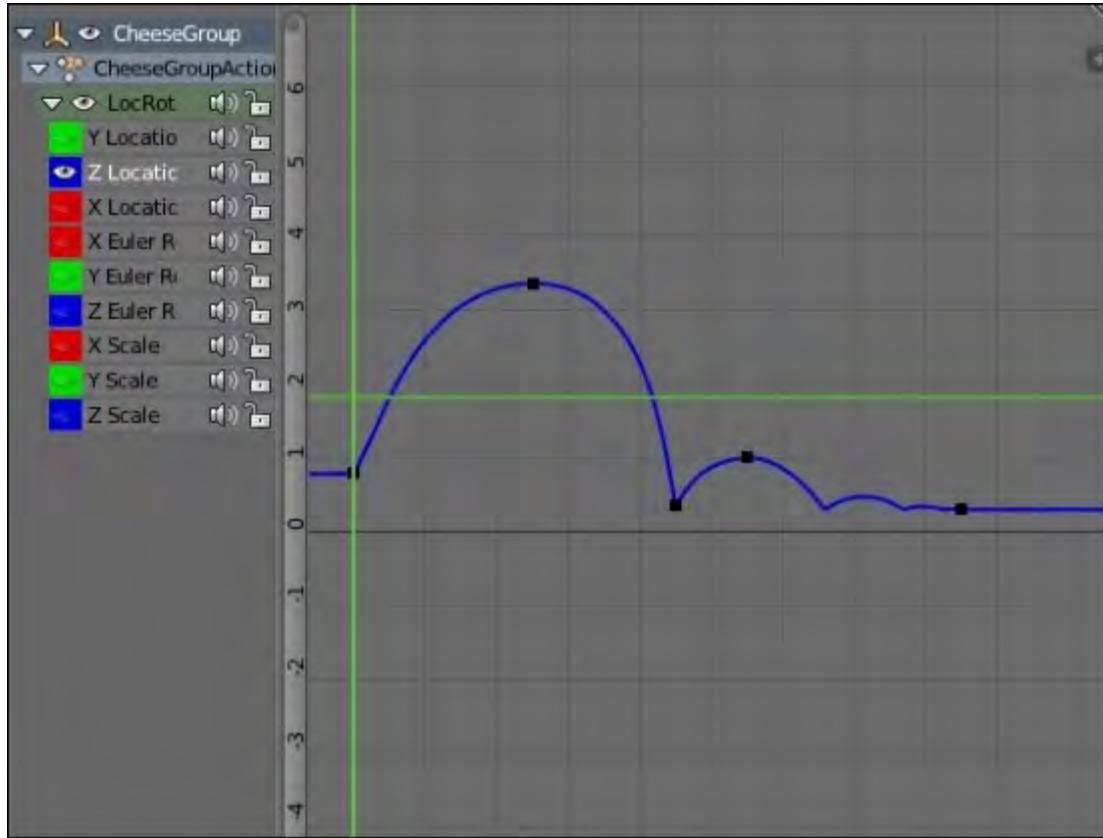
It's now time to animate one of the most technical shots of our sequence where the cheese gets shot on the trap. This will be done as follows:

1. We will start by opening the **09.blend** file. Then we can frame the trap with camera as done in the layout.
2. The first thing to do is to add a key frame on each bone of the trap only for rotation, so we select them and press *I* and select **Rotation**. We won't use AutoKey for now. Remember that the trap is linked. If you can't access the rig, it's simply because you don't have a proxy in order to manipulate it.
3. Now we can go to frame 13 and rotate the **TopStick** and the **TrapPlank** bone on their local axes as far as they can logically go.
4. The **Spring** bone animation will be shorter, though. We will key its extreme rotation on frame 7.
5. We can now test our animation to see what's wrong with it, but first we will hide the cheese with the *H* key. As you can see, the animation doesn't seem very natural. This is why we are going to tweak the curves in the graph editor.
6. We will open the graph editor and click on the **Show only selected** button in the header.
7. Now we can select the **TopStick** bone and unfold its **Rotation** values on the left-hand side of the **Graph** editor. We can even hide the other *y* and *z* rotations as we don't need them. To do so, we will use the Eye icon. As you can see, the curve has an ease in and ease out effect. We don't want this. We want a bounce effect. To do so, we can, of course, add more keys and waste time to animate it by hand, or we can use the power of the Dynamic effects of the interpolation menu. So we select our keys, and we press *T* and select **Bounce**. As you can see, the curve changes.
8. We can do the same with the **TrapPlank** and the **Spring** bones. For this, a later bounce will be more subtle. Now if we play our animation, everything looks more natural.
9. Now let's animate the cheese. First unhide it with *Alt + H*.
10. We can now reactivate the **AutoKey** option, it will be easier. Let's start by adding a key on frame 0 with press *I* and select **LocRotScale**. Now, because of autokey, we can move to frame 16 and move the cheese to its final destination.
11. We can now go to frame 5 and put the cheese in the air. This will change its *Z* location. We can also rotate it and scale it locally on its *z* axis (press *S* and then press *Z* twice).
12. The animation at this point is not very convincing. In order to add more realism to it, we can open the graph editor and change the curve of the *Z* location of the cheese, so it is much more like a "dome" at the beginning. To do so, we can set frames 0 and 9 to a vector type (press *V* and select **Vector**). We will need to change the handles of frame 5 too.
13. Next, we will change the end of the motion by adding two new keyframes on the *Z* location at frames 11 and 17. These keys will have a **Bounce** Dynamic effect. We will use the handles in order to smooth the curve as much as possible where it is needed. Remember that the animation process involves a lot of trial and error. You also need to constantly play your animation in order to know what you'll have to correct.
14. We can now reset the scale of the cheese at frame 11 by pressing *Alt + S*.
15. What we've done so far is the base of the cheese animation. Now we can polish the animation by changing its rotation here and there and by polishing the *Y* location curve.
16. We can now animate the camera. Between keys 0 and 16, the camera doesn't move, so we simply duplicate frame 0 to frame 16.



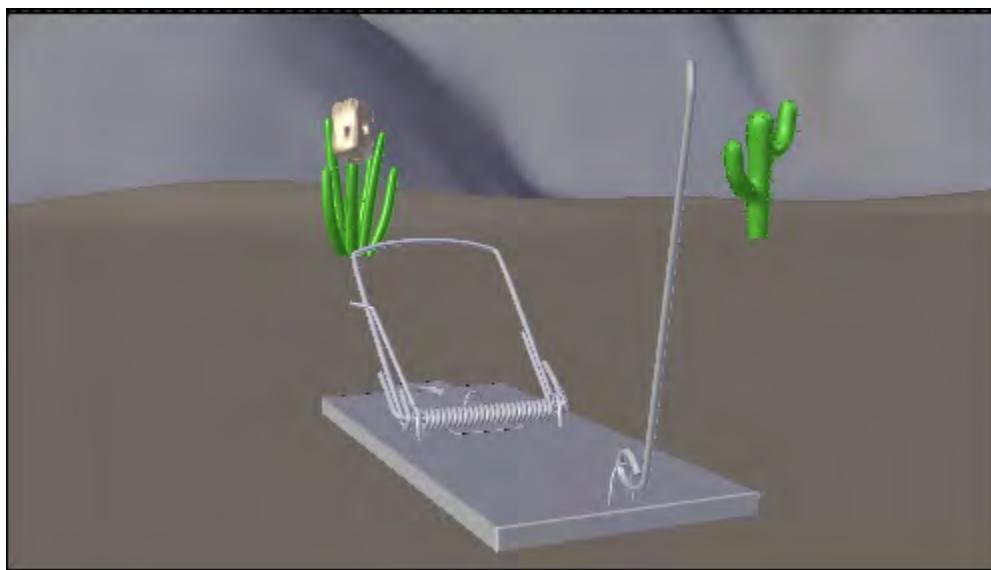
*The **TopStick** X rotation curve*

17. We can now place our timeline bar at frame 37 and focus on the cheese closely, so we can see the hole due to the gunshot. The key has been placed due to the AutoKey option.



The cheese Z location curve

The animation of the 09 shot is now completed, congratulations!

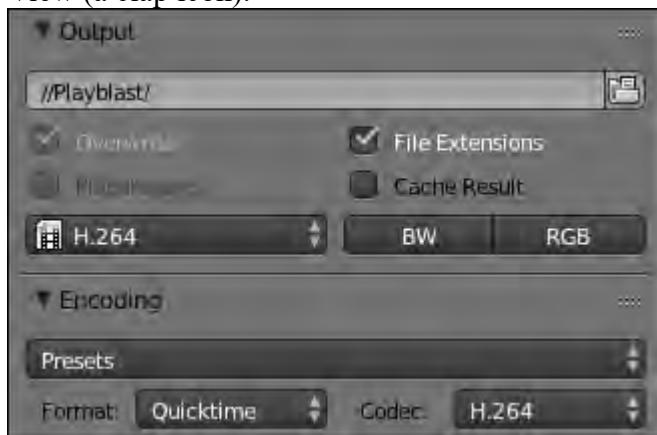


Frame 4 of the 09 shot.

Render a quick preview of a shot

The last thing we are going to do is render quick previews of each shot. This is often called a **Playblast**. This simply renders the animation of the viewport. In this section, we are only going to render the playblast of the first shot. So let's start:

1. We will open the **01.blend** file.
2. In the Properties editor, in the **Render** tab, we will go to the **Output** subpanel. Here we can choose the path where the render will be. To be well organized, let's create a **Playblast** folder in our **Scene** folder and set it as the output path.
3. Now we will need to choose the file format. By default, it will output a sequence of a PNG file. We change it to **H.264** in order to have a movie type file.
4. In the **Encoding** section, we can choose the extension of our movie, and we choose **Quicktime** to have a .mov file. Later, you'll see how to render the shots frame by frame in a non-video format, but for now in order to quickly see the result, we will use .mov.
5. Be sure that you are viewing from the camera by pressing the *0* numpad key. In order to render the playblast, we can press the **OpenGL Render active viewport** button in the header of the 3D View (a clap icon).



*The **Output** option for the **Playblast** rendering*

Summary

This was a very tough chapter, but we hope you have found it interesting. We covered a lot of things here, such as the principles of animation, the majority of the Blender animation tools, and how to prepare ourselves with a script, storyboard, and layout. We also learned how to animate a walk cycle and mix it with other actions using the NLA. We applied many of the 12 principles of animation such as pose-to-pose, squash, and stretch overlapping. Now, you should have all the knowledge to start telling your own stories with Blender. In the next chapter, we are going to finalize our sequence by rendering it with cycles and editing it with the VSE.

Chapter 9. Rat Cowboy – Rendering, Compositing, and Editing

Welcome to the last chapter of the book. In this chapter, you will learn advance material creation in Cycles, such as how to use a skin shader or how to create a realistic fur. Next, you will learn about passes and how to do a raw render with the different passes. Then, you will receive an introduction to the nodal compositing so that you can enhance your shots. Lastly, we will talk about the Video Sequence Editor in order to edit the final sequence. Let's start!

In this chapter, we will cover the following topics:

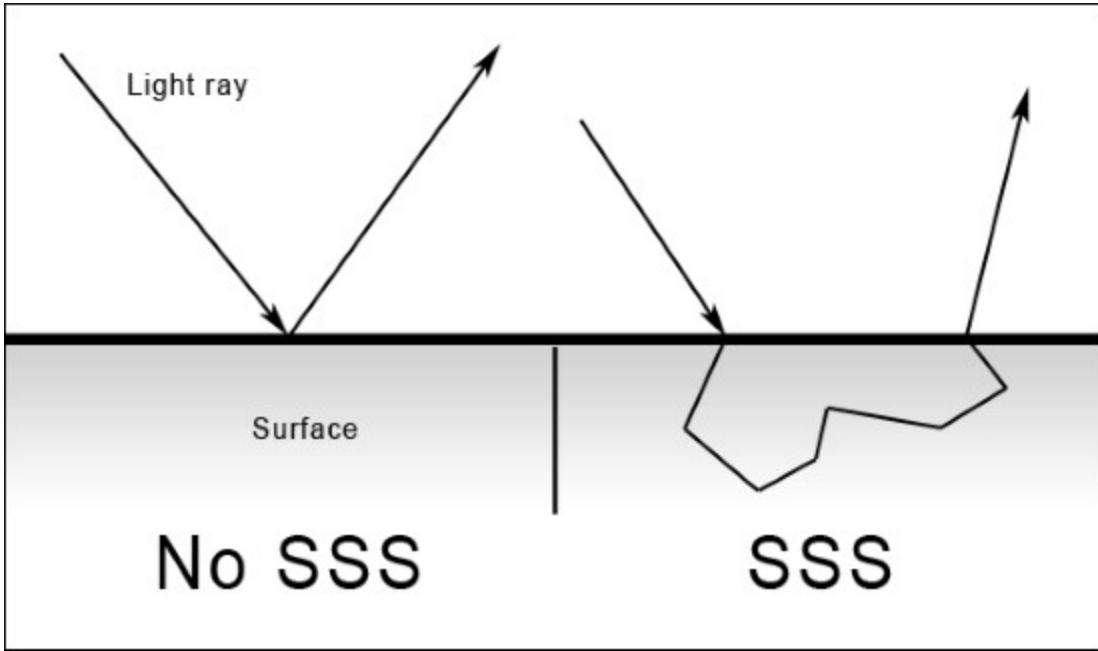
- Creating advance material
- Creating fur particle systems
- Setting up Cycles for an animated scene
- Using passes
- Introducing nodal compositing
- Editing a sequence

Creating advanced materials in Cycles

We already covered material creation with Cycles in the Haunted House project, but now we are going to go further by creating a skin material using subsurface scattering, a complete fur, and an eye material. Let's start!

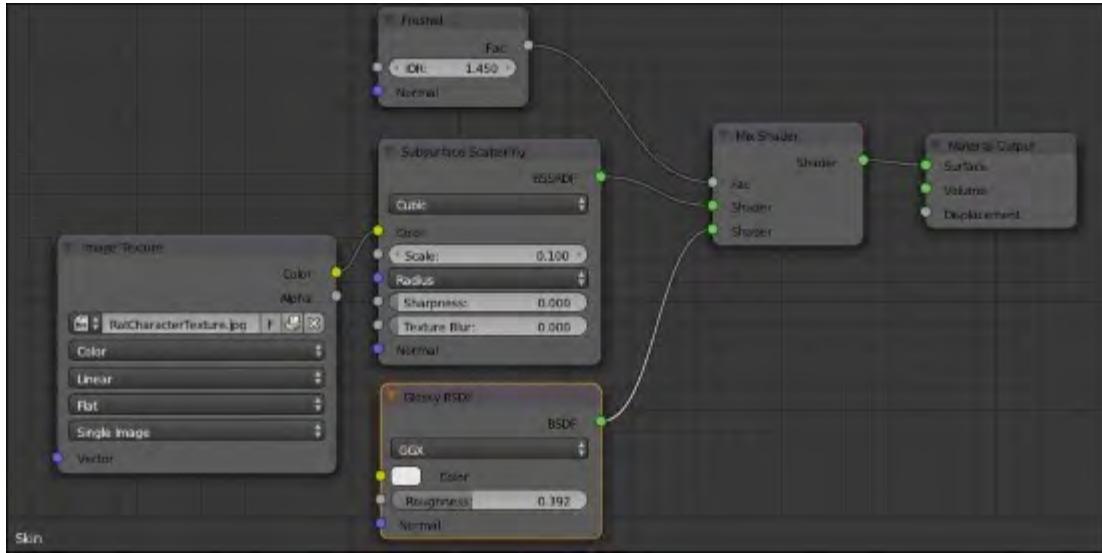
Skin material with Subsurface Scattering

The skin has a very translucent aspect. We can truly see this effect when we pass our hand in front of a lamp or in the thin part of the ear (the helix). So, when creating a skin material, we get this phenomena with a Subsurface Scattering node (usually abbreviated SSS). It is called this because the light rays are scattered through the geometry when intersecting the mesh. This is not the case with a diffuse shader, for instance, as the light rays are simply blocked. SSS often gives a reddish tint to the thin parts where light rays scatter a lot. So let's create the skin material of the rat.



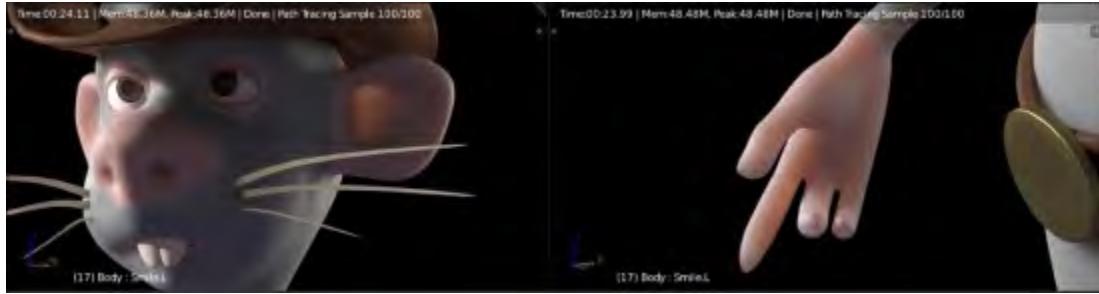
The way light rays react on SSS surfaces

1. We will open the RatCharacter.blend file and split our interface so that we have a second 3D view for the real-time renderer and a node editor. Note that the real-time renderer for the SSS shader will only work in the CPU mode or with the GPU in the experimental mode.
2. We will add a new slot in the material tab of the Properties editor and a new material that we name as **Skin**. We will press the **Use node** button in order to work in the node editor.
3. In the node editor, we will remove the default **Diffuse** shader by selecting it and pressing **X**. Then, we can add a **Subsurface Scattering** shader by pressing **Shift + A**.
4. We have some options to tweak for this shader. The first one is the **Scale option**, which corresponds to the amount of SSS that we want. In the case of the rat, we set it to **0.1**, but in order to have the correct value, you will need to perform a test. It's just a matter of placing a light in the back of the character and looking at the thin parts, such as the ears.
5. The next very important setting to tweak is the radius that corresponds to the predominant color that will result to the SSS effect. In many cases, we will let more red than green and blue because of the color of the blood that's under the skin. This is a set of three values that corresponds to R, G, and B, in our case, we set them to **1.0**, **0.7**, and **0.5** respectively.
6. Now, we will plug our texture in the **Color** input. Finally, we can connect the shader to the output.
7. We will now add a reflection to our skin by mixing our SSS shader with a glossy shader. To do so, we append a **Mix Shader** on top of the SSS to the Material output wire.
8. In the second shader input, we can bring a **Glossy BSDF** shader and change the **Roughness** value to **0.392** so that the reflection is less sharp.
9. For the **Fac** input of the **Mix Shader**, we will add a **Fresnel** node. This skin shader will be sufficient for our needs, but note that we can go much deeper in the subject by creating a shader with multiple maps that corresponds to the different skin parts, such as sub-dermal and epidermal.



The skin material nodes

The SSS effect on the ears to the right and on the hand to the left in viewport rendered mode will look like the following:



Eye material

We are now going to create a less difficult material, that is, the one of the eye corneas. This will be like a glass material, but we will optimize it a little bit because the default glass shader is so physically accurate that it also casts shadows of the glass itself. These take a long time to render and are rarely visible. In order to apply our material, we will model a simple cornea in the eye mesh itself. The front part of the cornea is extruded a little bit. This allows us to catch the reflection rays better:

1. We will first need to add another material slot and a material that we rename as **Cornea**. As you can see, we already have three slots that correspond to the white part and the pupil of the eyes. Feel free to replace it with one material with a texture.

2. Next, we can select the cornea piece of the mesh in the **Edit Mode** with the **L** key and press the **Apply** button in order to apply the cornea material on this part of the mesh.
3. In the node editor, we will replace the default Diffuse shader with a **Glass BSDF** shader.
4. We will now mix the **Glass BSDF** shader with a **Transparent BSDF** shader. A transparent shader simply lets every ray to pass through.
5. Now, in the **Fac** input of the **Mix Shader**, we will plug a **Light Path** node (press **Shift + A** and select **Input**) with the **Is Shadow Ray** output. This will tell the render engine to use the transparent shader for the incoming shadow rays and the glass shader for the others. At this point, we should have a nice reflecting eye.

The fur of the rat

Now, let's dive into a complex section about fur creation. In order to create a convincing fur, we will have to create a complex material, have a perfect hair particle combing, and correct lighting. If one of these three parameters is sloppy, it won't look great. Let's start with the particle systems:

1. In order to add more realism, we are going to create three particle systems. Let's first select the character in the **Edit Mode** and add the main particle system in the Particle tab of the Properties editor. We will name both the system and its settings **Basic_FUR**.
2. We will change the system's type from **Emitter** to **Hair**. In the **Emission** subpanel, we will change the **Number** to **500**, which correspond to the guiding hairs. We can also change the hair length to **0.140**.
3. In the **Children** section, we will choose the **Interpolated** method. The number of children that will follow the guiding hairs is too low. We will change the **Render** option to **600**, so each guide will have 600 children. We can also change the display option to **100** to preview the result in the viewport.
4. In the **Children** subpanel, we can change the **Length** setting to **0.640** and the **Threshold** to **0.240**. This will add some randomness to the length of the children.
5. Then, in the **Roughness** section, we can change the **Endpoint** value to **0.046** and the **Random** to **0.015**. **Endpoint** will spread the tips of each hair strand.
6. Now, we will create a Vertex group that will determine where the fur will be located on the rat. In the **Object data** tab of the Properties editor, in the **Vertex Groups** section, we will start by locking all our skinning groups by selecting the black arrow button and choosing the **Lock All** option. This will ensure that we don't change them inadvertently. We can now add a new group with the **+** button and name it **Fur**. In the **Edit Mode**, we can select the hands, nose, ears, tail, mouth, and eye contour. We invert the selection with **Ctrl + I** and press the **Assign** button with a weight of **1.0**.
7. Back in **Particle** tab, in the **Vertex Groups** section, we can set the **Density** field to our new vertex group. We should now only have hairs on the needed parts.
8. In order to improve our system, we will create a new vertex group named **Fur_Length** that will affect the length of the hair on certain parts. To create the group, we can duplicate the previous **Fur** group with the corresponding option in the black arrow drop-down menu. We can then use the weight paint tools in order to subtract weight from different parts. In our case, the head is green and the arms and the legs are orange and blue under the belt.
9. In the **Vertex Groups** section of the **Particle Settings** tab, we can change the **Length** field to this new group.

Now, we need to comb our particles as follows:

1. To do so, we can use the **Particle Mode** (located in the same drop-down menu of the **Object Mode** or **Edit Mode**). By pressing **T**, we open the **Brush** panel where we can use the **Comb** brush to comb the character's hair.
2. We will use the same shortcuts as the sculpt mode for the brush settings. The **Add** brush is nice in order to add new strands when you find some gaps. When using this brush, it's best to check the **Interpolate** option so that it smoothly blends with the others.
3. In the header of the 3D view, you have three new buttons (to the right of the layers) that allow you to select the particle in different ways. With the **Path** mode (the first one), you can only control them with brushes, while with the **Point** mode (the middle one), you have access to each point of the hair, and with the **Tip** mode (the last one), you only control the tip. With the last two modes, you can grab and rotate your character's hair with **G** and **R**. In the left panel, we can also activate the **Children** option in order to see the children.
4. Now let's add another particle system and name it **Random_FUR**. We can then copy the settings of the first one by choosing it in the **Settings** field and pressing the **2** button to make a unique copy of it. Now, we can safely change the setting without affecting the other system. We can click on the **Advanced** option.
5. We will start by changing the amount of guiding hairs to **50** and their length to **0.1**.
6. In the **Emit from** section, we will choose **Verts** and check the **Random** option.
7. In the **Physics** subpanel, we can change the **Brownian** value to **0.090** to add a little bit of randomness.
8. In the children section, we will change the **Render** and **Display** sliders to **50**.
9. We will then change the **Length** to **0.288** and the **Threshold** to **0.28**.
10. We will then ensure that the **Density** field still contains the **Fur** group, but we will remove the **Length** field.



The children settings of the Basic_Fur system.

11. Just as we did for the first two systems, we will add a new system for the fur of the ears. This is very subtle. It has a short hair length and a vertex group for the **Density** field. It also has only 20 children.



The Particle Edit Mode

You are now done with the particle systems. It's now time to create the materials and change the strand thickness and shape. This will be done as follows:

1. Let's add a new slot in the material tab and a new material named **Fur**.
2. In the node editor, we will delete the default **Diffuse** shader and add two **Hair BSDF** shaders. The first one will be of the **Reflection** type with **RoughnessU** and **RoughnessV** set to **0.500**, and the second will be of the **Transmission** type with **RoughnessU** to **0.1** and **RoughnessV** to **0.2**. We will mix them with **Mix Shader**. The **Fac** input will be plugged with the **intercept** output of a **Hair info** node.
3. We will then add **Musgrave Texture** node with a scale of **538**. We will add a **HueSaturationValue** node with the **Fac** output of the texture connected to the **color** input. We will then mix the result with a **MixRGB** node and **ColorRamp**. The **Fac** input of the color ramp is fed with the **intercept** output of a Hair info node. This will add a gradient map on each strand.
4. We will then plug the output of the **MixRGB** node into another **MixRGB** node. The second input of this node will be fed with a **HueSaturationValue** node. The **Hue** input will receive the

color output of a **Musgrave** texture node. This will add color variety to the strand. But in order to lessen the coloring effect, we will change the **Fac** of the **MixRGB** node to **0.047**.

5. Finally, we can plug the result of the last **MixRGB** node in the **color** input of the first **Hair BSDF** node. Our hair material is now completed!
6. We now have to change the **Render** settings of each particle system. In the **Render** subpanel of the particle settings of each system, we will choose our fur material and activate the **B-Spline** option with a value of **4**. This will smoothen the render of the hairs.
7. Then, for the **Basic_Fur** system in the **Cycles Hair Settings** subpanel, we will change the root to **0.5** with a scaling of **0.01** and a shape of **-0.09**. We will use the same settings for the **Random_Fur**, except for the root that we set to **0.15**. Again, we will use the same settings for the **Ear_Fur** system, except for the root that we set to **0.05**.
8. We can now add temporary lights in our **RatCharacter.blend** file in order to do test renders. We will set our samples to 300 and render a preview of our rat.



The fur material in the Node editor

The image with a preview render with low render settings is as follows:



Now you have the knowledge to create even more complex materials with Cycles! We are now going to show you how to render the first shot of the sequence, and what's nice with the link is that we will see our fur and materials in each shot file.

The Raw rendering phase

Previously, we have seen how to render an image in Cycles. It is quite different for an animation. It's best to first do a raw render of the shots with the following settings:

1. We will start by adjusting the device. If you have a good graphics card, remember to check the **GPU** device option.
2. Let's now adjust the samples in the **Render** panel (**Properties** | **Render** | **Sampling**). The skin needs enough samples to reduce a noisy effect. 100 or 150 samples are enough to have an idea, but consider setting a higher value for the final render.
3. Still in the **Sampling** tab, we will put **1.00** to **Clamp Direct** and **1.00** to **Clamp Indirect**. It allows us to reduce the noise, but you may lose a little bit of the bright colors.
4. We should remember to check the **Cache BVH** and the **Static BVH** options in the **Performance** tab. It allows us to optimize the render time.
5. You can make a test by just rendering a frame (*F12*). Pay attention to the time it takes to complete the rendering process for only one frame. Thus, you can calculate the time needed for a shot.
6. In the **Passes** tab (**Properties** | **Render Layers** | **Passes**), we will verify whether **Combined** and **Z** passes are checked.

Note

Passes in Cycles

Passes are a decomposition of the 3D image rendered in Cycles. We can also render passes with Blender Internal but in a different way. Once the rendering calculation is over, we can combine these passes and combine each pass together to create the final image with directly compositing in Blender or another software. The passes allow us to get a control to considerably improve an image and make changes even after the image is rendered. So, it gives more fine-tuning opportunities and saves us a lot of render time.

If you want to explore all the passes of Cycles, visit this link:

<http://wiki.blender.org/index.php/Doc:UK/2.6/Manual/Render/Cycles/Passes>

7. Now that the image quality parameters are set, we must choose an output format of our animation. In the **Output** tab, we will choose the **OpenEXR MultiLayer** format.

This format has the advantage of containing all active passes with a lossless compression. The passes are a decomposition of the rendered picture (Diffuse, Shadows, Ambient Occlusion, and so on). In our case, we are going to save some time by only rendering the combined and the Z passes. The combined pass corresponds to the final image with all the different passes already combined, and the Z pass gives us a black and white image corresponding to the depth of the scene.

8. In the **Output** tab, we must choose an **Output** path. We will write the following address: `/Render\01\`. We will press *Enter* to validate the address. The `//` symbols create a file just next to the blend file.

9. It is a raw render, so we uncheck the **Compositing** option in the **Post Processing** tab (**Properties|Processing**).

Note

OpenEXR

This is a high dynamic-range (HDR) image file format created and used for special effects in the VFX industry. It is now a standard format supported by most of 3D and compositing softwares. The OpenEXR Multilayer format is a variation. It can hold unlimited layers and passes.

If you want more information about OpenEXR in Blender, visit this link:

http://blender.org/manual/data_system/files/image_formats.html#openexr

You are now ready to render the animation. You can press the **Animation** button to start rendering. We must repeat this process for each shot.

Enhance a picture with compositing

Now that we have a raw render, it is time to learn how to improve it using the compositing tools of Blender.

Introduction to nodal compositing

Blender is a complete tool that also allows compositing. This is the ability to edit an image or a sequence after the rendering phase. You probably have already tried compositing, maybe unknowingly. For example, Adobe Photoshop© is a software that allows us to composite a single image. Unlike Adobe Photoshop©, Blender uses a nodal system that provides a great flexibility. We can make changes at any point without the loss of information. Let's try this:

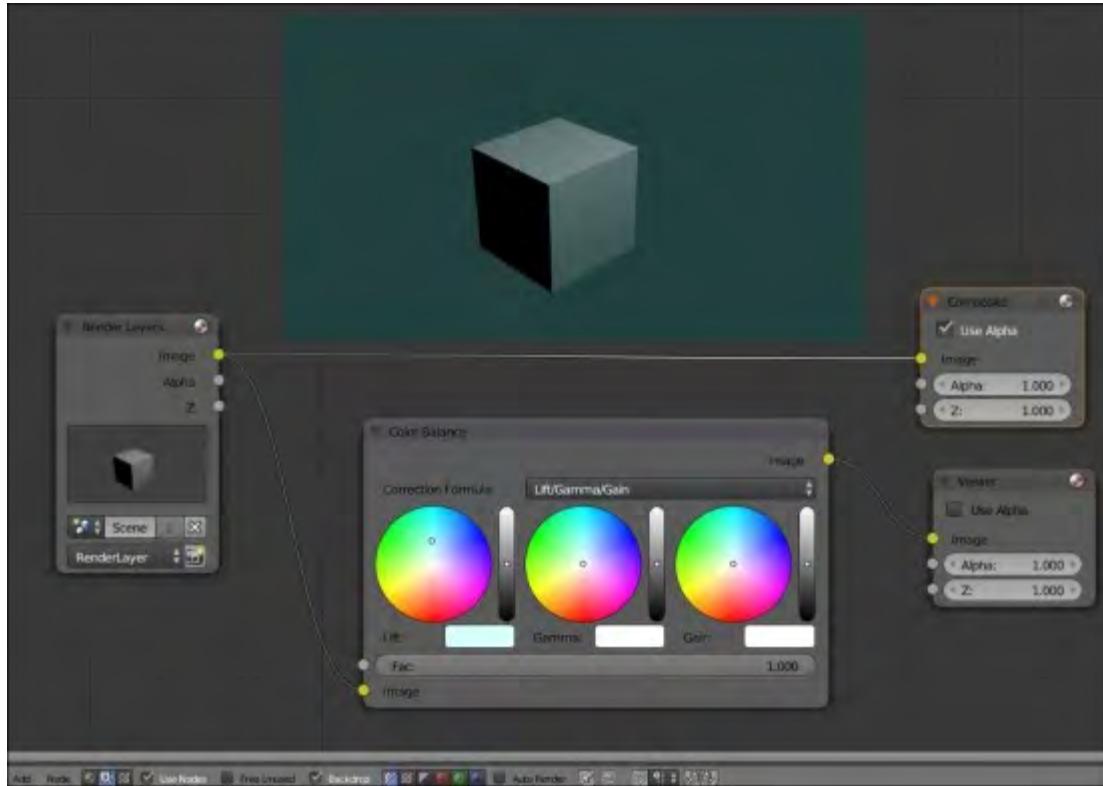
1. For a first approach of nodal compositing, let's open a new Blender scene.
2. In order to access the compositing mode, you must open the **Node Editor**. This is the same as the Node Editor for materials.
3. We must then check the **Compositing** button near the **Shader** button in the **Header** options. It is a small icon button symbolizing an image over another.
4. We must also check the **Use Nodes** button.

We have now two nodes, a Render Layer node and a Composite node.

1. We can split our scene to open the 3D View and make a render of the cube in the middle of the scene (**F12**).
2. We can also add a **Viewer** node (press *Shift + A* and select **Output | Viewer**). This node will allow us to visualize the compositing result directly in the **Node Editor**. You only need to connect the **Image** output socket of the **Render Layer** node to the **Image** input socket of the **Viewer** node and check the **Backdrop** option of the **Header**.

Now the render image appears behind the nodes. It will be pretty useful to do compositing in full screen. If you want to move the render image, use *Alt* and *MMB*. Two other interesting short keys are *V* to zoom in and *Alt + V* to zoom out.

1. We will add a **Color Balance** (press *Shift + A* and select **Color | Color Balance**) and connect it between the **Render Layers** node (to the **Image** output socket) and the **Viewer** node (to the **Image** input socket).
2. If we change the lift color, the render image is directly updated.



3. We can also replace the render of the cube by any other picture, by adding an **Image** node (press *Shift + A* and select **Input | Image**), and connecting the **Image** output socket to the **Image** input socket of the **Color Balance** node.

Note

Looking at a texture node through the Viewer

There is a node that can help you to better visualize what the compositing looks like at a certain point. You can press *Ctrl + Shift* and right-click on any node to append a **ViewNode** and connect to it.

The possibilities of compositing in Blender are enormous. For instance, you can easily use keying techniques that are often needed in the movie industry. It consists of replacing a green or a blue screen behind an actor by a virtual set. Compositing is an art and it takes too long to explain everything, but we will see some of its basic concepts so that we can improve the shots of our sequence.

Now, we are going to work on the first shot of the sequence:

1. As with any other image file, we must add an **Image** node (press *Shift + A* and select **Input | Image**) and connect it to the viewer.
2. We press the **Open** button of the **Image** node, and we take the corresponding OpenEXR Multilayer file. We also check **Auto-refresh**.

Depth Pass

The following is the combined output socket of the Image node, and there is a Depth output socket. This pass will allow us to simulate an atmospheric depth. It is an effect that can be observed when we look at a distant landscape and a kind of haze is formed. The Depth pass is a visual representation of the Z-Buffer on a grayscale. The objects near the camera will have a gray value close to black, unlike the distant elements, which will have a value close to white. This pass could serve for other things such as masking or blurring the focal depth. It all depends on the context. A controlled atmospheric effect may bring realism to the image.

1. We will start by adding a **Normalize** node (press *Shift + A* and select **Vector | Normalize**). This allows us to clamp all pixel values between 0 and 1. We cannot visualize the Depth pass without a Normalize node.
2. We will add **RGB Curves** (press *Shift + A* and select **Color | RGB Curves**) to change the contrast of the ZDepth pass and control its strength effect.
3. We will then add a **Mix** node (press *Shift + A* and select **Color | Mix**) with a **Mix** blend mode.

Now that we have added the nodes, we are going to connect them as follows:

1. We will need to connect the **Z** output of the **Image** node to the input of the **Normalize** node and the output of the **Normalize** to the **Image** input of the **RGB Curves** node.
2. We will also connect the **Image** output of the **RGB Curves** node to the **Fac** input of the **Mix** node and the **Combined** output socket of the **Image** node to the first **Image** input socket.
3. We must adjust the **RGB Curves** node by adding another point to the curve. The point is located at **X: 0.36667** and **Y: 0.19375**.
4. We will also need to change the color of the second **Image** input socket of the **Mix** node. The hex code of the color is **D39881**. It will color the white pixels.



A render before and after the ZDepth pass



Color correction of the shot

One of the most important aspects of compositing is color calibration. Fortunately, there are easy-to-use tools in Blender to do that.

1. In order to quickly change the hue, we will add a **Color Balance** node (press *Shift +A* and select **Color | Color Balance**). The hue corresponds to the color tint of the image.
2. We must be very careful to slightly change the value of **Lift**, **Gamma**, and **Gain**. They become strong quickly. The RGB values of Lift are **R: 1.000**, **G: 0.981**, and **B: 0.971**. The RGB values of Gamma are **R: 1.067**, **G: 1.08**, and **B: 1.068**. The RGB values of Gain are **R: 1.01**, **G: 0.998**, and **B: 0.959**.

Note

There are two correction formulas: **Lift/Gamma/Gain** and **Offset/Power/Slope**. These are the two ways to get the same result. For each one, there are three color wheels and a value controller (**Fac**). You can modify the darker, the mid-tone, and the highlight values separately.

If you want more information about the color balance node in Blender, visit this link :

https://www.blender.org/manual/composite_nodes/types/color/color_balance.html

A render image with an adjustment of the **Color Balance** node will look like this:

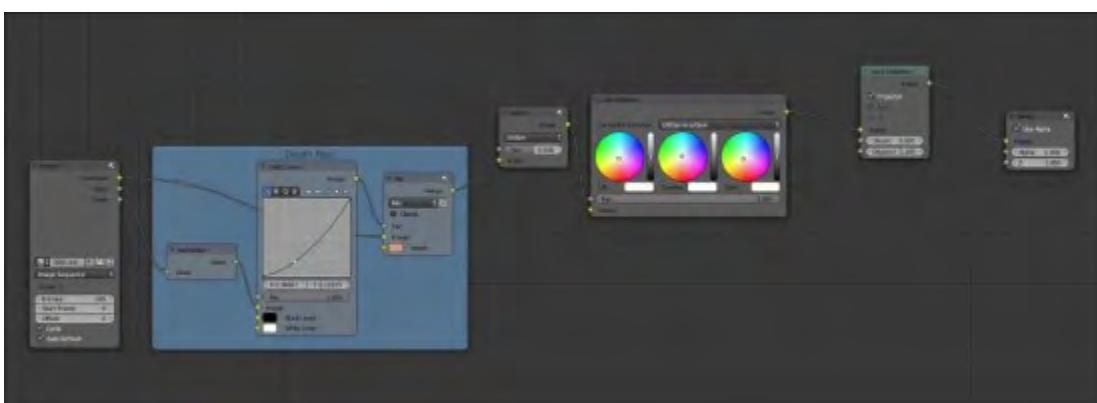


Before finishing the compositing, let's add a few effects.

Adding effects

We will add a **Filter** node (press *Shift + A* and select **Filter | Filter**). We must connect the **Image** output socket of the **Mix** node to the **Image** input socket of the **Softten** node, and we will connect the image output socket of the **Softten** node to the **Image** input socket of the **color balance** node. We will keep the filter type to **Softten** with **Fac** of **0.500**. This blends the pixels so that the image is less sharp. A photo is never perfectly sharp.

We will add then a **Lens distortion** node (press *Shift + A* and select **Distort | Lens Distortion**). We must connect the **Image** output socket of the **Color Balance** to the **Image** input socket of the **Lens Distortion** node and the **Image** output socket of the **Lens Distortion** node to the **Image** input socket of the **Viewer** node. We will check the **Projector** button. The **distort** option is at **0.000**, and the **dispersion** option at **0.100**. This node usually allows us to make a distortion effect such as a fish eye, but in our case, it will allow us to make a chromatic aberration. This adds a soft and nice effect.



The nodes of compositing

We have now completed the appearance of the shot.



A render with final compositing

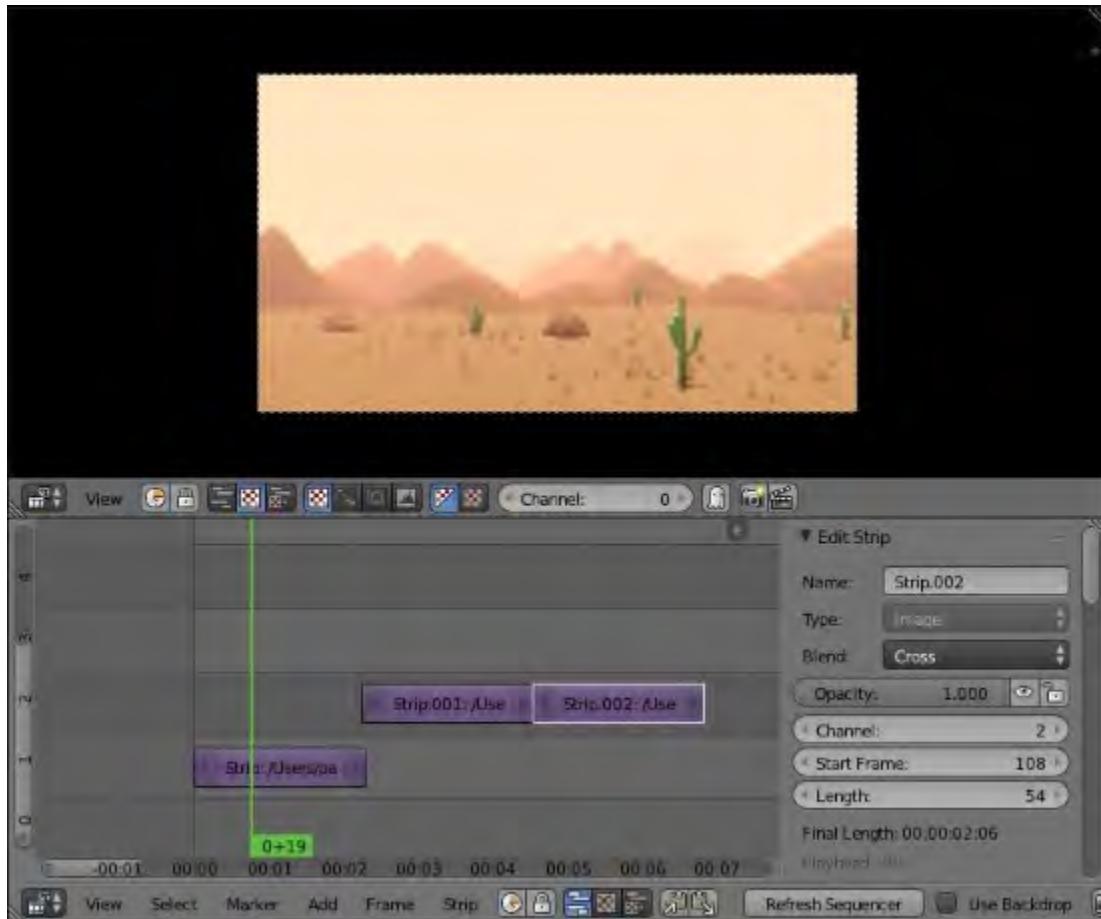
Compositing rendering phase

We are now ready to make the render of our compositing:

1. In the **Output** options (**Properties | Render | Output**), we must change the Output path. We will write the following address: **//Render\01\Compositing**. We will press *Enter* to validate the address.
2. We will also change the Output format to **TGA**.
3. It is a compositing render, so we will check the **Compositing** option in the **Post Processing** tab (**Properties | Processing**).
4. Now, we are ready to render the scene. We will use the same process for each scene.

Editing the sequence with the VSE

Now that we've rendered and done some compositing on each shot, it's time to bring back the whole sequence in one final place. In this section, we are going to do a basic video editing with the VSE.



Two VSE, one is set to the Image Preview (top), the other to the Sequencer (bottom)

Introduction to the Video Sequence Editor

The **VSE** or Video Sequence Editor is a method of video editing in Blender. It is really simple to use and could be very powerful. The best way to use it is to use the **Video Editing** layout located in the menu bar. We usually don't use the Graph editor here, so we can join it back. We have now an interface with two Video Sequence Editors and the Timeline at the bottom. On the head of the VSE, we have three icons that are used to display **Sequencer**, the place where we edit strips, and **Image Preview**, where we see the result of our editing, or both. In Sequencer, we can add different types of strips with *Shift + A*. We are mainly using **Image**, **Movie**, or **Sound**. We can import Image Sequences with the **Image** option. You can select a strip with RMB and move it with *G*. When you have a strip selected, you have access to two buttons to the left and right represented with arrows that define the start and the end of the strip. You can cut a strip by placing the timeline where you want the split to be and pressing *K* (for knife).

We are not going to go deep into every setting of the VSE, but for each selected strip, you have some options in the right panel of the editor (*N*), such as the **opacity** of an image or movie strip or the volume of an **audio** strip. Of course, you can animate each option by right-clicking on them and choosing the **Insert Keyframe** option, or by simply pressing *I* while hovering over them. You can press *Ctrl* to snap a selected strip to another strip. You can also use the *Shift + D* shortcut in order to duplicate a selected strip.

Edit and render the final sequence

Let's now create the editing of our sequence with the shots that we've composited and rendered before:

1. In order to edit our sequence, we will open a new fresh file. We will also change the layout of our interface so that we have two **VSEs**, one with **Image Preview** and the other with **Sequencer**.
2. Now, we can add our first shot by pressing *Shift + A* in the sequencer and by choosing **Image**. In the file browser, we will go to the **Render** folder and select the **01-compositing** folder in order to select every **.targa** file with *A*. We will now have a new strip that corresponds to the first shot. We will repeat the same process with the other shots.
3. Now, we can move each shot one after the other to create continuity. Be sure that each strip is snapped to the previous one by pressing the *Ctrl* key while moving them.
4. We will also need to readjust the animation start and end in the timeline from the beginning of sequence to its end frame.
5. If you want, you can add **Sound** strips in order to add music or sound effects.
6. We are now ready to render our final edited sequence. To do this, we will change the **Output** path in the **Render** tab of the Properties editor, and we will choose the **H.264** file type. In the **Encoder** section, we will use the **Quicktime** type, and we will, finally, press the render button.

Summary

First, congratulation for arriving at this point of the book; we hope you've learned a lot of things and that you can now realize all the ideas that you ever dreamed about. In this last chapter, we learned how to finalize our sequence by creating advance materials. We also learned how to use the particle system to create a complex fur. Then we set up our render with the OpenEXR MultiLayer format and discovered what passes are. After this, we saw the power of nodal compositing by changing the color balance and adding effects to our shot. As a bonus, we learned how to use the VSE in order to edit our full sequence. Be aware that we didn't explain a lot of things, and you can go deeper into each subject. We recommend that you practice a lot and skim the web and the other books of the PacktPub collection in order to extend your knowledge. Remember, you can learn a tool, but creativity is one of the most important things in this field. We wish you a successful continuation.

Appendix A. Bibliography

This Learning path is a blend of text and projects, all packaged up keeping your journey in mind. It includes content from the following Packt books:

- *Blender 3D Cookbook - Second Edition* by Enrico Valenza
- *Blender 3D Incredible Machines - Second Edition* by Christopher Kuhn
- *Blender 3D By Example* by Romain Caudron and Pierre-Armand Nicq

Index

A

- actions
 - tweaking, in Graph Editor / [Tweaking the actions in Graph Editor](#), [Getting ready](#), [How to do it...](#)
- additional UV layers
 - setting up / [Setting up additional UV layers](#), [How to do it...](#)
- advanced materials, Cycles
 - creating / [Creating advanced materials in Cycles](#)
 - skin material, with Subsurface Scattering node / [Skin material with Subsurface Scattering](#)
 - skin material, with Subsurface Scattering node / [Skin material with Subsurface Scattering](#)
 - eye material / [Eye material](#)
 - fur, creating / [The fur of the rat](#)
- alien character
 - retopology, creating / [Making the retopology of the alien character](#)
- alien character retopology
 - creating / [Making the retopology of the alien character](#)
 - environment, preparing / [Preparing the environment](#)
- alpha / [Anatomy of a brush](#)
- Alt key / [Using a pen tablet](#)
- ambient occlusion
 - baking / [The baking of an ambient occlusion](#)
 - about / [Understanding the ambient occlusion map](#)
 - colors, multiplying / [Understanding the ambient occlusion map](#)
 - bake, creating / [Creation of the bake](#)
 - displaying, in viewport / [Displaying the ambient occlusion in the viewport](#)
- animation
 - OpenGL playblast, rendering of / [Rendering an OpenGL playblast of the animation](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - principles / [Principles of animation](#)
 - preparation / [Preparation of the animation](#)
- animation, preparing
 - short script, writing / [Writing a short script](#)
 - shot, field sizes / [Writing a short script](#)
 - shot, angle shot / [Writing a short script](#)
 - Storyboard, creating / [Making a storyboard](#)
 - final camera placements, searching / [Finding the final camera placements and the timing through a layout](#)
 - final camera timing, searching / [Finding the final camera placements and the timing through a layout](#)
 - references / [Animation references](#)
 - organization / [Organization](#)

- animation controls
 - building / [Building the animation controls and the Inverse Kinematic](#), [How to do it...](#)
- animation principles
 - about / [Principles of animation](#)
 - Squash and Stretch / [Squash and Stretch](#)
 - anticipation / [Anticipation](#)
 - staging / [Staging](#)
 - Straight Ahead Action / [Straight Ahead Action and Pose to Pose](#)
 - Pose to Pose / [Straight Ahead Action and Pose to Pose](#)
 - Follow Through / [Follow Through and Overlapping Action](#)
 - Overlapping Action / [Follow Through and Overlapping Action](#)
 - Slow In / [Slow In and Slow Out](#)
 - Slow Out / [Slow In and Slow Out](#)
 - arcs / [Arcs](#)
 - Secondary Action / [Secondary Action](#)
 - timing / [Timing](#)
 - exaggeration / [Exaggeration](#)
 - solid drawing / [Solid drawing](#)
 - appeal / [Appeal](#)
- animation process
 - URL / [See also](#)
- animations, rendering
 - URL / [See also](#)
- animation tools, Blender
 - about / [Animation tools in Blender](#)
 - timeline editor / [The timeline](#)
 - keyframe / [What is a keyframe?](#)
 - Dope Sheet / [The Dope Sheet](#)
 - Graph editor / [The Graph editor](#)
 - Non-Linear Action editor / [The Non-Linear Action editor](#)
- Anticipation principle / [Anticipation](#)
- appeal principle / [Appeal](#)
- Append and Link
 - URL / [How it works...](#)
- arcs principle / [Arcs](#)
- Armature
 - about / [Using the Skin modifier's Armature option](#), [Introduction](#), [Introduction](#), [How to do it...](#)
 - skinning, to Gidiosaurus mesh / [Introduction](#)
 - parenting, Automatic Weights tool used / [Parenting the Armature and Mesh using the Automatic Weights tool](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
- Armature, character
 - building, from scratch / [Building the character's Armature from scratch](#), [How to do it...](#)
 - rig, building for secondary parts / [Building the rig for the secondary parts](#)
 - rig, completing / [Completing the rig](#), [How it works...](#)

- perfecting, as rig for Armor / [Perfecting the Armature to also function as a rig for the Armor](#), [How to do it...](#), [How it works...](#)
 - building, through Human Meta-Rig tool / [Building the character's Armature through the Human Meta-Rig](#), [Getting ready](#), [How to do it...](#)
 - generating, by Rigify add-on / [Generating the character's Armature by using the Rigify add-on](#), [How to do it...](#), [How it works...](#)
- Armature modifier
 - URL / [Editing the mesh](#)
- Armature option, Skin modifier
 - using / [Using the Skin modifier's Armature option](#), [Getting ready](#), [How to do it...](#)
- Armor
 - detailing, Curve used from Mesh tool / [Detailing the Armor by using the Curve from Mesh tool](#), [Getting ready](#), [How to do it...](#), [There's more...](#)
- armor plates
 - modeling / [Modeling the armor plates](#), [How to do it...](#), [How it works...](#)
- armor shaders
 - building, in Cycles / [Building the armor shaders in Cycles](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - building, in Blender Internal / [Building the armor shaders in Blender Internal](#), [How to do it...](#), [How it works...](#), [There's more...](#)
- Armor textures
 - defining / [The Armor textures](#), [How to do it...](#), [There's more...](#), [See also](#)
 - references / [See also](#)
- Array modifier / [Adding instantiated objects](#)
- Artistic Anatomy
 - about / [An introduction to artistic anatomy](#)
 - body, sculpting / [Sculpting the body](#)
- AutoKey option / [What is a keyframe?](#)
- Automatic Weights tool
 - used, for parenting Armature / [Parenting the Armature and Mesh using the Automatic Weights tool](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - used, for parenting Mesh / [Parenting the Armature and Mesh using the Automatic Weights tool](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - about / [Getting ready](#)

B

- B-bones
 - about / [How it works...](#)
- Background Images tool
 - about / [Introduction](#)
 - templates, setting with / [Setting templates with the Background Images tool](#), [How to do it...](#)
 - using / [Setting templates with the Background Images tool](#), [How to do it...](#)
- base mesh
 - about / [Introduction](#)
 - building / [Introduction](#)

- preparing, for sculpting / [Preparing the base mesh for sculpting](#), [How to do it...](#), [How it works...](#)
 - creating, with Skin modifier / [Creating a base mesh with the Skin modifier](#)
 - Matcap, using / [Visual preparation](#)
 - visual preparation / [Visual preparation](#)
- base mesh, character
 - building, with Skin modifier / [Building the character's base mesh with the Skin modifier](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - sculpting / [Sculpting the character's base mesh](#), [How to do it...](#), [There's more...](#)
- basic settings, Cycles render engine
 - about / [Understanding the basic settings of Cycles](#)
 - sampling / [The sampling](#)
 - light path settings / [Light path settings](#)
 - performance settings / [Performances](#)
- bevel modifier / [Simulate a stack of wooden planks with physics](#)
- Bevel tool
 - about / [An introduction to the Subdivision Surface modifier](#)
- Big Buck Bunny
 - URL / [How it works...](#)
- Blender
 - URL / [Performances](#), [Cycles versus Blender Internal](#)
 - bone anatomy / [An introduction to the rigging process](#)
 - animation tools / [Animation tools in Blender](#)
- Blender bug tracker
 - URL / [There's more...](#)
- Blender Internal
 - tileable scales image, creating / [Making a tileable scales image in Blender Internal](#), [Getting ready](#), [How to do it...](#)
 - color maps, painting in / [Painting the color maps in Blender Internal](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - reptile skin shaders, building in / [Building the reptile skin shaders in Blender Internal](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - eyes' shaders, building in / [Building the eyes' shaders in Blender Internal](#), [How to do it...](#), [How it works...](#)
 - armor shaders, building in / [Building the armor shaders in Blender Internal](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - three-point lighting rig, setting in / [Setting a three-point lighting rig in Blender Internal](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - used, for rendering robot toy / [Using Blender Internal to render our Robot Toy](#)
 - used, for rendering / [Doing a quick render with Blender Internal](#)
 - lights, setting / [Setting lights](#)
 - camera, placing / [Placing the camera](#)
 - environment, setting / [Setting the environment \(sky and mist\)](#)
 - versus Cycles render engine / [Cycles versus Blender Internal](#)
- Blender Internal shaders
 - creating / [Introduction](#)
- Blender Render engine

- about / [Introduction](#)
 - URL / [Getting ready](#)
- Blender units / [Working with a scale](#)
- blocking method
 - about / [Blocking the bases of the house](#)
 - refining / [Refining the blocking](#)
 - instantiated objects, refining / [Adding instantiated objects](#)
- blocking objects
 - reworking / [Reworking the blocking objects](#)
- body parts
 - joining / [How to do it...](#)
- body sculpting, Artistic Anatomy
 - head / [The head](#)
 - torso / [The torso](#)
 - arms / [The arms](#)
 - legs / [The legs](#)
 - belt / [The belt](#)
- bones, Armature
 - URL / [Editing the mesh](#)
- Boolean modifier
 - using / [There's more...](#)
- Breakdowns / [Straight Ahead Action and Pose to Pose](#)
- Bridge tool
 - about / [How to do it...](#)
- brush
 - anatomy / [Anatomy of a brush](#)
- brushes, Texture Paint tool
 - discovering / [Discovering the brushes](#)
 - TexDraw brush / [The TexDraw brush](#)
 - Smear brush / [The Smear brush](#)
 - Soften brush / [The Soften brush](#)
 - Clone brush / [The Clone brush](#)
 - Fill brush / [The Fill brush](#)
 - Mask brush / [The Mask brush](#)

C

- character
 - animating / [Introduction](#)
 - linking / [Linking the character and making a proxy](#), [Getting ready](#), [How to do it...](#)
- Child Of constraint / [Rigging the gun](#)
- Clay brush / [The torso](#)
- Clay Strips brush / [The head](#)
- Clone brush
 - using / [There's more...](#)
 - / [The Clone brush](#)
- color balance node
 - URL / [Depth Pass](#)

- color maps
 - painting, in Blender Internal / [Painting the color maps in Blender Internal](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - painting, in Cycles / [Painting the color maps in Cycles](#), [Getting ready](#), [How to do it...](#)
- colors
 - adding / [Adding colors](#)
 - Texture Paint tool, basics / [Basics of the Texture Paint tool](#)
 - scene, painting / [Painting the scene](#)
 - laying down / [Laying down the colors](#)
 - tiled textures / [Tiled textures](#)
 - roof-tiled texture, painting / [Painting the roof-tile texture](#)
 - hand painted tiled textures types, tips / [Quick tips for other kinds of hand-painted tiled textures](#)
 - tiled textures, baking / [Baking our tiled textures](#)
 - transparent textures, creating / [Creating transparent textures](#)
- color wheel window / [The grass texture](#)
- composite nodes
 - URL / [How it works...](#)
- control bones
 - creating / [How to do it...](#)
- Copy Attributes Menu add-ons
 - about / [Getting ready](#)
- Copy Rotation constraint
 - about / [How it works...](#)
- / [The arm and the hand](#)
- cranium/neck section, mesh / [How to do it...](#)
- Crease brush / [The head](#)
- Cube primitive
 - about / [Getting ready](#)
- curves
 - URL / [There's more...](#)
 - used, for modeling tree / [Modeling a tree with curves](#)
 - about / [Modeling a tree with curves](#)
- custom shapes
 - about / [Custom shapes](#)
- Cycles
 - color maps, painting in / [Painting the color maps in Cycles](#), [Getting ready](#), [How to do it...](#)
 - reptile skin shaders, building / [How to do it...](#), [How it works...](#), [There's more...](#)
 - eyes shaders, building / [Building the eyes' shaders in Cycles](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - armor shaders, building / [Building the armor shaders in Cycles](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - advanced materials, creating / [Creating advanced materials in Cycles](#)
 - passes / [The Raw rendering phase](#)
 - URL / [The Raw rendering phase](#)
- cycles

- Normal Maps, applying in Cycles / [Making and applying normal maps in Cycles](#)
- Cycles materials and textures
 - URL / [See also](#)
- Cycles Render engine
 - defining / [Image based lighting in Cycles](#)
 - rendering settings, tweaking / [Obtaining a noise-free and faster rendering in Cycles](#), [Getting ready](#), [How to do it...](#)
- Cycles render engine
 - basic settings / [Understanding the basic settings of Cycles](#)
 - realistic grass, creating / [Creating realistic grass](#)
 - textures, backing / [Baking textures in Cycles](#)
 - versus Blender Internal / [Cycles versus Blender Internal](#)

D

- 3D Cursor
 - about / [How to do it...](#)
- 3D manipulator widget
 - enabling / [How to do it...](#), [Getting ready](#)
- 3D scene layout
 - setting up / [Setting the library and the 3D scene layout](#), [How to do it...](#), [How it works...](#)
- Damped Track constraint / [The head and the eyes](#)
- data system
 - URL / [How it works...](#)
- Depth pass / [Depth Pass](#)
- display, Render
 - references / [Obtaining a noise-free and faster rendering in Cycles](#)
- DJV Imaging
 - URL / [There's more...](#)
- Dope Sheet / [The Dope Sheet](#)
- drivers
 - assigning, to shape keys / [Assigning drivers to the shape keys](#), [Getting ready](#), [How to do it...](#), [There's more...](#)
 - about / [How it works...](#)
 - URL / [Setting movement limit constraints](#)
- Dumped Track constraint
 - about / [How it works...](#)
- Duplicate Linked tool / [Modeling the buttons](#), [Adding instantiated objects](#)
- Duplicate Object / [Modeling the buttons](#)
- Dynamic topology feature
 - using / [Using the Multiresolution modifier and the Dynamic topology feature](#), [How to do it...](#), [How it works...](#)
- Dynamic topology setting
 - about / [How it works...](#)
- Dyntopo
 - versus Multires modifier / [Dyntopo versus the Multires modifier](#)
 - first touch / [First touch with Dyntopo](#)

E

- edge-loops flow
 - planning, Grease Pencil tool used / [Using the Grease Pencil tool to plan the edge-loops flow](#), [Getting ready](#), [How to do it...](#)
- edge loop
 - about / [Modeling the head](#)
- edge loops / [Creating a base mesh with the Skin modifier](#)
- Empties
 - preparing / [Getting ready](#)
- environment
 - modeling / [Modeling the environment \(8 pages\)](#)
 - cliff, modeling / [Modeling the cliff](#)
 - tree with curves, modeling / [Modeling a tree with curves](#)
 - scene, enhancing with barrier / [Enhancing the scene with a barrier, rocks, and a cart](#)
 - scene, enhancing with rocks / [Enhancing the scene with a barrier, rocks, and a cart](#)
 - scene, enhancing with carts / [Enhancing the scene with a barrier, rocks, and a cart](#)
- environment preparation, alien character retopology
 - steps / [Preparing the environment](#)
 - head / [The head](#)
 - polygon pairs / [The head](#)
 - F2 add-on / [The head](#)
 - smooth option / [The head](#)
 - neck / [The neck and the torso](#)
 - torso / [The neck and the torso](#)
 - arms / [The arms and the hands](#)
 - hands / [The arms and the hands](#)
 - legs / [The legs](#)
- exaggeration principle / [Exaggeration](#)
- expressions shape keys
 - defining / [How to do it...](#)
- Extremes / [Straight Ahead Action and Pose to Pose](#)
- extrusion tool
 - about / [Modeling the head](#)
- eye, creature
 - modeling / [Modeling the eye](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - building / [How to do it...](#), [How it works...](#)
- eyeball rotation
 - transferring, to eyelids / [Transferring the eyeball rotation to the eyelids](#), [Getting ready](#)
- eyes' shaders
 - building, in Blender Internal / [Building the eyes' shaders in Blender Internal](#), [How to do it...](#), [How it works...](#)
- eyes shaders
 - building, in Cycles / [Building the eyes' shaders in Cycles](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
- Eyes sphere
 - eyeballs / [Building the eyes' shaders in Cycles](#)

- irises / [Building the eyes' shaders in Cycles](#)
- pupils / [Building the eyes' shaders in Cycles](#)

F

- fac input / [Creating the materials of the house, the rocks, and the tree](#)
- fangs
 - about / [Modeling the armor plates](#)
- Fill brush / [The Fill brush](#)
- FK (Forward Kinematic) / [The leg and the foot](#)
- FK/IK sliders
 - about / [How to do it...](#)
- Flatten/Contrast brush / [The legs](#)
- Follow Through principle / [Follow Through and Overlapping Action](#)
- fork, robot toy
 - modeling / [Modeling the fork](#)
 - protections, modeling / [Modeling protections for the fork](#)
- Fresnel / [Creating the materials of the house, the rocks, and the tree](#)
- Fresnel node
 - about / [How it works...](#)

G

- Gidiosaurus
 - about / [How to do it...](#), [Introduction](#), [Building the character's Armature from scratch](#), [How it works...](#), [Perfecting the Armature to also function as a rig for the Armor](#)
- Gidiosaurus character
 - defining / [Introduction](#)
- Gidiosaurus object / [Getting ready](#)
- Gimp
 - about / [Making a tileable scales image in Blender Internal](#), [The Quick Edit tool](#), [How to do it...](#)
- Gizmo tool / [The Edit Mode versus the Object Mode](#)
- Global Illumination effect
 - about / [Getting ready](#)
- GPU graphic cards, for Cycles
 - URL / [See also](#)
- GPU rendering
 - URL / [See also](#)
- Grab (G) tool / [Modeling the cliff](#)
- Grab brush / [The head](#)
- Grab tool / [Adding the head primitive](#)
- Graph editor / [The Graph editor](#)
- Graph Editor
 - actions, tweaking in / [Tweaking the actions in Graph Editor](#), [Getting ready](#), [How to do it...](#)
 - URL / [Using the Non Linear Action Editor to mix different actions](#)
- Grease Pencil tool

- about / [How it works...](#), [How to do it...](#)
- used, for planning edge-loops flow / [Using the Grease Pencil tool to plan the edge-loops flow](#), [Getting ready](#), [How to do it...](#)
- URL / [There's more...](#)
- groups
 - using / [Grouping objects](#)
- Gstretch tool
 - about / [How to do it...](#)
- guides
 - creating / [How to do it...](#)

H

- haunted house
 - blocking / [Blocking the house](#)
 - world scale, working on / [Working with a scale](#)
 - bases, blocking / [Blocking the bases of the house](#)
 - element geometry, working on / [Breaking and ageing the elements](#)
 - stack simulation of wooden planks, adding / [Simulate a stack of wooden planks with physics](#)
 - stack simulation of wooden planks, creating / [Creation of the simulation of a stack of planks](#)
- HDR (High Dynamic Range) image
 - about / [Setting image based lighting \(IBL\)](#)
- hdr image
 - about / [How to do it...](#)
- head deformation, Gidiosaurus mesh
 - fixing / [How to do it...](#)
- high resolution mesh
 - more details, sculpting on / [Sculpting more details on the high resolution mesh](#), [Getting ready](#), [How to do it...](#)
- Hooks
 - using / [Using the Laplacian Deform modifier and Hooks](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - URL / [How it works...](#)
- Human Meta-Rig
 - about / [Introduction](#)
- Human Meta-Rig tool
 - used, for building Armature / [Building the character's Armature through the Human Meta-Rig](#), [Getting ready](#), [How to do it...](#)
 - about / [Building the character's Armature through the Human Meta-Rig](#)
 - using / [Getting ready](#)

I

- IBL
 - painting / [Painting and using an Image Base Lighting](#)
 - using / [Painting and using an Image Base Lighting](#)

- IK constraint / [The leg and the foot](#)
- IK solver
 - about / [How to do it...](#)
- image based lighting (IBL)
 - setting / [Setting image based lighting \(IBL\)](#), [Image based lighting in Cycles](#), [Image based lighting in Blender Internal](#), [How it works...](#)
 - in Cycles / [Image based lighting in Cycles](#)
 - in Blender Internal / [Image based lighting in Blender Internal](#), [How it works...](#)
- Image Empties method
 - about / [Introduction](#), [Setting templates with the Image Empties method](#)
 - templates, setting with / [Setting templates with the Image Empties method](#), [How to do it...](#), [How it works...](#)
- images
 - requisites / [Introduction](#)
- Images as Planes add-on
 - about / [Introduction](#), [Setting templates with the Images as Planes add-on](#)
 - templates, setting with / [Setting templates with the Images as Planes add-on](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
- image texture
 - Vertex Colors layer, baking to / [Adding a dirty Vertex Colors layer and baking it to an image texture](#), [How to do it...](#), [How it works...](#)
- Inflate/Deflate brush / [The head](#)
- inset
 - about / [Modeling the antenna](#)
- Inverse Kinematic
 - building / [Building the animation controls and the Inverse Kinematic](#), [How to do it...](#)
- Inverse Kinematic (IK) / [The leg and the foot](#)
- Inverse Kinematics
 - about / [How to do it...](#)
- Inverse Kinematics constraint
 - about / [How it works...](#), [How to do it...](#)

J

- join tool (J) / [Modeling the thunderbolts](#)

K

- keyframe / [What is a keyframe?](#)
- Key Poses / [Straight Ahead Action and Pose to Pose](#)
- knife tool (K) / [Modeling the thunderbolts](#), [Modeling the arm](#)
- Knife Topology Tool
 - about / [How to do it...](#), [How it works...](#)
- Knife Topology Tool (K) / [Modeling the chest](#)
- Krita
 - URL / [Making a tileable scales image in Blender Internal](#)

L

- Laplacian Deform modifier
 - using / [Using the Laplacian Deform modifier and Hooks](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - URL / [How it works...](#)
- Laplacian modifier
 - setting up / [How to do it...](#)
- lattice
 - about / [Modeling a tree with curves](#)
- Lattice object / [How to do it...](#)
- Layered Painting, in Blender 2.72
 - references / [How it works...](#)
- library
 - setting up / [Setting the library and the 3D scene layout](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
- light path settings, Cycles render engine
 - Max and Min Bounces / [Light path settings](#)
 - Filter Glossy / [Light path settings](#)
 - reflective and refractive caustics / [Light path settings](#)
- lights
 - settings / [How to do it...](#)
 - about / [Lighting](#)
 - testing material, creating / [Creating a testing material](#)
 - types / [Understanding the different types of light](#)
 - using, in scenes / [Lighting our scene](#)
 - IBL, painting / [Painting and using an Image Base Lighting](#)
 - IBL, using / [Painting and using an Image Base Lighting](#)
- light types
 - about / [Understanding the different types of light](#)
 - point / [Understanding the different types of light](#)
 - sun / [Understanding the different types of light](#)
 - spot / [Understanding the different types of light](#)
 - hemi / [Understanding the different types of light](#)
 - area / [Understanding the different types of light](#)
- Link and Append file structures / [Organization](#)
- Linux Ubuntu
 - about / [How to do it...](#), [Getting ready](#)
- Live Unwrap tool / [Editing the UV islands](#)
- Locked Track constraint
 - about / [How it works...](#)
- Loft tool
 - about / [How to do it...](#)
- LoopCut tool
 - about / [An introduction to the Subdivision Surface modifier](#)
- LoopTools add-on
 - about / [Getting ready](#)

- using / [Modeling the armor plates](#), [How to do it...](#)
- URL / [How it works...](#)
- used, for re-topologizing mesh / [Using the LoopTools add-on to re-topologize the mesh](#), [How to do it...](#)
- enabling / [Getting ready](#)
- low poly mesh / [The legs](#)
- low resolution mesh
 - preparing, for unwrapping / [Preparing the low resolution mesh for unwrapping](#), [Getting ready](#), [How to do it...](#)

M

- mask / [The head](#)
- Mask brush / [The legs](#), [The Mask brush](#)
- Matcap
 - using / [Visual preparation](#)
- Matcaps
 - about / [How it works...](#)
- materials
 - creating, with nodes / [Creating materials with nodes](#)
 - of house, creating / [Creating the materials of the house, the rocks, and the tree](#)
 - of rock, creating / [Creating the materials of the house, the rocks, and the tree](#)
 - of tree, creating / [Creating the materials of the house, the rocks, and the tree](#)
 - mask, adding for windows / [Adding a mask for the windows](#)
 - procedural textures, using / [Using procedural textures](#)
 - Normal Maps, creating in Cycles / [Making and applying normal maps in Cycles](#)
- materials, Blender Render Engine
 - URL / [Building the eyes' shaders in Blender Internal](#)
- mesh
 - editing / [Editing the mesh](#), [How to do it...](#)
 - re-topologizing, Snap tool used / [Using the Snap tool to re-topologize the mesh](#), [How to do it...](#), [How it works...](#)
 - re-topologizing, Shrinkwrap modifier used / [Using the Shrinkwrap modifier to re-topologize the mesh](#), [How to do it...](#)
 - re-topologizing, LoopTools add-on used / [Using the LoopTools add-on to re-topologize the mesh](#), [How to do it...](#)
 - unwrapping / [UV unwrapping the mesh](#), [How to do it...](#)
 - modifying / [Modifying the mesh and the UV islands](#), [Getting ready](#), [How to do it...](#)
- Mesh
 - parenting, Automatic Weights tool used / [Parenting the Armature and Mesh using the Automatic Weights tool](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
- Mesh Deform modifier
 - used, to skin character / [Using the Mesh Deform modifier to skin the character](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
- Mesh to Curve technique
 - used, for adding details / [Using the Mesh to Curve technique to add details](#), [How to do it...](#), [How it works...](#)

- metric system / [Working with a scale](#)
- mirror modifier / [Modeling the thunderbolts](#)
- mist pass
 - compositing / [Compositing a mist pass](#)
- model
 - preparing, for using UDIM UV tiles / [Preparing the model to use the UDIM UV tiles, How to do it..., How it works...](#)
- modeling tools, robot toy
 - using / [Using the basic modeling tools](#)
 - extrusion / [Modeling the head](#)
- modifier
 - about / [An introduction to the Subdivision Surface modifier](#)
- morphing
 - about / [Introduction](#)
- movement limit constraints
 - setting / [Setting movement limit constraints, How to do it...](#)
 - URL / [How to do it...](#)
- Multires modifier
 - about / [Dyntopo versus the Multires modifier](#)
 - first touch / [First touch with the Multires modifier](#)
- Multiresolution modifier
 - using / [Using the Multiresolution modifier and the Dynamic topology feature, How to do it..., How it works...](#)
 - about / [How it works...](#)
- Multiresolution modifier method
 - defining / [How to do it...](#)
- MyPaint
 - URL / [Making a tileable scales image in Blender Internal](#)

N

- N-Gon / [Modeling the antenna](#)
- negative frames keys
 - about / [How to do it...](#)
- negative low value
 - about / [There's more...](#)
- NLA Editor
 - about / [How to do it...](#)
 - used, to mix different actions / [Using the Non Linear Action Editor to mix different actions, How to do it...](#)
 - URL / [How to do it...](#)
- nodal compositing / [Introduction to nodal compositing](#)
- node group, of skin shader
 - creating, for reusing / [Making a node group of the skin shader to reuse it, How to do it..., How it works...](#)
- noise reduction
 - URL / [See also](#)
- Non-Linear Action editor / [The Non-Linear Action editor](#)

- Non-Photorealistic Rendering (NPR)
 - about / [Introduction](#)
- normal map
 - baking / [The baking of a normal map](#)
 - about / [What is a normal map?](#)
 - creating, with bake tools / [Making of the bake](#)
 - displaying, in viewport / [Displaying the normal map in the viewport](#)
- normals, of sculpted mesh
 - baking, on low resolution / [Baking the normals of the sculpted mesh on the low resolution one](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)

O

- OpenEXR
 - about / [The Raw rendering phase](#)
 - URL / [The Raw rendering phase](#)
- OpenGL playblast
 - rendering, of animation / [Rendering an OpenGL playblast of the animation](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
- OpenGL rendering
 - starting / [How to do it...](#)
- origin/pivot / [Modeling the fork](#)
- outliner / [Modeling the thunderbolts](#)
- Overlapping Action principle / [Follow Through and Overlapping Action](#)

P

- Paint Tool
 - using / [Preparing the model to use the UDIM UV tiles](#)
 - used, for fixing seams / [Painting to fix the seams and to modify the baked scales image maps](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - used, for modifying baked scales image maps / [Painting to fix the seams and to modify the baked scales image maps](#), [Getting ready](#), [How to do it...](#), [How it works...](#), [There's more...](#)
 - about / [How to do it...](#)
- pelvis island
 - editing / [How to do it...](#)
- pen tablet
 - using / [Using a pen tablet](#)
- performance considerations
 - URL / [See also](#)
- performance settings, Cycles render engine
 - Viewport BVH Type / [Performances](#)
 - Tiles / [Performances](#)
- PET
 - enabling / [How to do it...](#)
- Photoshop

- about / [Making a tileable scales image in Blender Internal](#), [The Quick Edit tool](#), [How to do it...](#)
- picture
 - enhancing, with composing / [Enhance a picture with compositing](#)
 - nodal compositing / [Introduction to nodal compositing](#)
 - Depth pass / [Depth Pass](#)
 - effects, adding / [Adding effects](#)
 - rendering phase, compositing / [Compositing rendering phase](#)
- Pinch/Magnify brush / [The head](#)
- pin tool / [Editing the UV islands](#)
- Pitchipoy human rig
 - URL / [How it works...](#)
- playblast
 - about / [Rendering an OpenGL playblast of the animation](#)
- PlayBlast / [Render a quick preview of a shot](#)
- polygonal modeling
 - scene, preparing for / [Preparing the scene for polygonal modeling](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
- poly modeling / [Choosing sculpting over poly modeling](#)
- Pose Mode
 - about / [Getting ready](#)
- Pose to Pose principle / [Straight Ahead Action and Pose to Pose](#)
- Properties sidepanel / [Getting ready](#)
- Proportional Editing / [Modeling the chest](#)
- proxy
 - making / [Linking the character and making a proxy](#), [Getting ready](#), [How to do it...](#)
 - / [Organization](#)
- proxy object
 - about / [Linking the character and making a proxy](#)
 - URL / [Creating a simple walk cycle for the character by assigning keys to the bones](#)

Q

- Quick Edit tool
 - about / [The Quick Edit tool](#)
 - defining / [The Quick Edit tool](#), [How to do it...](#), [How it works...](#)

R

- Rat Cowboy
 - rigging / [Rigging the Rat Cowboy](#)
 - about / [Rigging the Rat Cowboy](#)
 - deforming bones, placing / [Placing the deforming bones](#)
 - Display options / [Placing the deforming bones](#)
 - leg, rigging / [The leg and the foot](#)
 - foot, rigging / [The leg and the foot](#)
 - arm, rigging / [The arm and the hand](#)
 - hand, rigging / [The arm and the hand](#)

- arm / [The arm and the hand](#)
 - hand / [The arm and the hand](#)
 - hips, rigging / [The hips](#)
 - tail, rigging / [The tail](#)
 - head, rigging / [The head and the eyes](#)
 - eyes, rigging / [The head and the eyes](#)
 - rig, mirroring / [Mirroring the rig](#)
 - gun, rigging / [Rigging the gun](#)
 - holster, rigging / [Rigging the holster](#)
 - root bone, adding / [Adding a root bone](#)
- raw rendering phase
 - about / [The Raw rendering phase](#)
- re-topologized mesh
 - concluding / [Concluding the re-topologized mesh, How to do it..., There's more...](#)
- re-topology
 - starting / [How to do it...](#)
- realistic grass
 - creating / [Creating realistic grass](#)
 - grass, generating with particles / [Generating the grass with particles](#)
 - grass shader, creating / [Creating the grass shader](#)
- render layers
 - compositing / [Compositing the render layers, Getting ready, How to do it..., How it works...](#)
 - URL / [How it works...](#)
- render passes
 - URL / [How it works...](#)
- render settings
 - URL / [See also](#)
- reptile skin shaders
 - building, in Cycles / [How to do it..., How it works..., There's more...](#)
 - building, in Blender Internal / [Building the reptile skin shaders in Blender Internal, Getting ready, How to do it..., How it works...](#)
- retopologized mesh / [The baking of textures](#)
- retopology
 - creating / [Why make a retopology?](#)
 - polygons arranging, possibilities / [Possibilities of arranging polygons](#)
 - creating, best practices / [Errors to avoid during the creation of retopology](#)
 - polygon density / [Density of polygons](#)
 - of alien character, creating / [Making the retopology of the alien character](#)
- retopology process / [Introduction](#)
- Return key / [Modeling the antenna](#)
- rig
 - creating / [How to do it...](#)
 - about / [How it works..., Getting ready](#)
- rigging process
 - defining / [Introduction](#)
 - URL / [Generating the character's Armature by using the Rigify add-on, How it works...](#)

- about / [An introduction to the rigging process](#)
- rigging stage
 - about / [Introduction](#)
- Rigify add-on
 - about / [Introduction](#)
- robot toy
 - modeling / [Let's start the modeling of our robot toy](#)
 - about / [Let's start the modeling of our robot toy](#)
 - image reference, adding for preparing workflow / [Preparing the workflow by adding an image reference](#)
 - naming shortcuts / [Adding the head primitive](#)
 - image rHead primitive, adding / [Adding the head primitive](#)
 - Edit Mode, versus Object Mode / [The Edit Mode versus the Object Mode](#)
 - head, modeling / [Modeling the head](#)
 - antenna, modeling / [Modeling the antenna](#)
 - thunderbolts, modeling / [Modeling the thunderbolts](#)
 - eyes, modeling / [Modeling the eyes](#)
 - chest, modeling / [Modeling the chest](#)
 - neck, modeling / [Modeling the neck](#)
 - torso, modeling / [Modeling the torso](#)
 - buttons, modeling / [Modeling the buttons](#)
 - fork, modeling / [Modeling the fork](#), [Modeling protections for the fork](#)
 - main wheel, modeling / [Modeling the main wheel](#)
 - arm, modeling / [Modeling the arm](#)
 - rendering, with Blender Internal / [Using Blender Internal to render our Robot Toy](#)
- root bone
 - about / [Adding a root bone](#)

S

- sampling, Cycles render engine
 - Render samples setting / [The sampling](#)
 - Preview samples setting / [The sampling](#)
- scale operator
 - using / [How to do it...](#)
- scaling
 - about / [How to do it...](#)
- scene
 - organization / [Organizing the scene](#)
- scene animation
 - about / [Animating the scene](#)
 - walk cycle / [The walk cycle](#)
 - actions, mixing / [Mixing actions](#)
 - close shot animation / [Animation of a close shot](#)
 - gunshot / [Animation of the gunshot](#)
 - gunshot animation / [Animation of the gunshot](#)
 - trap animation / [Animation of the trap](#)
- scene organization

- objects, grouping / [Grouping objects](#)
 - layers, working with / [Working with layers](#)
- scenes
 - URL / [How it works...](#)
- sculpting
 - base mesh, preparing for / [Preparing the base mesh for sculpting](#), [How to do it...](#), [How it works...](#)
 - selecting, over poly modeling / [Choosing sculpting over poly modeling](#)
 - pen tablet, using / [Using a pen tablet](#)
 - sculpt mode / [The sculpt mode](#)
- sculpting process
 - about / [Understanding the sculpting process](#), [An introduction to sculpting](#)
- sculpt mode
 - about / [The sculpt mode](#)
 - viewport, optimizing / [Optimizing the viewport](#)
 - Undo function / [The head](#)
- Secondary Action principle / [Secondary Action](#)
- shader
 - creating, for metal plates / [How to do it...](#)
- shape key
 - types / [How it works...](#)
- shape keys
 - creating / [Creating shape keys](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - drivers, assigning to / [Assigning drivers to the shape keys](#), [Getting ready](#), [How to do it...](#), [There's more...](#)
 - about / [The shape keys](#)
 - controlling / [What is a shape key?](#)
 - basic shapes, creating / [Creating basic shapes](#)
 - driving / [Driving a shape key](#)
- shot
 - quick previews, rendering / [Render a quick preview of a shot](#)
- Shrinkwrap modifier
 - used, for re-topologizing mesh / [Using the Shrinkwrap modifier to re-topologize the mesh](#), [How to do it...](#)
 - tweaking / [There's more...](#)
- Shrinkwrap modifier technique
 - setting up / [Getting ready](#)
 - using / [How it works...](#)
- sIBL addon
 - about / [See also](#)
 - URL / [See also](#)
 - references / [See also](#)
- sIBL Archive
 - URL / [How to do it...](#)
- sIBL archive
 - URL / [See also](#)
- simple walk cycle, character

- creating, by assigning keys to bones / [Creating a simple walk cycle for the character by assigning keys to the bones](#), [Getting ready](#), [How to do it...](#), [There's more...](#), [See also](#)
- skin, Gidiosaurus
 - defining / [Introduction](#)
- Skin modifier
 - about / [Introduction](#), [Introduction](#)
 - base mesh, building with / [Building the character's base mesh with the Skin modifier](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - using / [Building the character's base mesh with the Skin modifier](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - used, for creating base mesh / [Creating a base mesh with the Skin modifier](#)
- skinning
 - performing / [Introduction](#)
 - about / [Skinning](#)
 - Weight Paint tools / [The Weight Paint tools](#)
 - weight, manually assigning to vertices / [Manually assigning weight to vertices](#)
 - foot deformation, correcting / [Correcting the foot deformation](#)
 - belt deformation, correcting / [Correcting the belt deformation](#)
- skinning to shapes
 - URL / [There's more...](#)
- skin node group
 - creating / [How to do it...](#), [How it works...](#)
- Slow In effect / [Slow In and Slow Out](#)
- Slow Out effect / [Slow In and Slow Out](#)
- Smart UV Project tool
 - using / [Using the Smart UV Project tool](#), [Getting ready](#), [How to do it...](#)
- Smear brush / [The Smear brush](#)
- Smooth brush / [The head](#)
- Snake Hook brush / [The head](#)
- Snap tool
 - used, for re-topologizing mesh / [Using the Snap tool to re-topologize the mesh](#), [How to do it...](#), [How it works...](#)
- Soften brush / [The Soften brush](#)
- solid drawing principle / [Solid drawing](#)
- Squash and Stretch principle / [Squash and Stretch](#)
- stack simulation of wooden planks
 - adding / [Simulate a stack of wooden planks with physics](#)
 - creating / [Creation of the simulation of a stack of planks](#)
- staging principle / [Staging](#)
- Straight Ahead Action principle / [Straight Ahead Action and Pose to Pose](#)
- Stroke option
 - about / [The Stroke option](#)
 - space method / [The Stroke option](#)
 - curve method / [The Stroke option](#)
 - line method / [The Stroke option](#)
 - anchored method / [The Stroke option](#)
 - airbrush method / [The Stroke option](#)

- Drag Dots method / [The Stroke option](#)
- dots method / [The Stroke option](#)
- Subdivision Surface modifier / [How to do it...](#)
 - about / [How to do it...](#), [An introduction to the Subdivision Surface modifier](#)
 - head shape, improving / [Improving the head shape](#)
 - work, saving / [Improving the head shape](#)

T

- templates
 - setting, with Images as Planes add-on / [Setting templates with the Images as Planes add-on](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - setting, with Image Empties method / [Setting templates with the Image Empties method](#), [How to do it...](#), [How it works...](#)
 - setting, with Background Images tool / [Setting templates with the Background Images tool](#), [How to do it...](#)
- TexDraw brush / [The TexDraw brush](#)
- texture node
 - viewing, through Viewer / [Introduction to nodal compositing](#)
- Texture Paint, Blender
 - references / [There's more...](#)
- Texture Paint tool
 - about / [Basics of the Texture Paint tool](#)
 - brushes, discovering / [Discovering the brushes](#)
 - Stroke option / [The Stroke option](#)
 - zones of painting, delimiting / [Delimiting the zones of painting according to the geometry](#)
 - direct painting / [Painting directly on the texture](#)
- textures
 - creating, for Gidiosaurus character / [Introduction](#)
 - backing, in Cycles render engine / [Baking textures in Cycles](#)
 - tree, backing / [Baking the tree](#)
- textures, Blender Render Engine
 - URL / [Building the eyes' shaders in Blender Internal](#)
- textures, UV's
 - baking / [The baking of textures](#)
 - normal map, baking / [The baking of a normal map](#)
 - normal map / [What is a normal map?](#)
 - size, selecting / [Making of the bake](#)
- three-point lighting rig
 - setting, in Blender Internal / [Setting a three-point lighting rig in Blender Internal](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - URL / [How it works...](#)
- tileable scales image
 - creating, in Blender Internal / [Making a tileable scales image in Blender Internal](#), [Getting ready](#), [How to do it...](#)
- tileable scales texture

- baking, into UV tiles / [Baking the tileable scales texture into the UV tiles](#), [Getting ready](#), [How to do it...](#), [There's more...](#)
- tiled textures
 - about / [Tiled textures](#)
 - workspace setting / [The settings of our workspace](#)
 - considerations / [Advice for a good tiled texture](#)
 - baking / [Baking our tiled textures](#)
 - baking, need for / [Why bake?](#)
 - baking, steps / [How to do it?](#)
- timeline editor / [The timeline](#)
- timing principle / [Timing](#)
- tools, re-topology
 - Gstretch / [How to do it...](#)
 - Loft / [How to do it...](#)
 - Bridge / [How to do it...](#)
- Tool Shelf
 - about / [How to do it...](#)
 - / [How to do it...](#)
- tracking shot / [Writing a short script](#)
- Transformation manipulators
 - about / [How to do it...](#), [How to do it...](#)
- Transform locks
 - defining / [How to do it...](#)
- transparent textures
 - creating / [Creating transparent textures](#)
- transparent textures, creating
 - grass texture / [The grass texture](#)
 - color wheel window / [The grass texture](#)
 - grunge texture / [The grunge texture](#)

U

- U-Dimension / [There's more...](#)
 - about / [Preparing the model to use the UDIM UV tiles](#)
- Ubuntu
 - about / [There's more...](#)
- UDIM UV Mapping / [There's more...](#)
 - about / [Preparing the model to use the UDIM UV tiles](#)
- UDIM UV Mapping standard / [There's more...](#)
- UDIM UV tiles
 - used, for preparing model / [Preparing the model to use the UDIM UV tiles](#), [How to do it...](#), [How it works...](#)
- UV islands
 - editing / [Editing the UV islands](#), [How to do it...](#)
 - modifying / [Modifying the mesh and the UV islands](#), [Getting ready](#), [How to do it...](#)
- UV Map layout
 - exporting / [Exporting the UV Map layout](#), [How to do it...](#)
- UV Mapping / [Introduction](#)

- UVs
 - unwrapping / [Unwrapping UVs](#)
 - tiling / [Tiling UVs](#)
 - tiling, goal / [What is tiling for?](#)
 - layers / [The UV layers](#)
 - layers options / [The UV layers](#)
- UV Sphere placeholders
 - about / [Modeling the eye](#)
- UV Spheres
 - about / [Building the eyes' shaders in Cycles](#)
- UV tiles
 - tileable scales texture, baking into / [Baking the tileable scales texture into the UV tiles, Getting ready, How to do it..., There's more...](#)
- UV unwrapping process / [Understanding UVs](#)
 - about / [Unwrapping UVs](#)
 - Project From View, using / [Using Project From View](#)
 - remaining house components / [Unwrapping the rest of the house](#)
 - tree, using with Smart UV project / [The tree with the Smart UV Project](#)
 - environment objects, unwrapping / [Unwrapping the rest of the environment](#)
- UV's
 - unwrapping / [Unwrapping UVs](#)
 - about / [Understanding UVs](#)
 - seam placement / [The placement of the seams](#)
 - island placement / [The placement and adjustment of the islands](#)
 - island adjustment / [The placement and adjustment of the islands](#)
 - textures, baking / [The baking of textures](#)

V

- Vertex Colors layer
 - adding / [Adding a dirty Vertex Colors layer and baking it to an image texture, How to do it...](#)
 - baking, to image texture / [Adding a dirty Vertex Colors layer and baking it to an image texture, How to do it...](#)
 - references / [How it works...](#)
- Vertex Connect Path tool / [Modeling the arm](#)
- vertex group
 - creating / [How to do it...](#)
- Vertex Groups
 - URL / [How it works...](#)
- VSE
 - used, for editing sequence / [Editing the sequence with the VSE](#)
 - about / [Introduction to the Video Sequence Editor](#)
 - final sequence, editing / [Edit and render the final sequence](#)
 - final sequence, rendering / [Edit and render the final sequence](#)

W

- walk cycle tutorial
 - URL / [See also](#)
- Weight Groups
 - assigning, by hand / [Assigning Weight Groups by hand](#), [Getting ready](#), [How to do it...](#), [How it works...](#)
 - editing, Weight Paint tool used / [Editing Weight Groups using the Weight Paint tool](#), [How to do it...](#)
- Weight Painting stage / [An introduction to the rigging process](#)
- Weight Paint mode / [How to do it...](#)
- Weight Paint tool
 - used, for editing Weight Groups / [Editing Weight Groups using the Weight Paint tool](#), [How to do it...](#)
 - URL / [How to do it...](#)
- Weight Paint tools
 - about / [The Weight Paint tools](#)
 - URL / [The Weight Paint tools](#)
 - skinning, editing / [Correcting the foot deformation](#)
- World, in Blender
 - references / [See also](#)