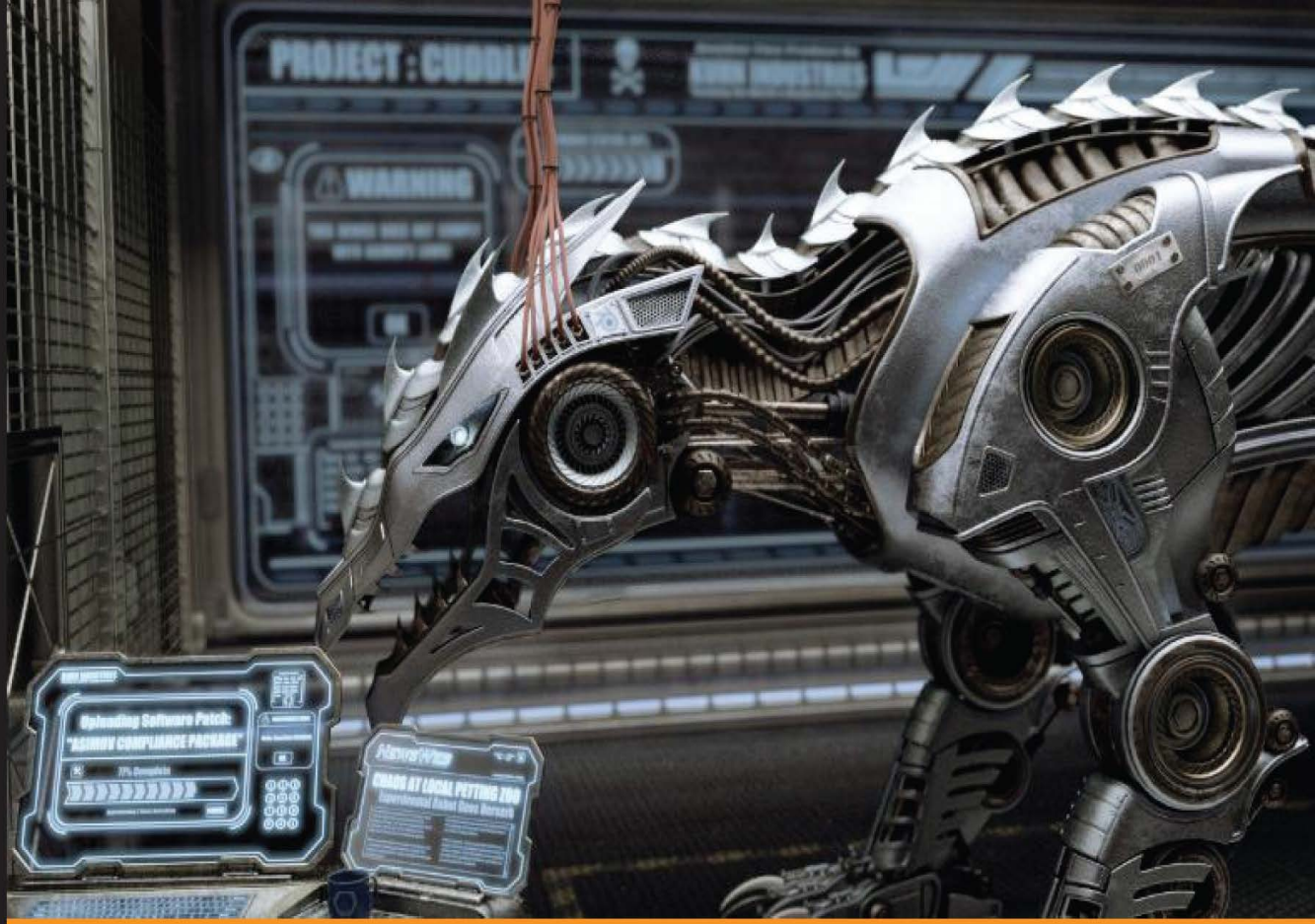


Community Experience Distilled

Blender 3D Incredible Machines

Design, model, and texture complex mechanical objects in Blender

Christopher Kuhn

[PACKT] PUBLISHING open source* community experience distilled

Blender 3D Incredible Machines

Design, model, and texture complex mechanical objects in Blender

Christopher Kuhn



BIRMINGHAM - MUMBAI

Blender 3D Incredible Machines

Copyright © 2016 Packt Publishing

All rights reserved. No part of this book 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 book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, 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 book.

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

Publishing Month: February 2016

Production reference: 1180216

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.
ISBN 978-1-78528-201-0

www.packtpub.com

Credits

Author

Christopher Kuhn

Reviewer

Jacek Herman

Commissioning Editor

Priya Singh

Acquisition Editor

Divya Poojari

Content Development Editor

Trusha Shriyan

Technical Editor

Murtaza Tinwala

Copy Editor

Charlotte Carneiro

Project Coordinator

Kinjal Bari

Proofreader

Safis Editing

Indexer

Rekha Nair

Graphics

Jason Monteiro

Production Coordinator

Melwyn Dsa

About the Author

Christopher Kuhn is a 3D artist and Blender enthusiast. He has been heavily involved in the Blender community since 2010. His company, Kuhn Industries LLC, creates custom 3D assets and educational materials for both professional and non professional uses. In addition to his 3D courses on `CGCookie.com`, he's written two previous books on Blender (*Build Your Own Rocket Bike* and *Death to the Armatures*).

About the Reviewer

Jacek Herman is a Master of Fine Arts, 3D generalist and Blender enthusiast. He spends most of his time locked in a basement, building game assets from scratch or designing 3D printable junk. He specializes in tedious craft of UV texturing ridiculously large models.

www.PacktPub.com

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface 1

Chapter 1: Sci-Fi Pistol - Creating the Basic Shapes 5

- **A project overview** 6
- **Creating the barrel** 7
  - Modeling a handgrip and other pieces 25
- **Summary** 32

Chapter 2: Sci-Fi Pistol - Adding Details 33

- **Finishing the handgrip** 33
- **Cutting shapes into our gun** 47
- **Circles, angles, and edge splits** 66
- **Adding final details** 72
- **General workflow for detailing** 76
- **Summary** 87

Chapter 3: Texturing and Rendering Your Sci-Fi Pistol 88

- **Preparing our scene** 89
- **Enabling GPU rendering** 89
- **Adding materials and textures** 99
- **Summary** 117

Chapter 4: Spacecraft – Creating the Basic Shapes 118

- **Shaping the body** 118
- **Creating the accessories** 132
- **Summary** 136

Chapter 5: Spacecraft - Adding Details 137

- **Refining our ship** 137
- **Adding detail with pipes** 166
- **Finishing our main objects** 170
- **Do it yourself – completing the body** 183
- **Building the landing gear** 185
- **Adding the cockpit** 197
- **Creating small details** 202
- **Working with Path objects** 209
- **Finishing touches** 217

Summary 221
Chapter 6: Spacecraft – Materials, Textures, and Rendering 222
Preparing the model and scene 223
Creating materials 228
Adding decals with UV maps 242
Other possibilities 247
Summary 249
Chapter 7: Modeling Your Freestyle Robot 250
Modeling for Freestyle 250
Blocking out your robot 261
Creating the body with Subdivision Surfacing 263
Finishing the robot 270
Summary 276
Chapter 8: Robot - Freestyle Rendering 277
Preparing the scene 277
Marking Freestyle edges 283
Creating materials 284
Rendering and material options 292
Summary 300
Chapter 9: Low-Poly Racer – Building the Mesh 301
Building for external applications 301
Starting the truck model 304
Manifold versus non-manifold meshes 312
Adding details 315
Summary 324
Chapter 10: Low-Poly Racer – Materials and Textures 325
Texturing workflow 325
Adding materials 326
Unwrapping and baking 337
Adding detail in an Image Editor 365
Checking the Texture Map 367
Summary 372
Index 373

Preface

Welcome to *Blender 3D Incredible Machines*. In this book, we're going to explore the world of hard-surface modeling in Blender. This is distinct from organic modeling (creating humans/animals, and more), and there are different methods that we can employ to get the best results. We'll focus on ways to create complex machinery, vehicles, and other similar models in Blender.

What this book covers

`Chapter 1`, *Sci-Fi Pistol – Creating the Basic Shapes*, covers the basic modeling tools as we start our first project. It should be a good refresher for more experienced users, and will set the groundwork for more advanced modeling projects.

`Chapter 2`, *Sci-Fi Pistol – Adding Details*, looks at specific modeling techniques for adding detail to hard surface models. These techniques will be used to finish building our gun model, but will be applicable to many other projects as well.

`Chapter 3`, *Texturing and Rendering Your Sci-Fi Pistol*, takes a look at the Cycles rendering engine and how to create basic materials for it. We'll also set up a basic render scene that we can use in the future.

`Chapter 4`, *Spacecraft – Creating the Basic Shapes*, begins a more complex modeling project—a Sci-Fi spacecraft. We'll focus more on general technique and workflow here, since we covered a lot of the basic tools already in the first two chapters.

`Chapter 5`, *Spacecraft – Adding Details*, focuses heavily on detail-oriented modeling techniques. We'll look at important considerations when adding detail to a model as well as a number of tools and procedures for doing so.

`Chapter 6`, *Spacecraft – Materials, Textures, and Rendering*, builds on the techniques from our last project. We'll create a number of different materials for our spacecraft. We'll also briefly cover UV mapping and managing materials slots.

`Chapter 7`, *Modeling Your Freestyle Robot*, looks at modeling for a specific type of rendering—FreeStyle. There are a number of special modeling techniques we'll want to use to get the best possible results.

`Chapter 8`, *Robot – Freestyle Rendering*, looks at the specifics of FreeStyle rendering. We'll also cover the creation on non-photorealistic (NPR) rendering in the Blender Internal render engine.

`Chapter 9`, *Low-Poly Racer – Building the Mesh*, takes a look at one of the more common uses of Blender—building game models. We'll explore how these models are different from the ones we've already done and what specific techniques we should use (and avoid) when we do it.

`Chapter 10`, *Low-Poly Racer – Materials and Textures*, covers the creation of materials and textures for our low-poly game model. We'll cover UV maps more extensively, and see how we can use them to create a more universal form of textures. This will allow our game assets to be used in a variety of external applications.

What you need for this book

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 book is for

This book is aimed at intermediate Blender users looking to increase their hard surface modeling skills. Although we do review some basic tools in the first chapter, this isn't meant to be an introduction to the Blender software package. Users should be reasonably familiar with the interface and fundamental concepts of 3D modeling (moving, rotating, scaling, and others).

Conventions

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

New terms and important words are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "This will allow us to add the **Mirror** modifier."



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or 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 book'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 book, we have a number of things to help you to get the most from your purchase.

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from: `http://www.packtpub.com/sites/default/files/downloads/blender3dincrediblemachines_ColorImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books-maybe a mistake in the text or the code-we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any

errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded 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 book 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 book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

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

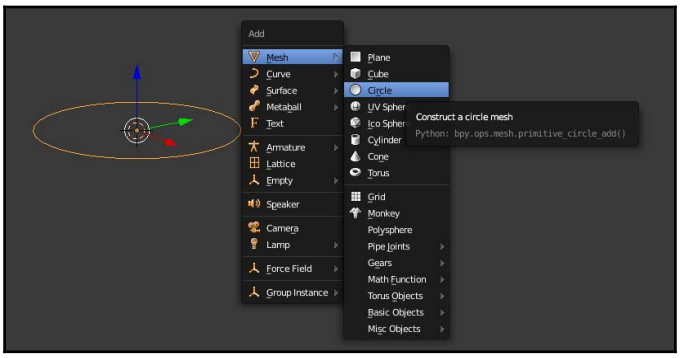
When we're finished with this section, our project will look like this:



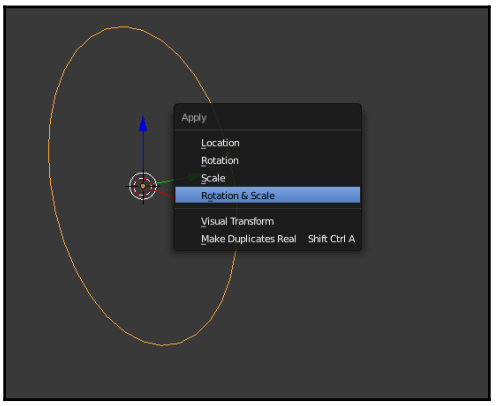
If you're ready, let's open a new scene in Blender and get started!

Creating the barrel

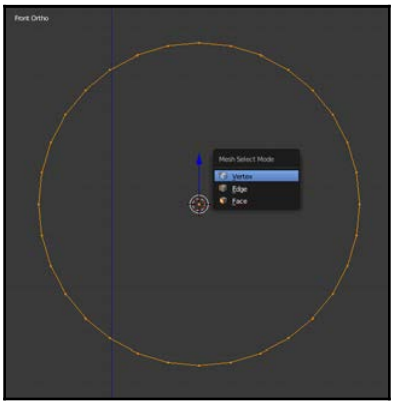
We'll start by adding a circle to our scene (press *Shift* + *A* to access the **Add** menu). By default, circles are added with 32 sides (and 32 vertices). This screenshot will work just fine for us:



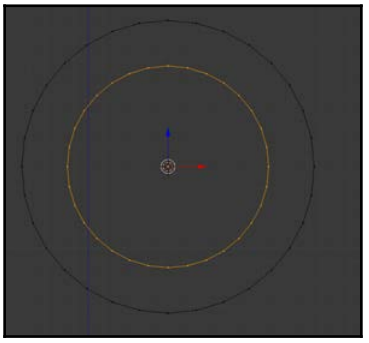
Next, we'll rotate the circle by 90° on the X axis (*R*, 90, *X*, and *Enter*). Then, we'll apply our rotation and scale by pressing *Ctrl* + *A* and selecting **Rotation & Scale**. We want to apply the rotation here for Blender to be aware of the object's default orientation. This is relevant to several 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:



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*:

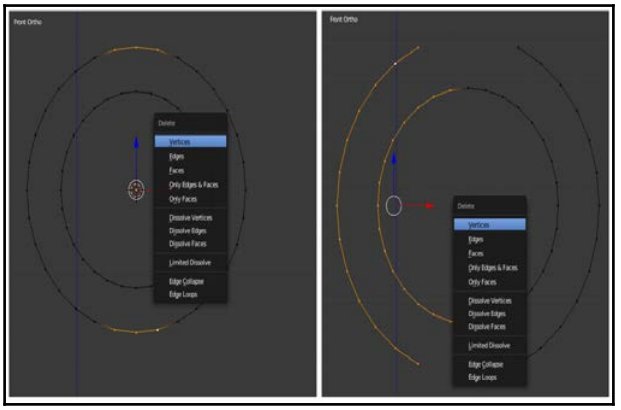


Next, we'll use *Shift* + *D* to duplicate them. We'll scale the new circle down using the *S* key and drag the cursor to the middle of the circle. This smaller circle will form the inside of the barrel and allow us to make changes to the outer circle without losing track of the center (and vertices) of the original one.

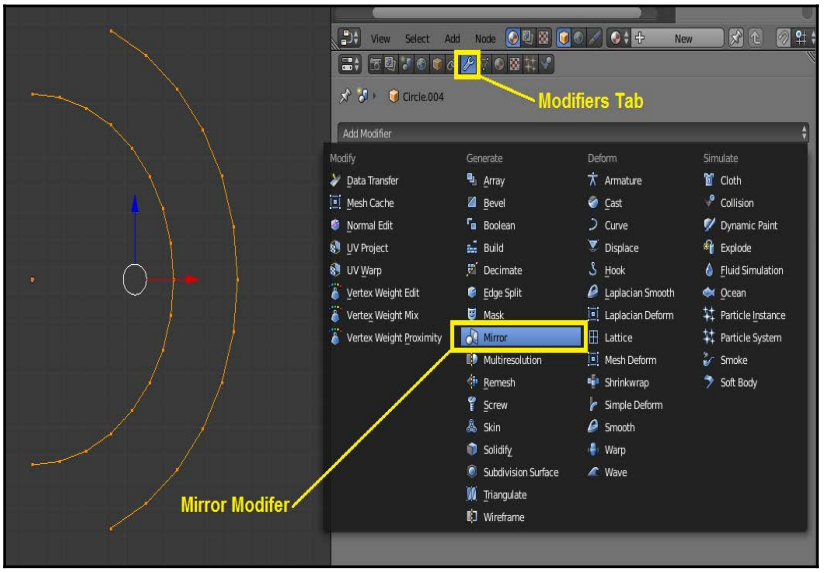


Next, we'll start deleting some vertices that we don't want. The outside of the barrel will not be a perfect circle, so we'll need to change it. First, we'll delete three vertices at both the top and bottom of the circle. To do this, we'll select the ones that we don't want any longer, and then hit *X*. 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.

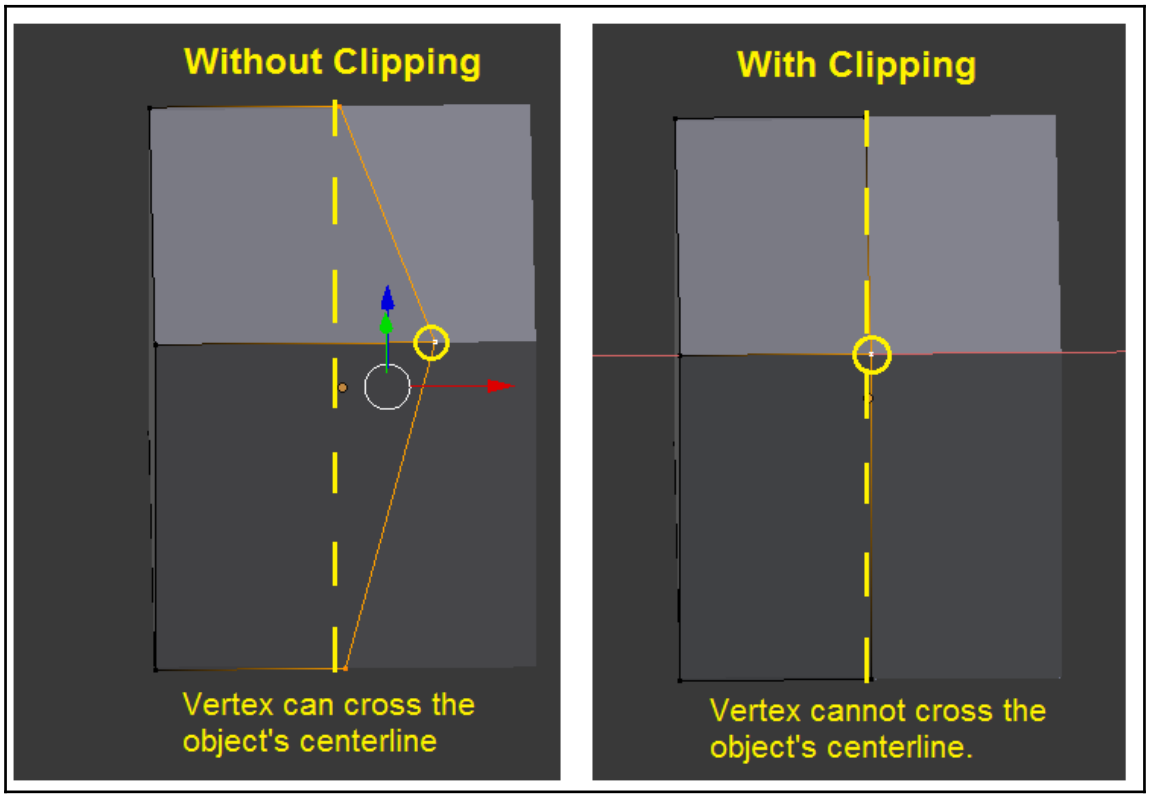
Next, we'll delete half of both the circles:



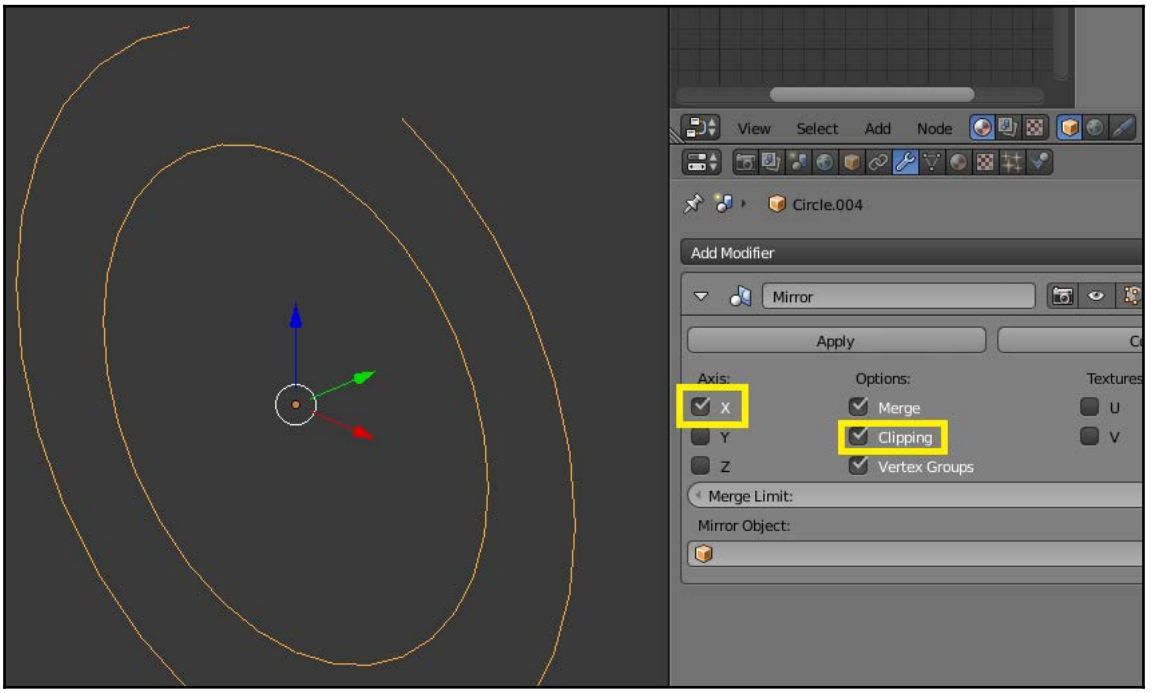
This will allow us to add the Mirror modifier. This modifier will duplicate half of an object across its axis. Using it, we can model one side of the gun, and Blender will automatically fill in the other side to match it. Obviously, this will save us a lot of time. In order to do this, go to the **Modifiers** tab of your properties panel and select **Mirror**.



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:



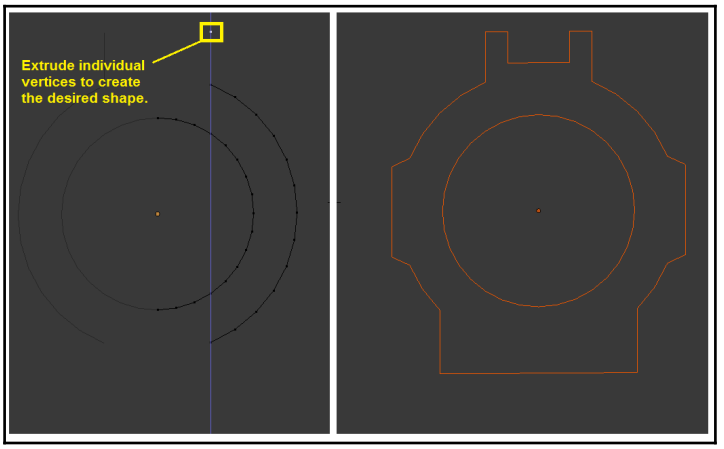
So, let's go ahead and select these options:



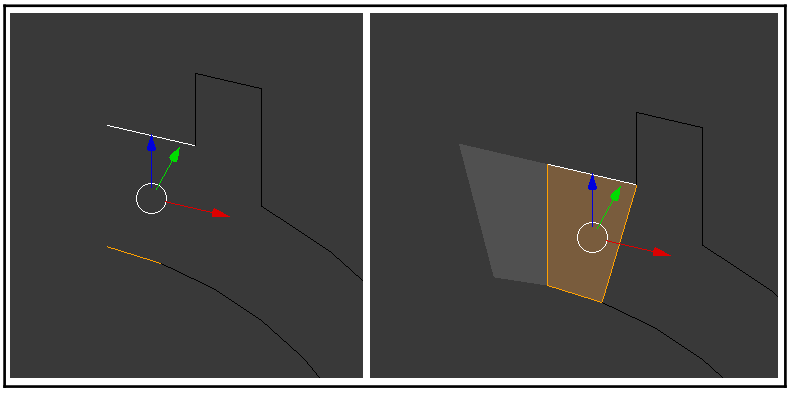
Now that we've done this, we'll be able to work on just one side of the object. So, let's start making changes to the mesh. I'm going to grab the innermost vertex at the top of our circle and extrude it upon the *Z* axis by pressing *E* and *Z* and hitting *Enter* when we've finished extruding.

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.

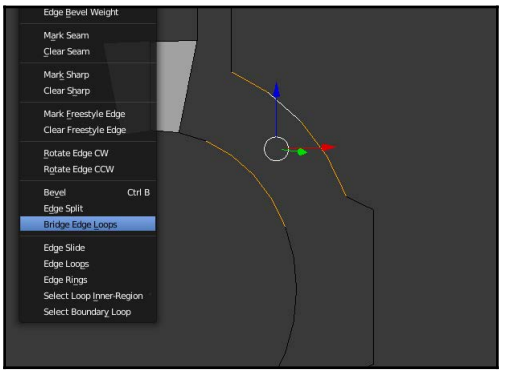
Obviously, there's a lot of room to personalize the model at this point—you don't need to follow these pictures exactly:



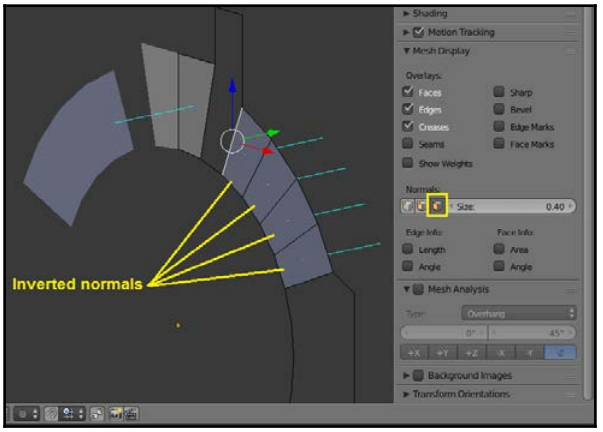
Once you have the cross-section that you're looking for, it's time to fill in some faces. It's not strictly necessary to do this right now (we'll actually end up deleting these), but it will enable us to see how Blender handles pre-existing faces during mesh editing. So, let's pick two edges at the top 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 *Ctrl* + and **Bridge Edge Loops**.

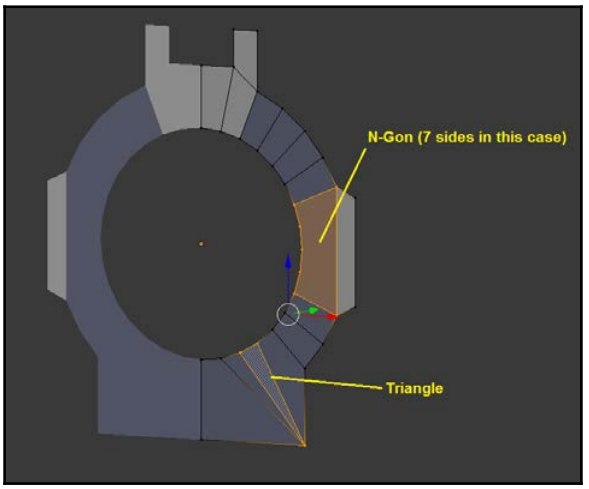


As soon as we do this, there's a good chance that you'll have a problem. Some of the faces will appear darker than others, which indicates that the **normals** are inverted. In Blender (and other 3D applications), a normal is the vector perpendicular to your face or the direction that your face is "pointing" 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:



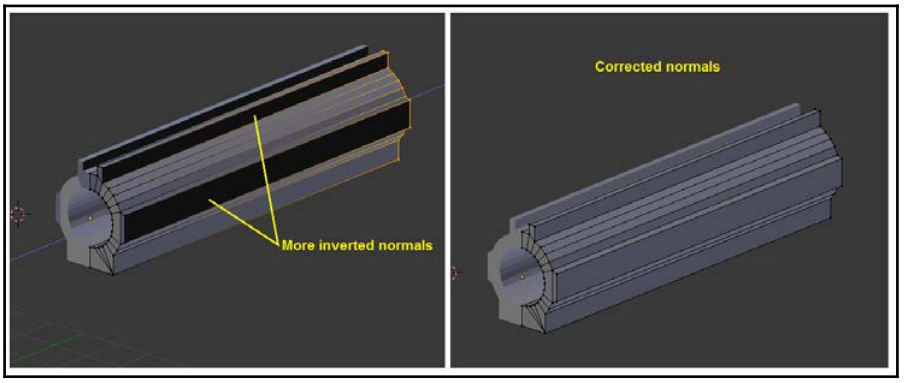
Blender does have a tool for automatically correcting normals. When you select all the faces with the A key and press Ctrl + N, Blender attempts to calculate the proper direction of the faces and flips the incorrect ones. Unfortunately, this doesn't work very well with 2D shapes. Since we just have a 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.

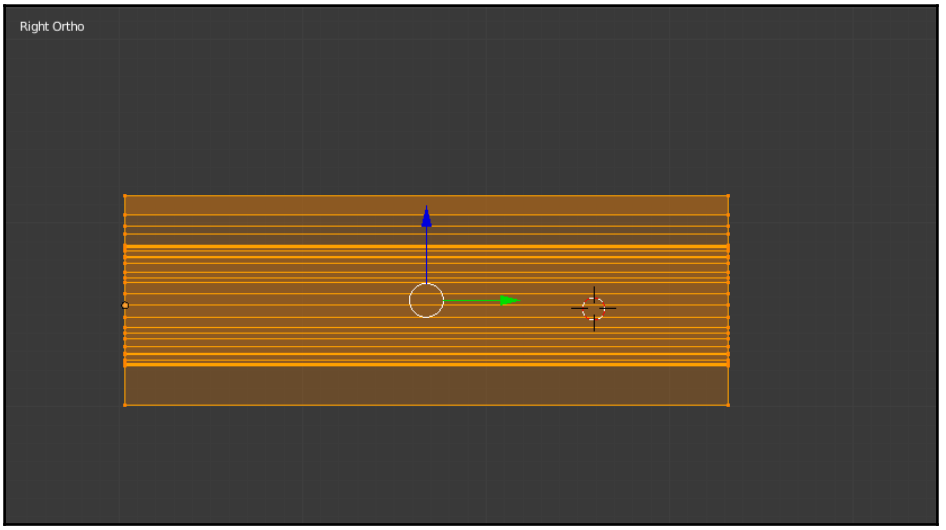


Once all your faces are filled in, you can select all of them with *A* key and extrude them with the *E* key. By default, the faces will extrude along the *Y* axis, which is exactly where we want them to go. In some cases, however, the faces will extrude along an axis that you don't want them to. To 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.

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.

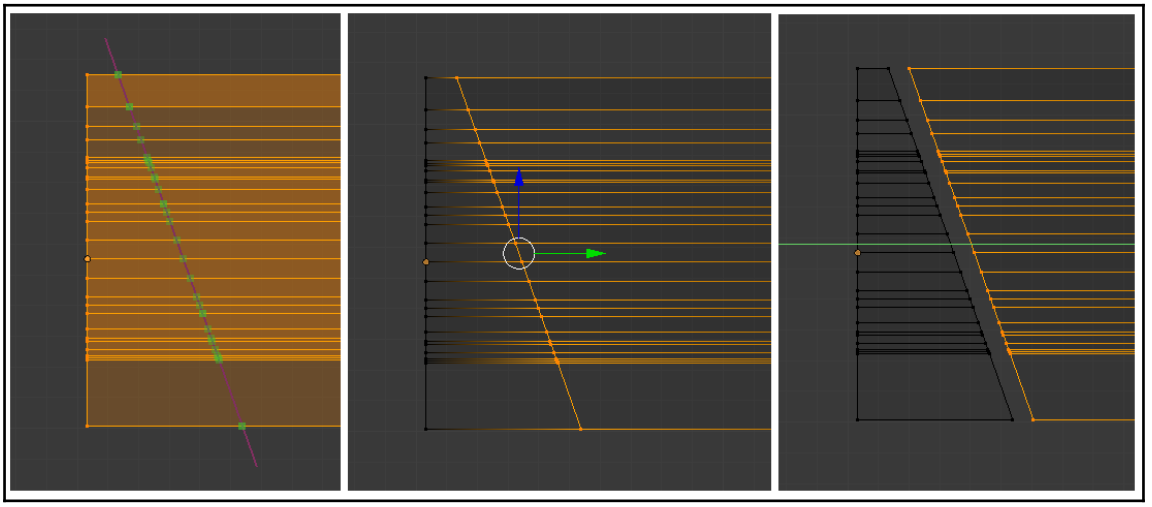


Now, let's move to our side view (numpad *3*).

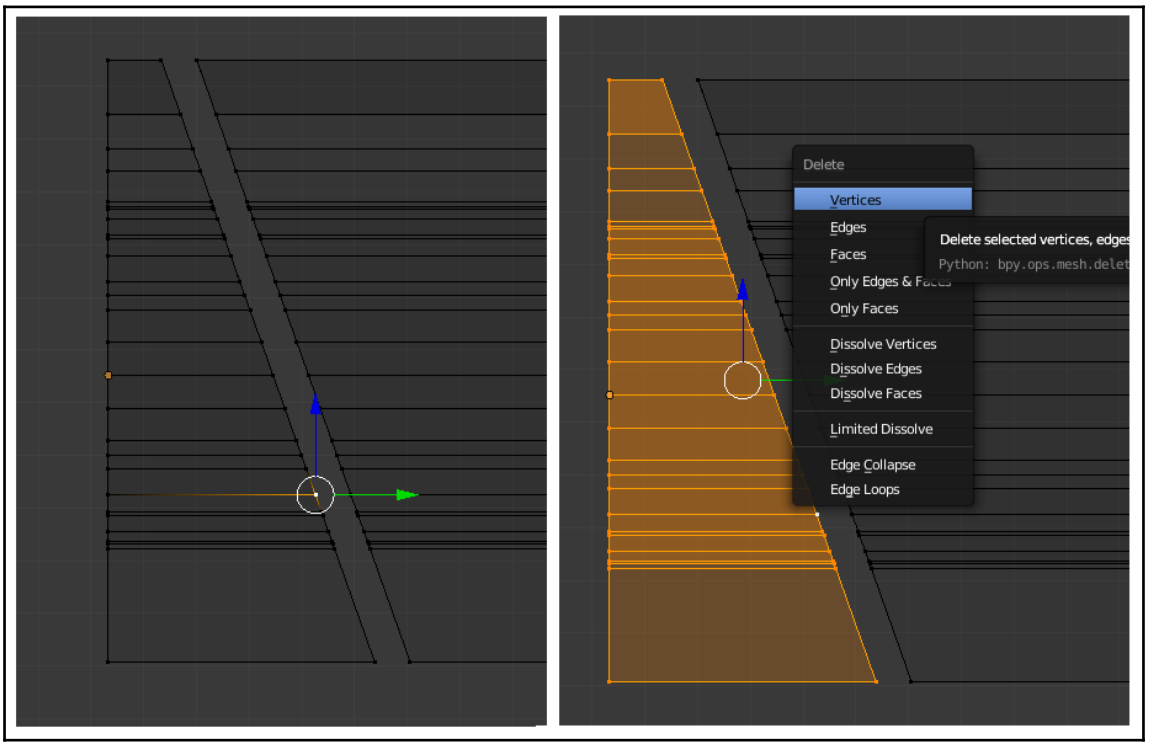


The front of our barrel will be angled, so we'll use the knife tool to cut across our existing mesh to create this slant. To activate the knife tool, press *Shift* + *K* (or *K* then *Z* to cut through the entire mesh). Move the knife at the top-left corner of the barrel, and anchor it with the left 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.

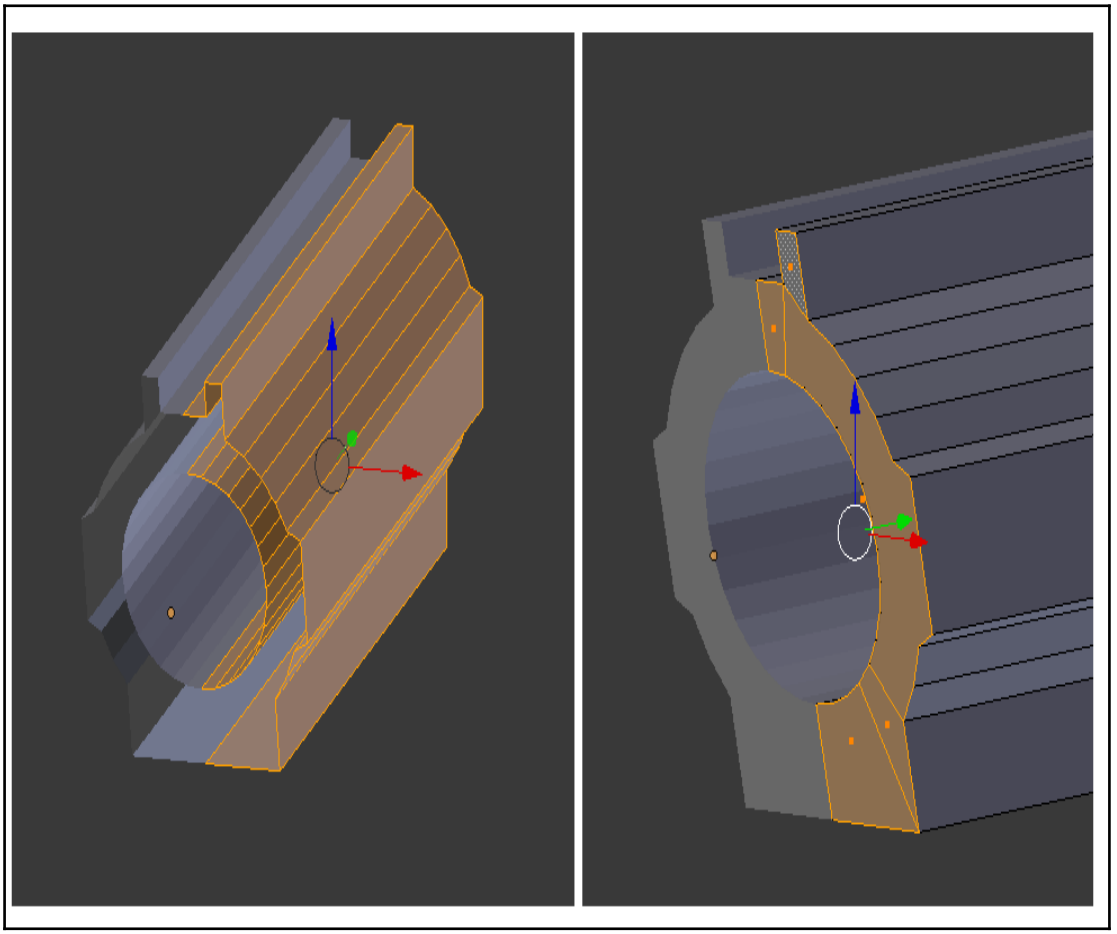
However, we're not done yet. With our new edges still selected, press the *V* key to rip them apart. This creates duplicate vertices that sit at the top of each other, "ripping" the mesh apart. You can move one set of vertices away from the other on the *Y* axis if you'd like; it makes things easier to see. You can also hit *Esc* immediately after ripping it apart.



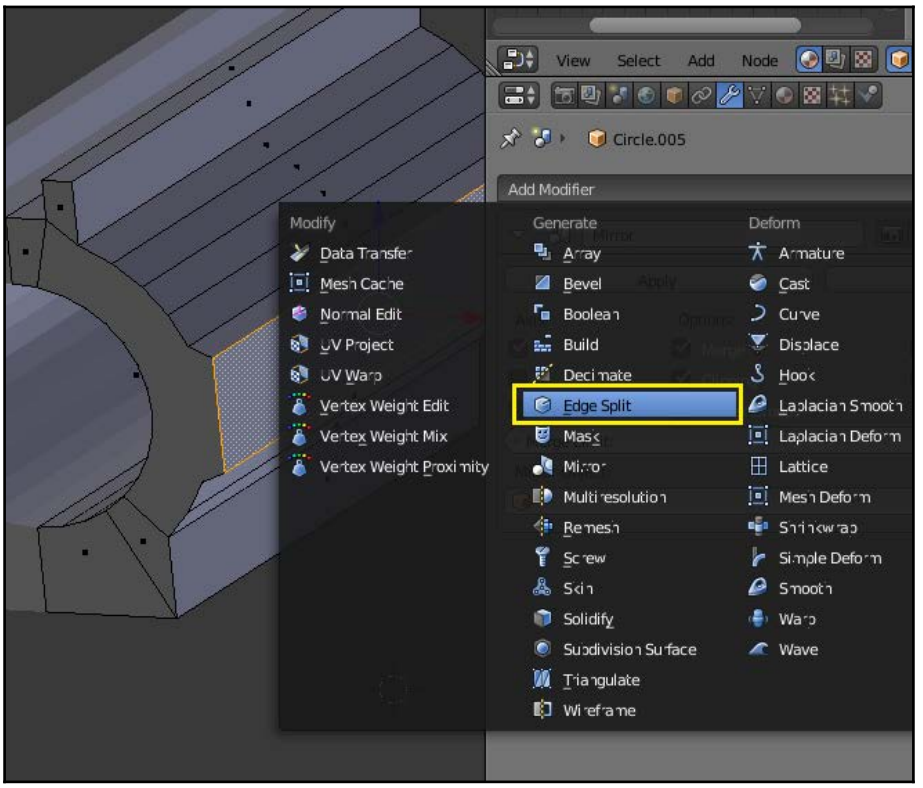
Next, we'll select one vertex in our front section of the barrel (the piece we don't want). By pressing *Ctrl + L*, Blender will automatically select all the linked (connected) vertices. You can then delete them, leaving just the section that you want.



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.

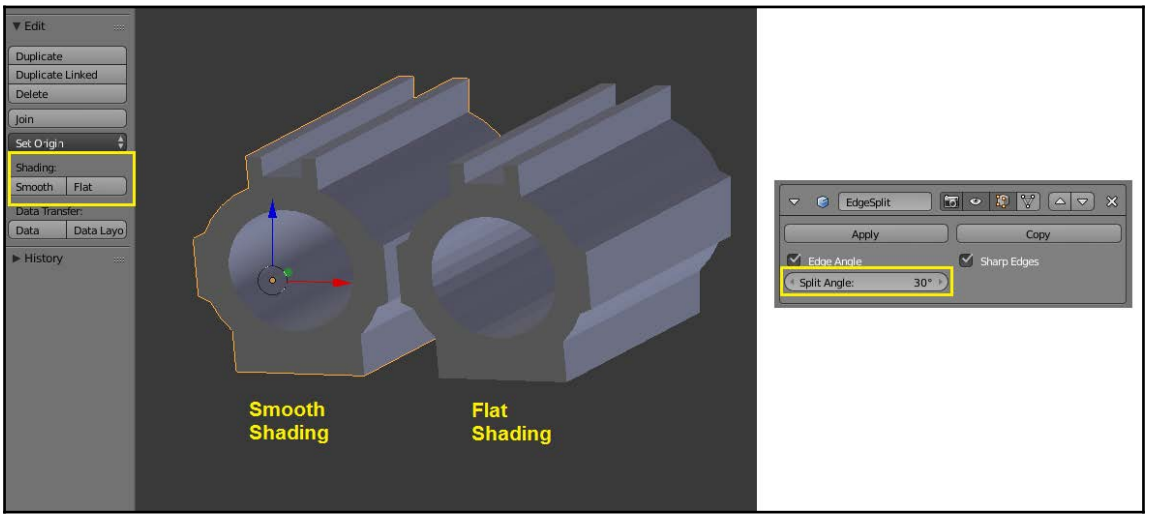


First, we'll add an Edge Split modifier to the gun. Most mechanical objects contain both smooth and sharp edges. Using the **E**dge Split modifier, we can tell Blender which angles should be sharp and which should be smooth. It's an incredibly useful tool for creating machines in Blender. To add it, just go back to your **Modifier** and select **Edge Split**.



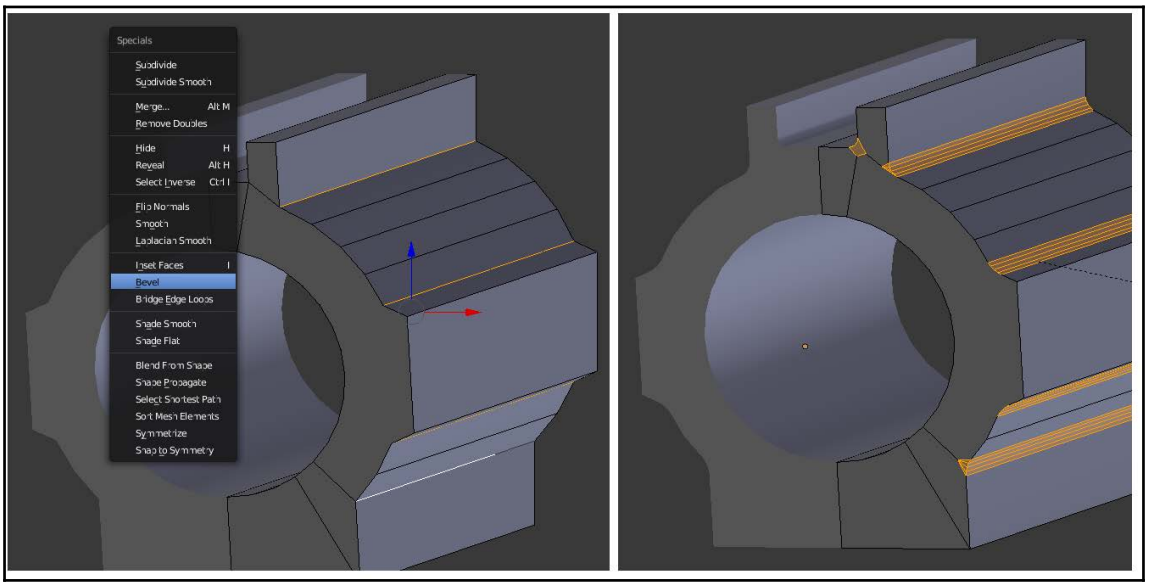
Once you add the modifier, go ahead and hit **Smooth** on the toolbar at the left-hand side of the screen. This will tell Blender that the entire object should use smooth shading except where the Edge Split modifier tells it not to. In this case, any angle that is greater than (or equal to) 30° will 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.

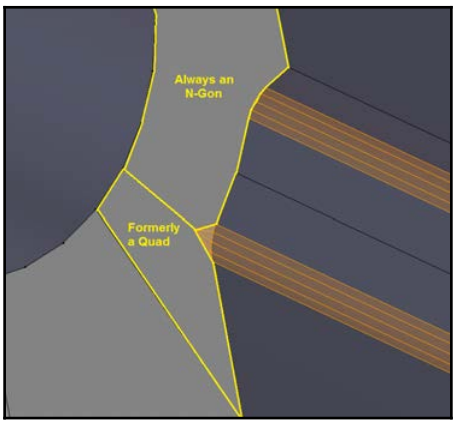


Now that we've got this taken care of, we'll smooth out our gun barrel a little bit. To do this, we'll use the **Bevel** tool. First, select the edges that you'd like to smooth out, then press *W* and select **Bevel** to activate the tool. Alternatively, you can activate the tool by pressing *Ctrl + B*.

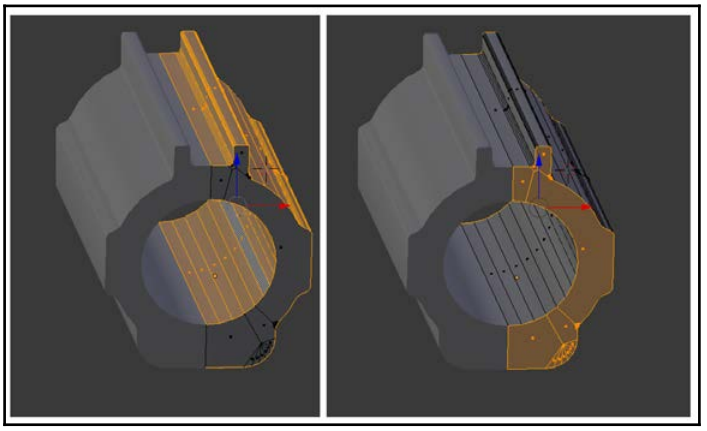
With the tool activated, you can drag the mouse across the screen to change the amount of beveling on an edge. By rolling the mouse wheel, you will increase the number of **segments** to the bevel. If you want a sharp edge (like the corner of a piece of metal) it's best to just use one segment. If you're 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.



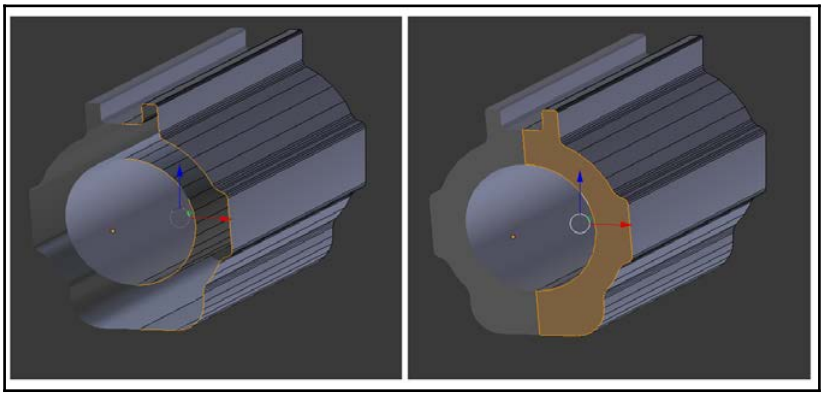
As we look at the front of the gun, we notice an interesting problem. Actually, it's not a problem; it's just the way that Blender does beveling. This is why we've filled in these faces at the front of the barrel, so we could observe the effects of the bevel tool on different types of faces. Here, you 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:



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.

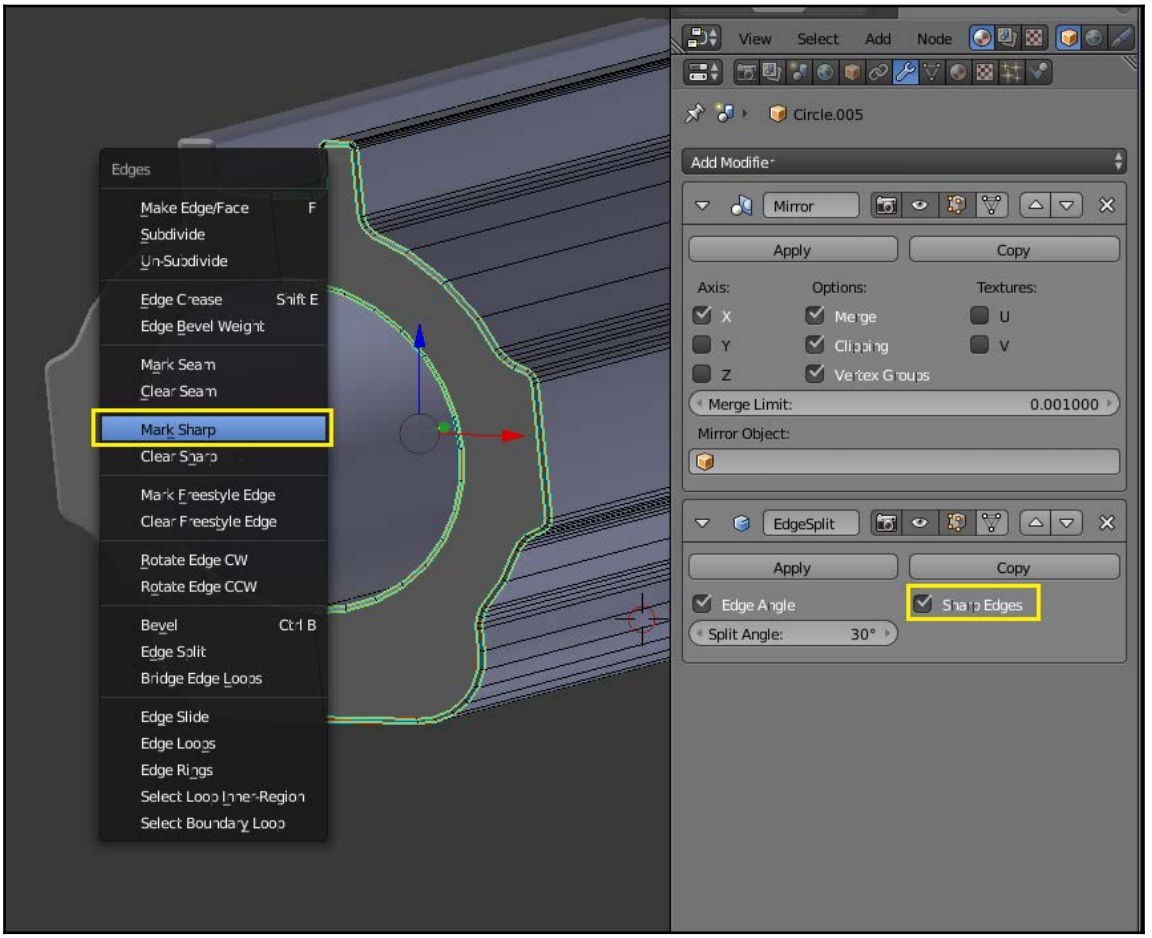


Next, you can just delete the faces with the X key. Then, select the edge loops that form the front of the barrel and press *F*. This will automatically fill in a single N-Gon on the front of the barrel.



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.

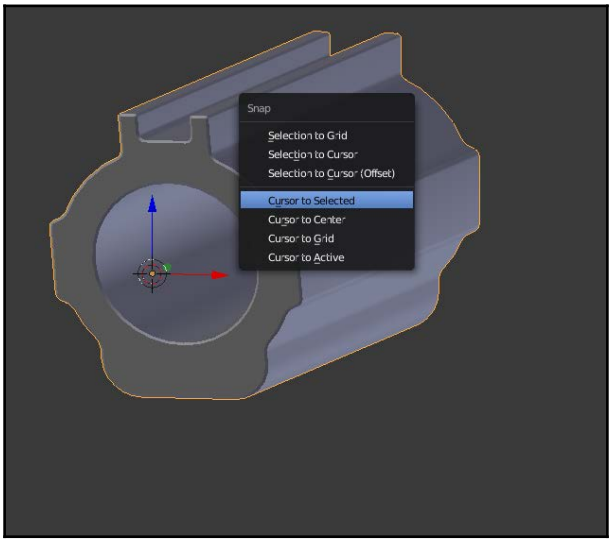
When you've done this, just select the two edge loops that make up the barrel and press *Ctrl + E* and **Mark Sharp**. The angles will show up as sharp anyway (since they probably exceed 30°), but if you know that you want an edge to be sharp, it never hurts to mark it that way. You may make changes down the line that affect an edge's angle. This way, you never have to worry about it.



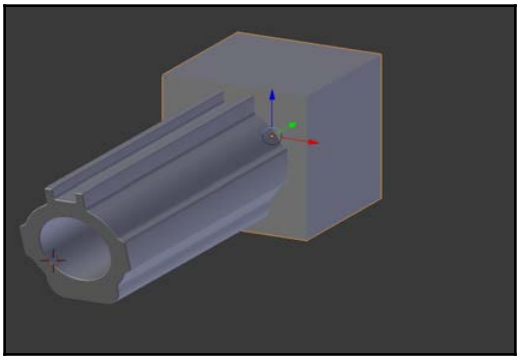
At this point, the basic shape of our barrel is complete. Make any final corrections that you'd like, and then we'll move on to the rest of the gun.

Modeling a handgrip and other pieces

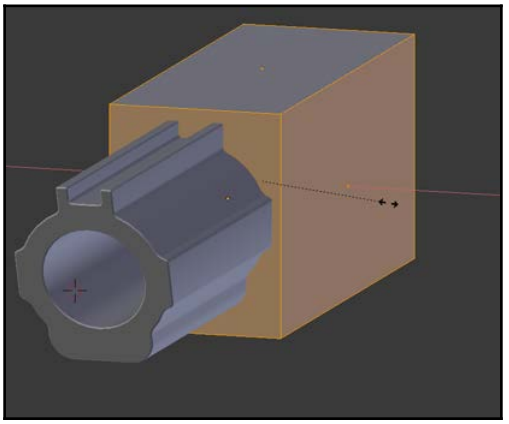
The first thing we want to do is select the barrel again, and then press *Shift +S* and **Cursor to Selected**. This will place the 3D cursor at the origin point of our gun object. Any objects that we add after this will be automatically placed there:



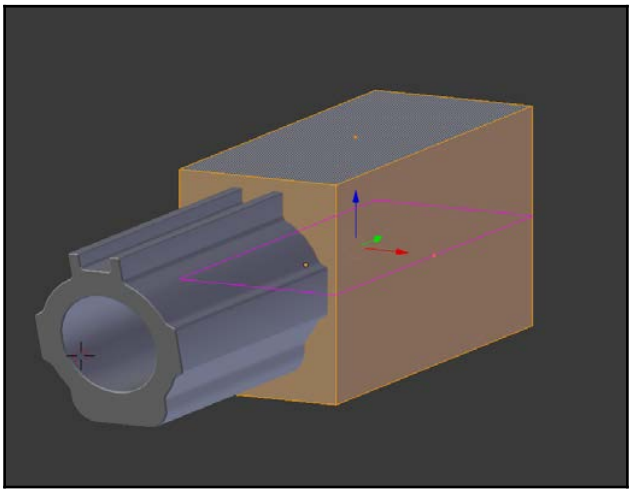
Next, we'll bring up the **Add** menu again and add a cube to our project. It will appear directly in line with the gun barrel. Depending on the scale of your barrel, the cube may appear slightly bigger or smaller than the image shown here:



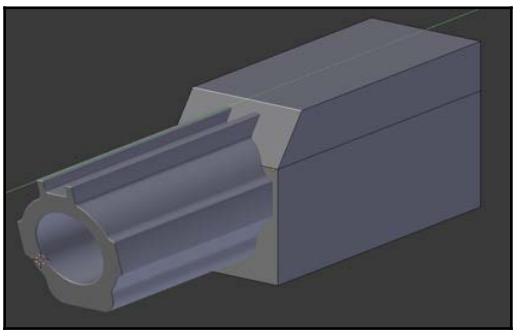
Tab into the **Edit mode** and adjust the scale until it's to your liking.



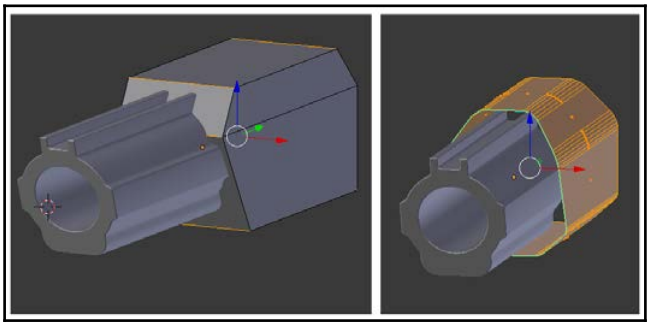
Next, use *Ctrl + R* to run a **loop cut** around the outside of the cube. This will enable us to make a few changes to the shape.



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.



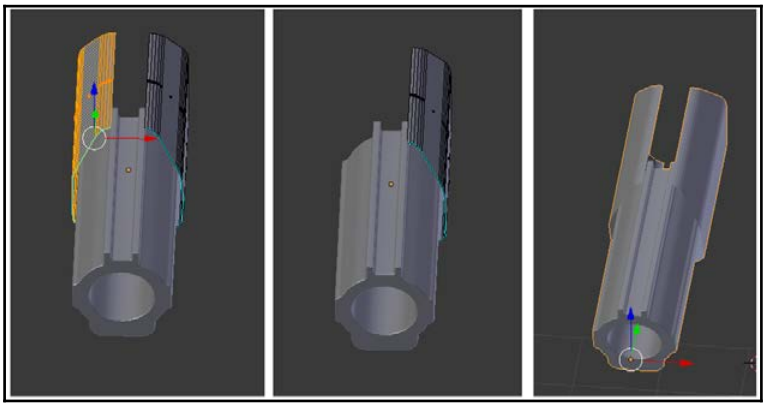
Once you have a shape that you like, we'll go ahead and join this object to our barrel. In order to do this, we'll need to make a few changes.

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).

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.

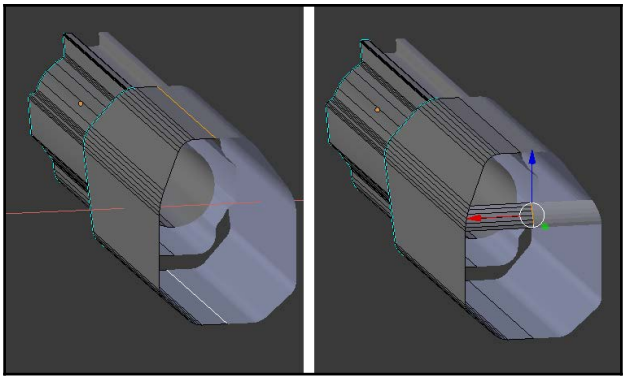


The first object that you pick will always be joined to the second object.



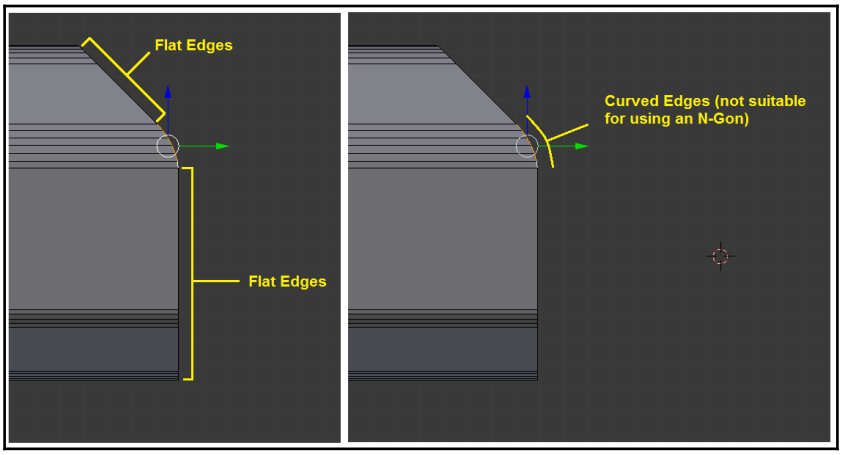
Next, we'll grab the very top and bottom edges of our gun block and extrude them in toward the center (on the *X* axis). Since **Clipping** was selected on our **Mirror** modifier, the edges will not go past the object's centerline.

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.

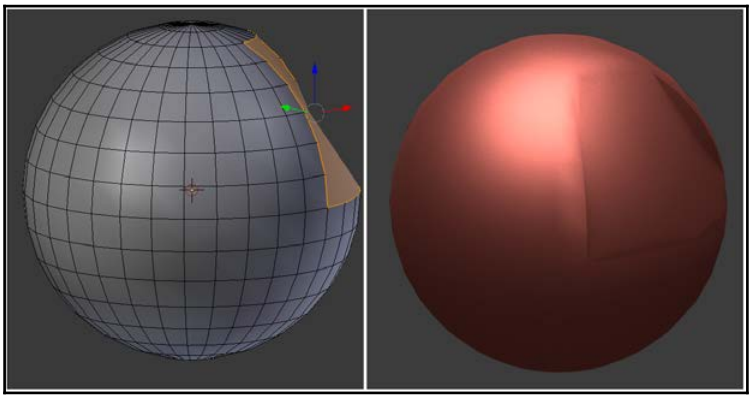


Next, we'll be filling in some faces on the back of the gun's body. We'll use a combination of N-Gons and quads to do this.

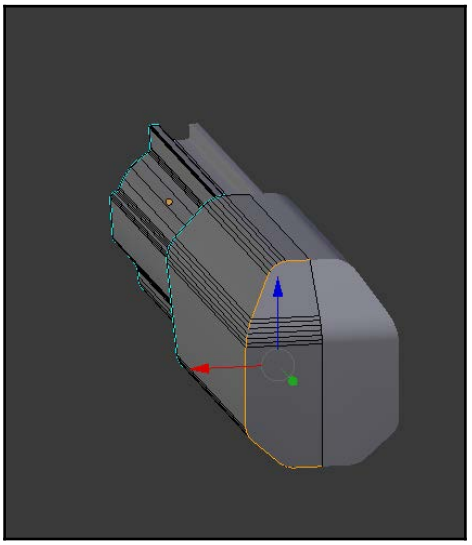
It's important to know where you can use an N-Gon and where you can't. The most important rule about N-Gons is that they must always be flat. As we look at the back of our gun's body, we can see a series of small curved edges in the middle:



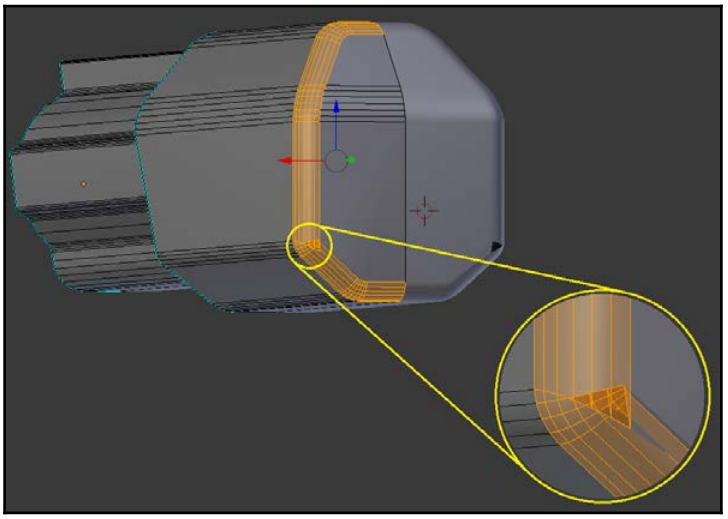
To better understand this rule, look at the following image. You can see that the portions of a sphere have been cut out and replaced with a single, curved N-Gon. The N-Gon is bending to connect various edges together. This dramatically affects both the shape and shading of your object, and it will create unwanted artifacts at render time:



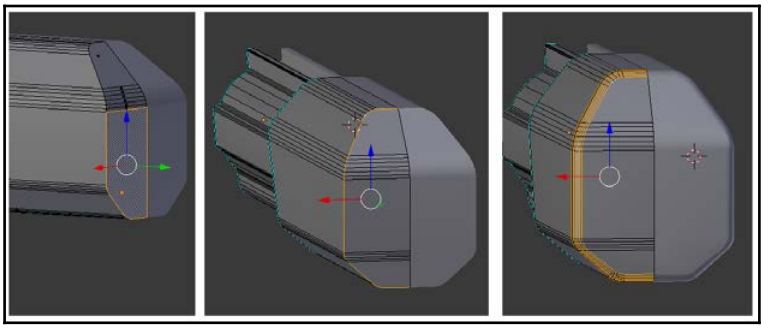
Going back to our gun's body, we'll fill flat edges in with N-Gons. When you're finished with this, select the outer ring of edges at the back.



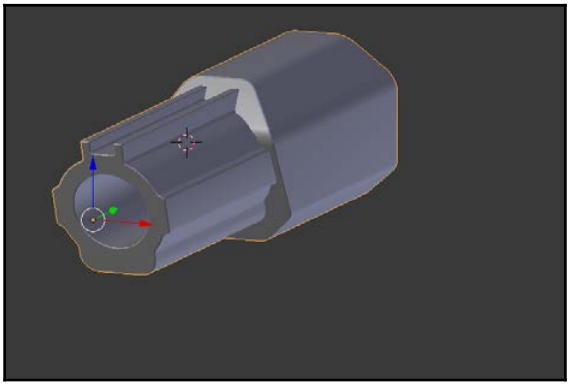
We'll attempt to bevel this in the same way we did before. However, depending on the exact shape of your gun's body, you may end up with something like this:



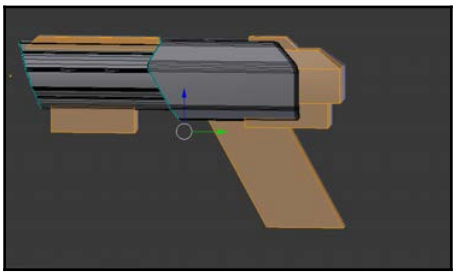
What you're seeing here is beveled edges that are overlapping each other. This is a fairly common occurrence with the bevel tool. When it happens, the best fix is to change the geometry a little bit. In this case, we'll remove the bottom N-Gon first. Then, we can extrude a series of small edges across, 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!

2
Sci-Fi Pistol - Adding Details

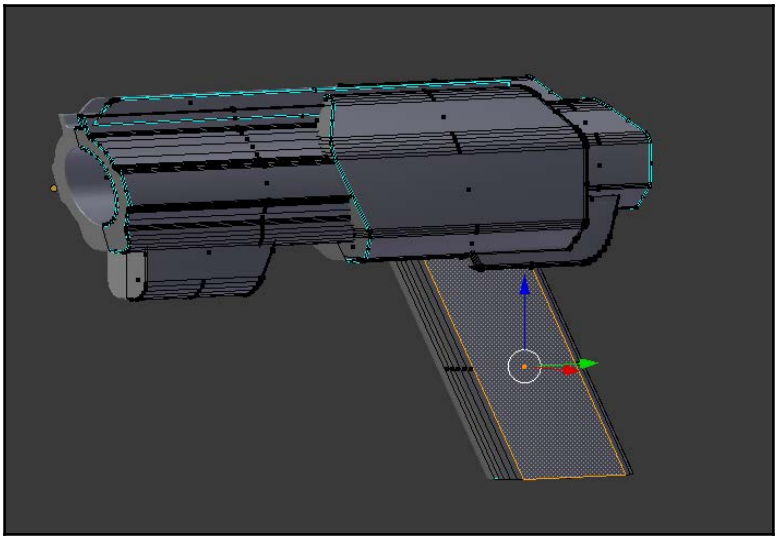
In this chapter, we'll cover some additional topics as we work on our pistol. These include the following:

- 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

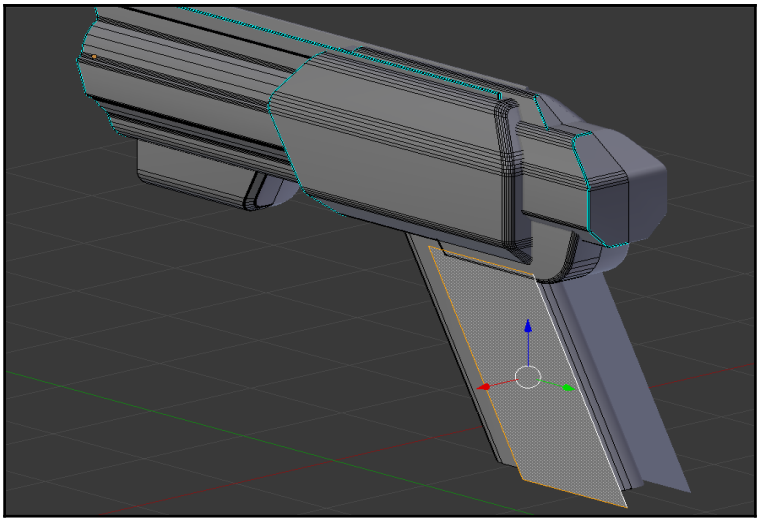
Finishing the handgrip

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.

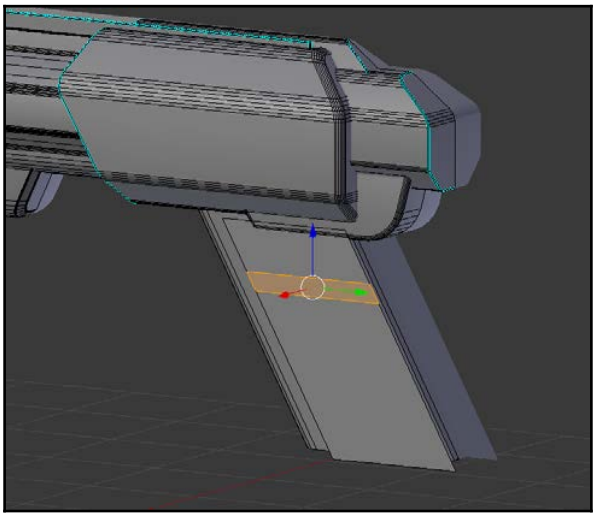
The first thing we'll do is select the large face at the side of the handgrip:



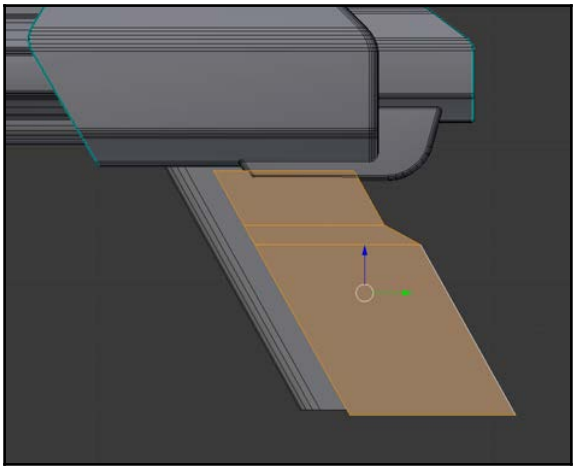
We'll duplicate this by pressing *Shift* + *D* and move it back a bit:



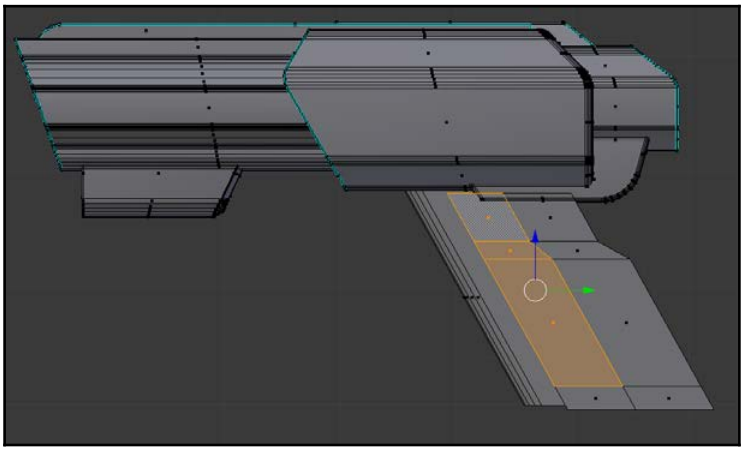
To add a little more detail, let's run a couple of loop cuts across this face by pressing *Ctrl* + *R* + wheel up (rolling the mouse wheel changes the number of edge loops):



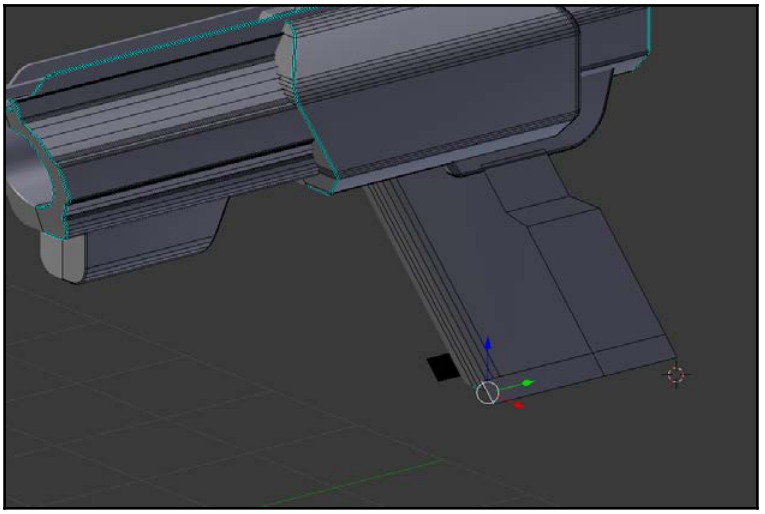
Then, we can drag the lower section back a bit by selecting the bottom-right edge and moving it to the *Y* axis:



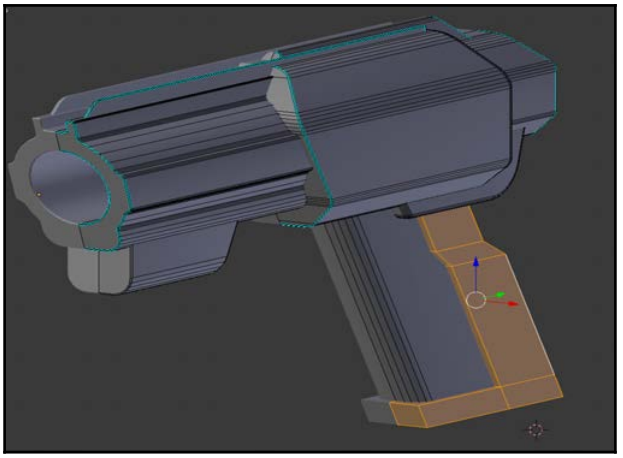
Next, let's run another edge loop vertically so that we can start forming the shape of the cut-out (for the grip):



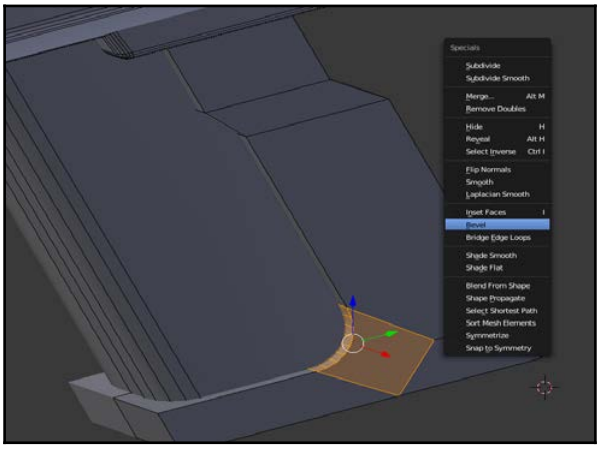
We'll delete these unneeded faces. Next, let's grab this front edge and pull it forward:



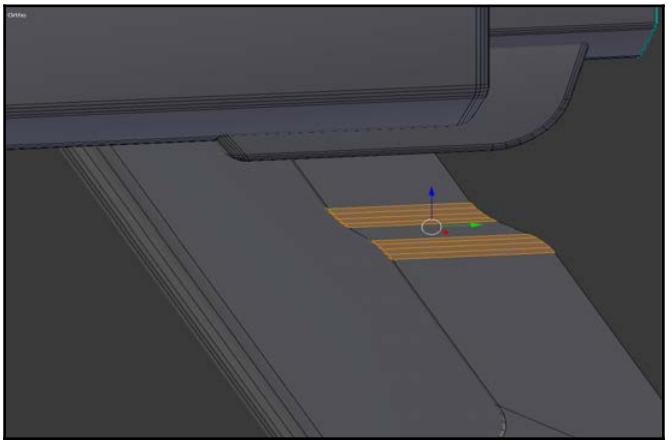
Then, we'll extrude everything across the whole plane to get the basic shape. Before going further, you'll want to remove the faces inside of the mesh (right where the two have joined at the mirror modifier). We don't want any faces inside of our mesh as it will affect shading later on.



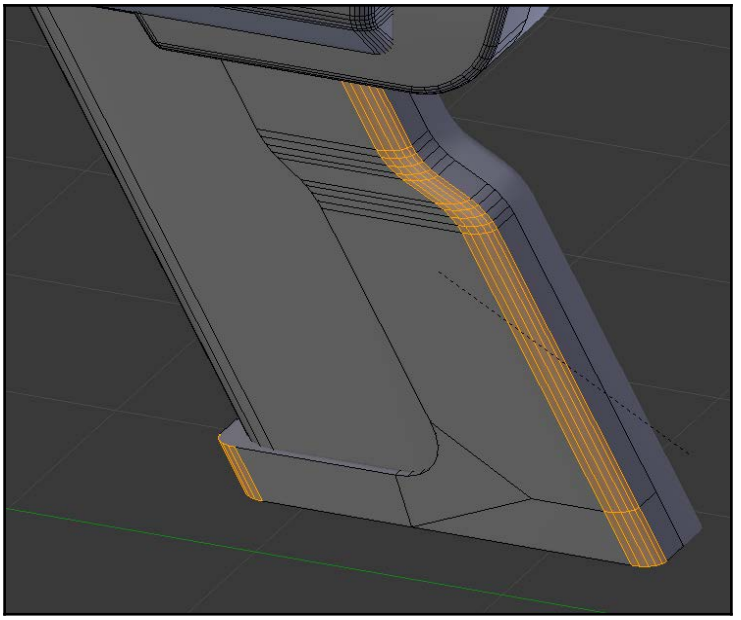
Let's **Bevel** this corner edge at the bottom to get a more rounded shape:



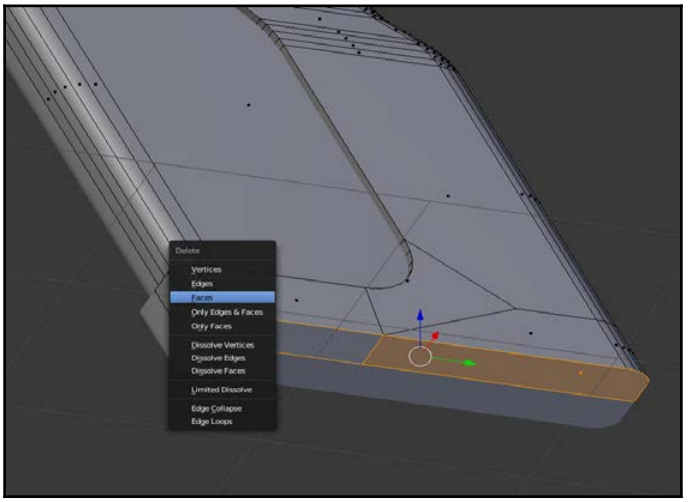
Next, we'll **Bevel** these edges here as well for a softer transition:



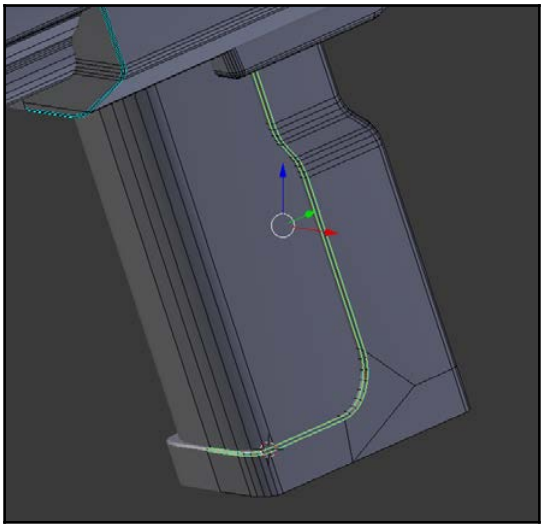
Next, we'll **Bevel** the corner edges so they're nice and smooth:



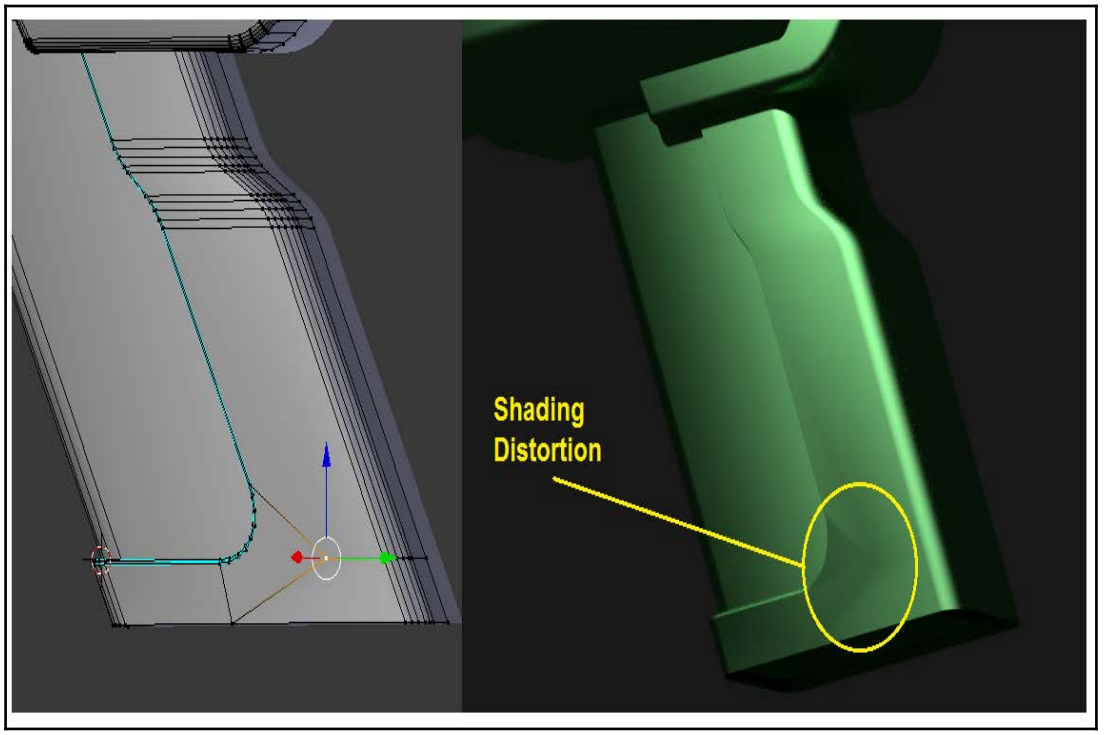
We can get rid of the **Faces** on the extreme bottom here as well:



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:

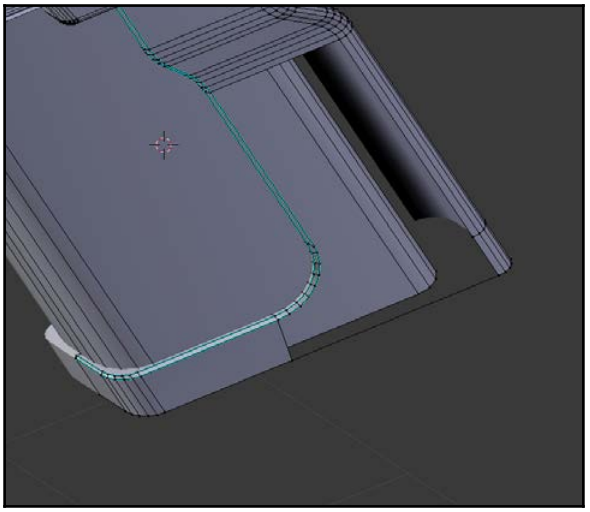


This looking pretty good; one problem we have here is that our earlier beveling has left a single vertex in the middle of the handle. If the back of the handle were hard (sharp), this wouldn't be a problem; because it's curved, it will affect the shading. I've added a temporary material and a rendering setup just to demonstrate what the problem will be here:

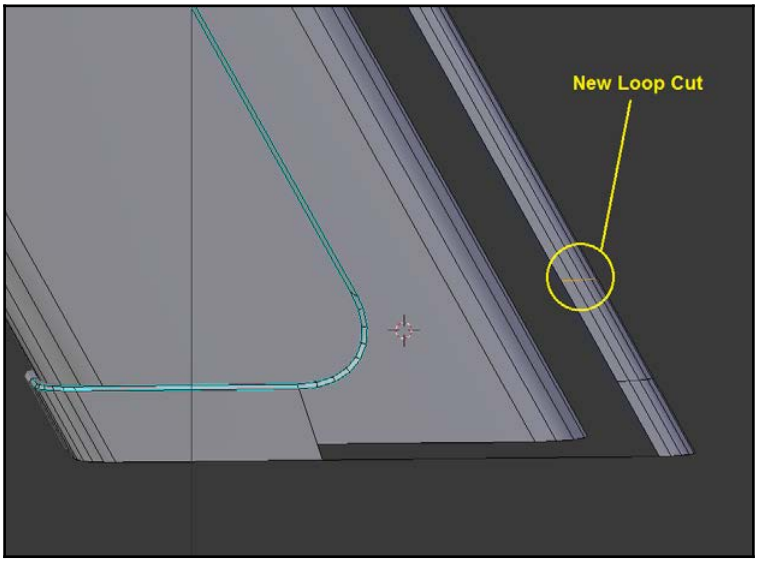


As you can see, the shading in the image is distorted. This is what can happen when you have a single vertex in the middle of a smooth (curved) surface. We don't want the vertex here, so let's fix it.

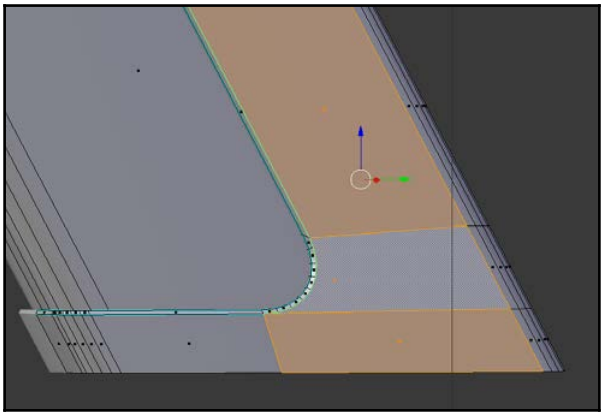
First, we'll just delete the vertex itself, which will eliminate the surrounding faces:



Then, we'll run a loop cut across the back of the handle:

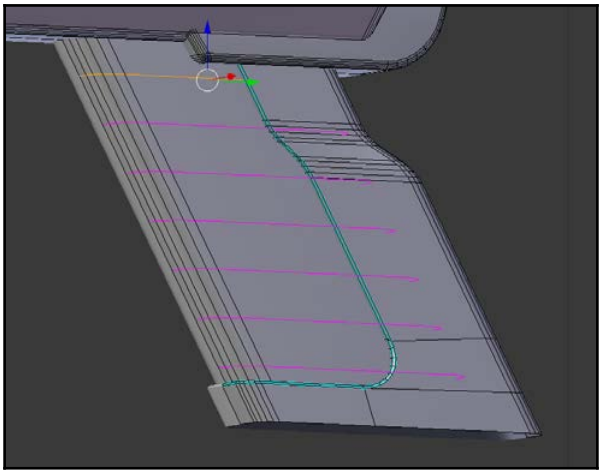


Next, we'll fill in some geometry:

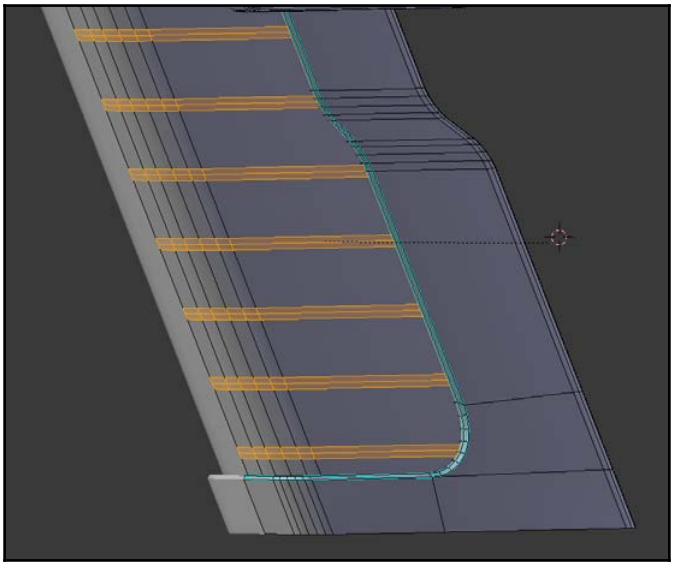


This will take care of our shading problem.

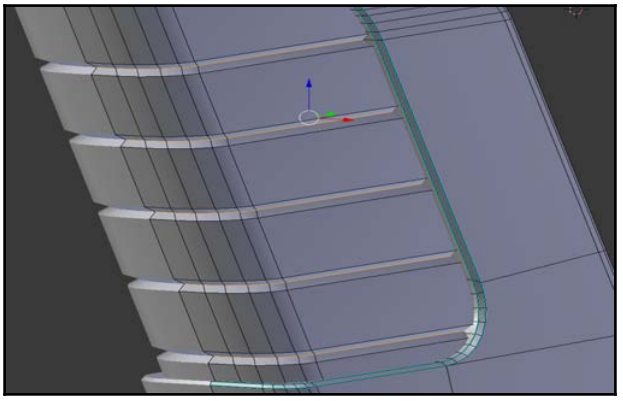
Next, let's run a few loop cuts along the pistol grip itself:



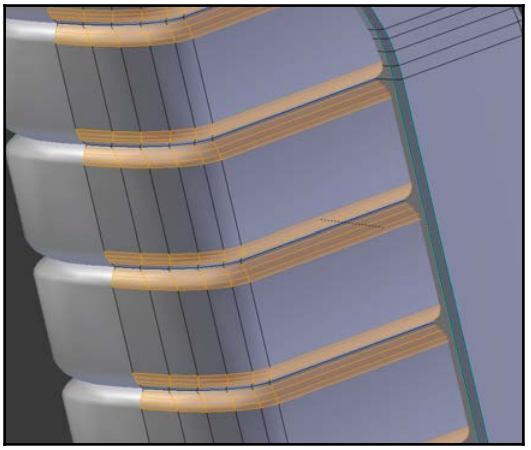
We can bevel these a little, rolling the mouse wheel as we do so to create a series of faces:



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.

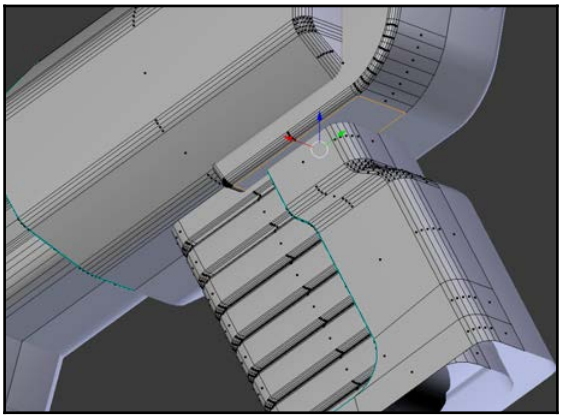


Now, we can bevel the edges a bit more to round things off:

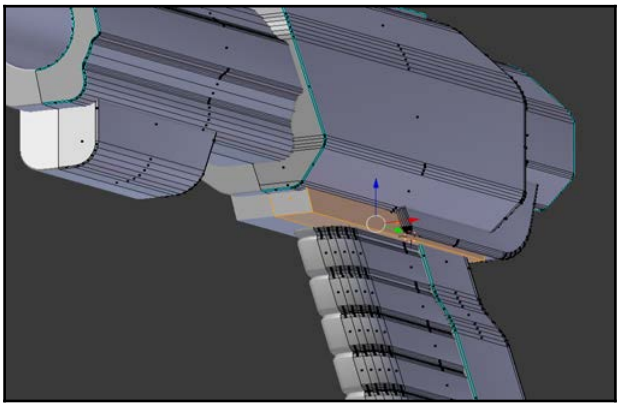


This looks pretty good.

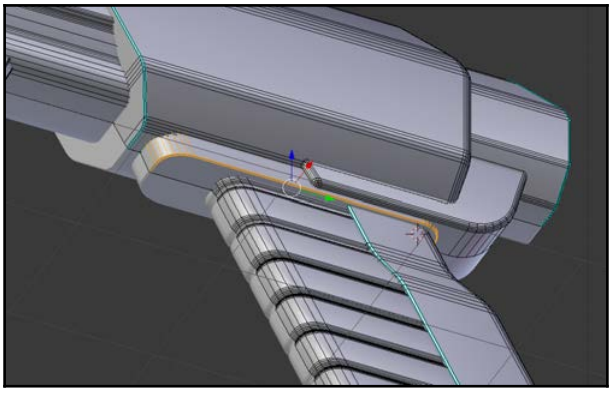
Next, let's just duplicate an existing face with *Shift* + *D*.



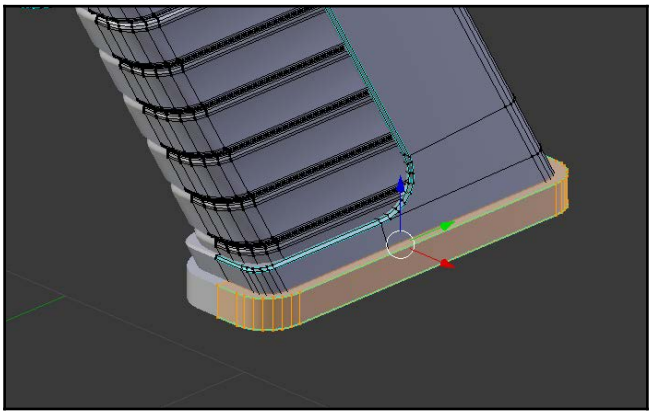
We can move and extrude it up to create a base for our pistol grip. This will just add a little more detail to the area where it meets the main body of the gun:



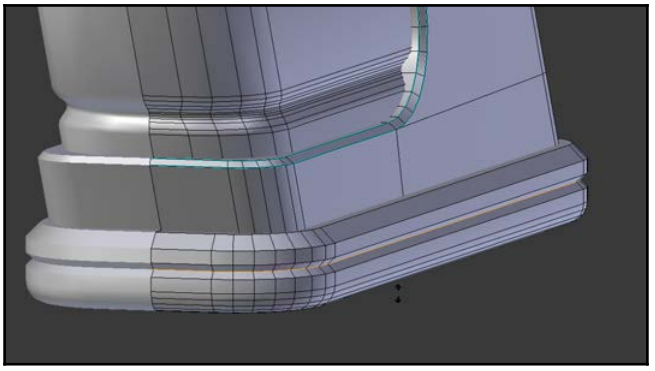
We'll bevel it until we get a nice shape using the same techniques that we've already used:



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:



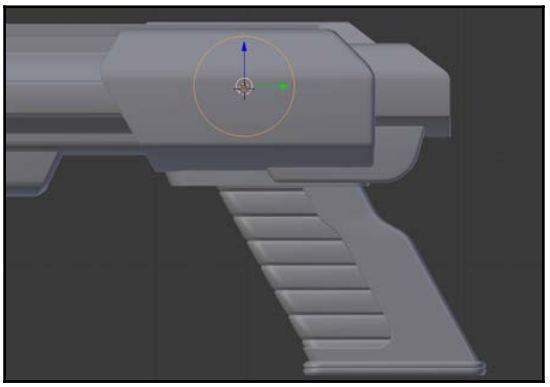
We can detail this area a bit using techniques that we've already covered.



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.

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:



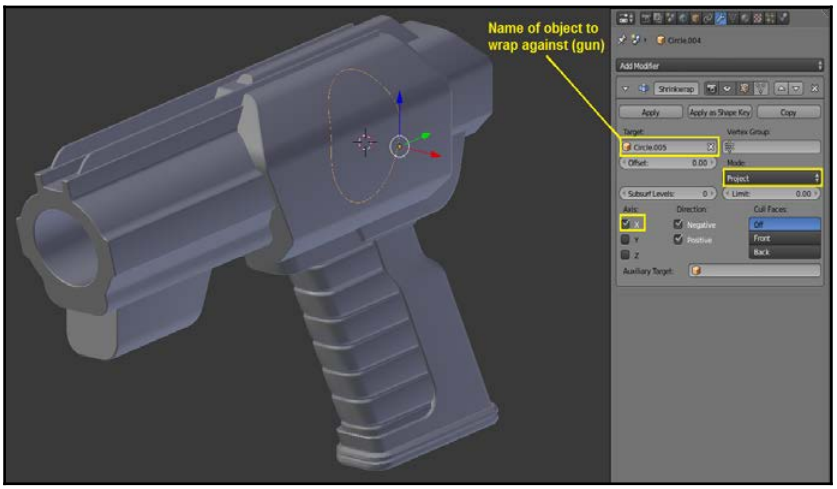
Next, we're going to add a **Shrinkwrap** modifier to the circle. This is an incredibly powerful modifier to create cuts or recesses in mechanical parts.

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.

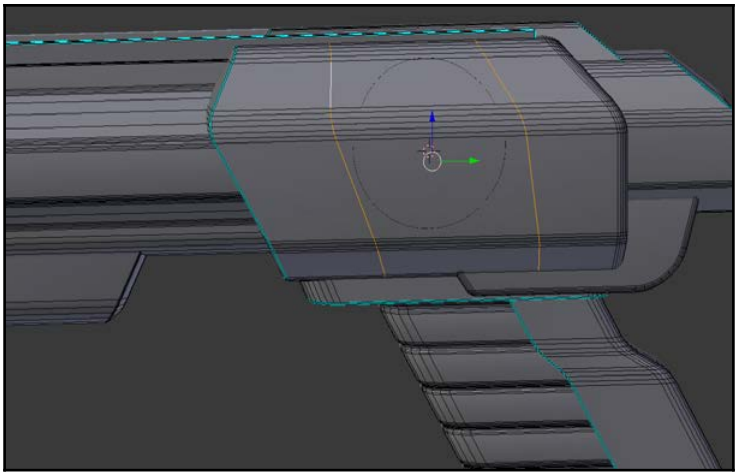
Once these settings are selected, you can drag the circle up against the gun, and it will conform to this shape.



The more geometry (vertices/edges/faces) an object has, the better it will be able to conform to the shape of something else.



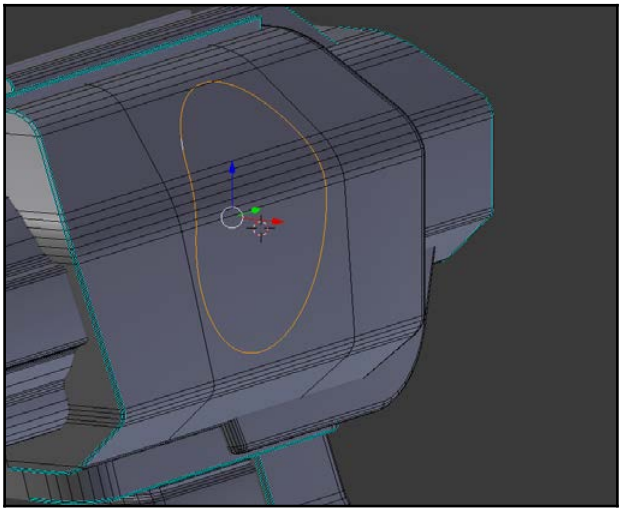
Let's pick our gun object again, and *Tab* back into **Edit mode**. We want to run a couple of loop cuts along the main body section. This will make it easier to fill in the shapes later:



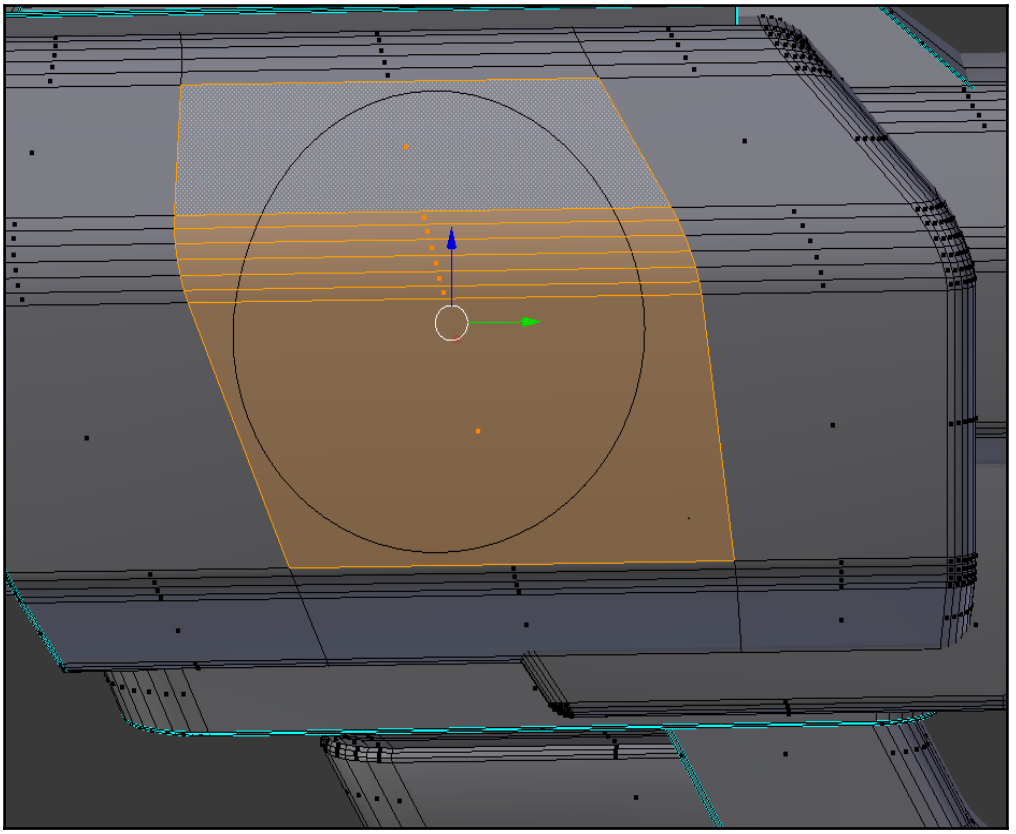
Once we've done this, let's go ahead and apply the **Shrinkwrap** modifier to our circle. This will permanently change its shape and conform to the gun:



Next, we'll join the circle object to our gun by pressing *Ctrl + J*.



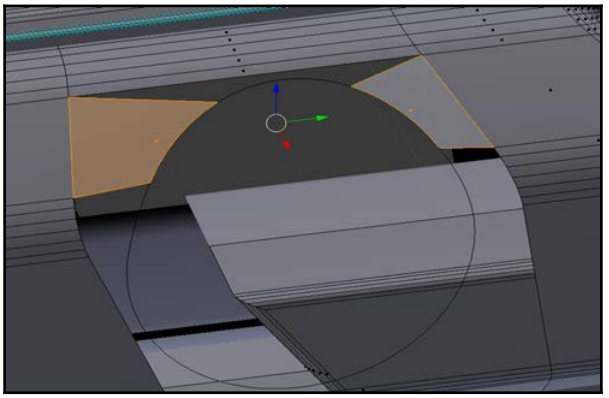
Let's delete the back faces here, so we can fill in our mesh:



This next part is a bit tricky. We'll need to fill in the spaces between the circle and the rest of our mesh.

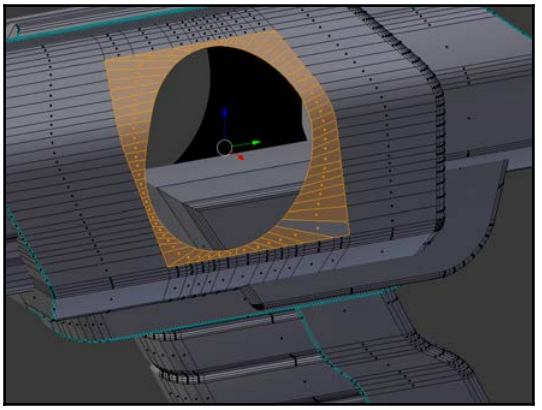
I should mention that cutting shapes into curved objects is one of the hardest skills to master. There are several tools that can help you (such as Knife Project, which we'll look at momentarily), but they all have their limitations. At the end of the day, it really comes down to practice.

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.

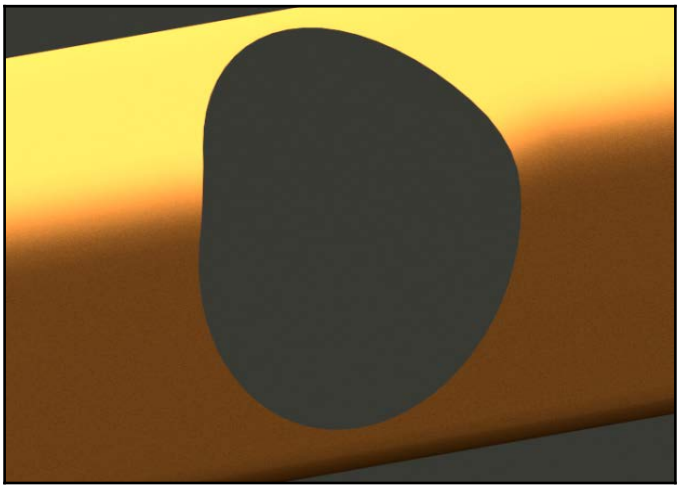


As you go through this part, you'll probably find yourself needing to add loop cuts to the existing gun body. This is fine; go ahead and do it. More geometry tends to give you better definition, so don't be afraid to add some. As much as possible, we'd like our new geometry to follow the lines/curves of what was there before.

Here's what I ended up with:



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.

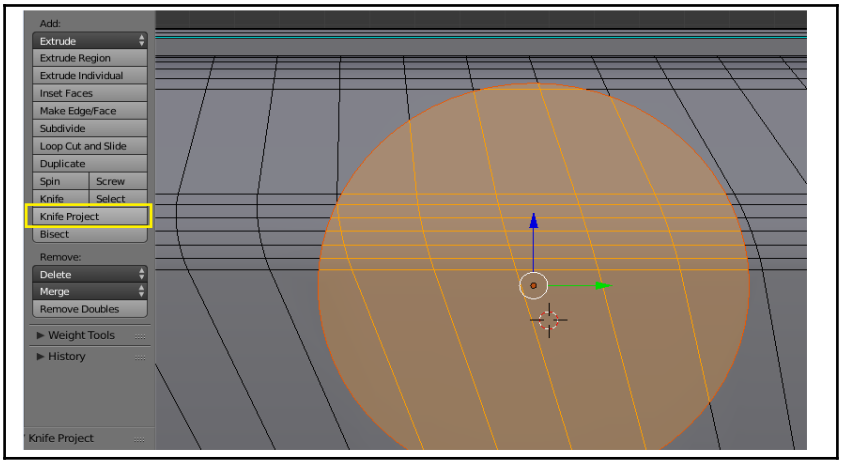
Before we move on to the next part of our project, it's worth taking a moment to examine the **Knife Project** tool. Its purpose is to cut shapes into existing objects; some readers will undoubtedly wonder why we didn't use it here.

Let's take a look.

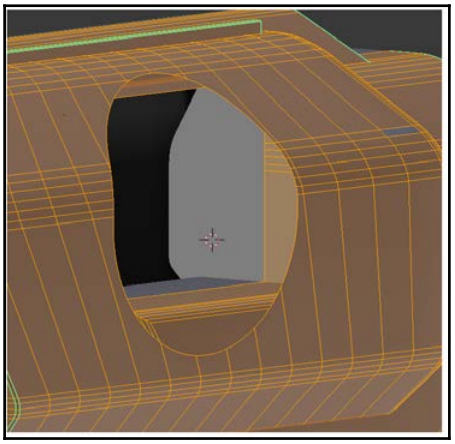
To use the **Knife Project** tool, you simply need to add a circle object and line it up with where you wanted your cut to go.

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:

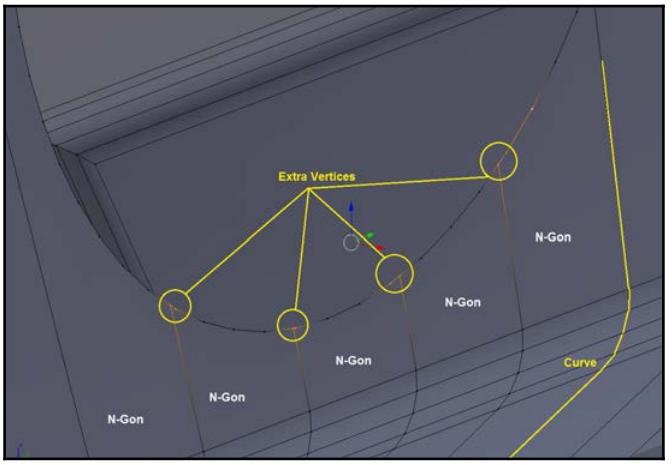


Once we've removed the faces, we'll be left with this:

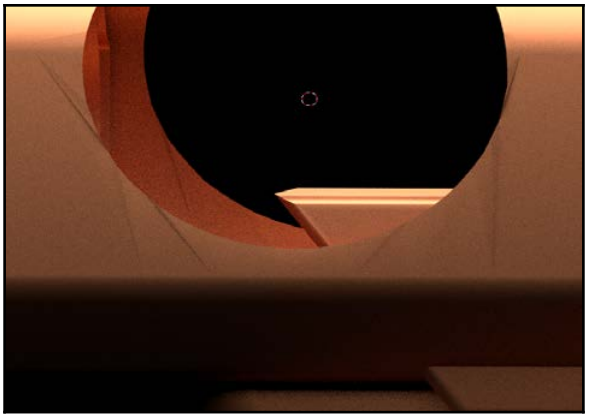


It looks pretty good, doesn't it? Nice and clean. So why don't we use it?

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.



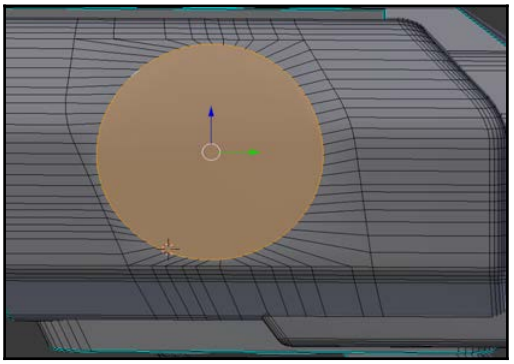
As we've seen before, this can have an adverse effect on shading.



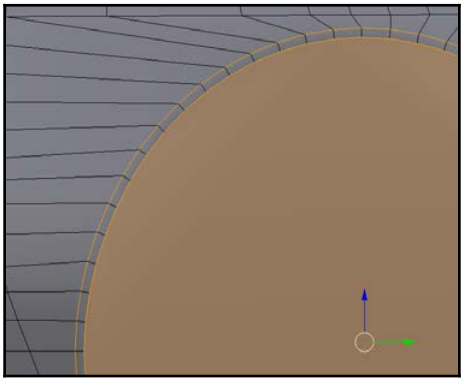
You can see here that extra vertices create a few shading problems. Is it possible to fix these problems? Sure! You can join (merge) the vertices together and move them around until the shading looks good.

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.

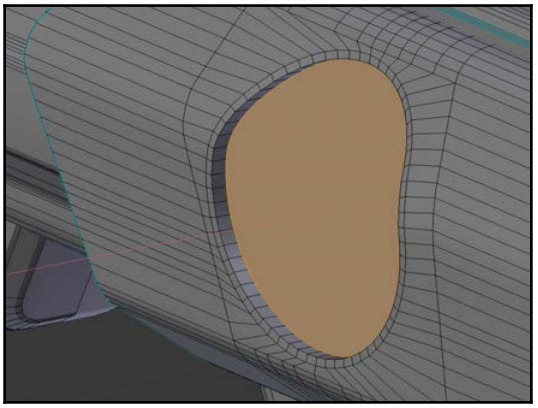


Then, we'll use **Inset** tool by pressing the *I* key. This will extrude a new face towards the center, leaving a nice even border.

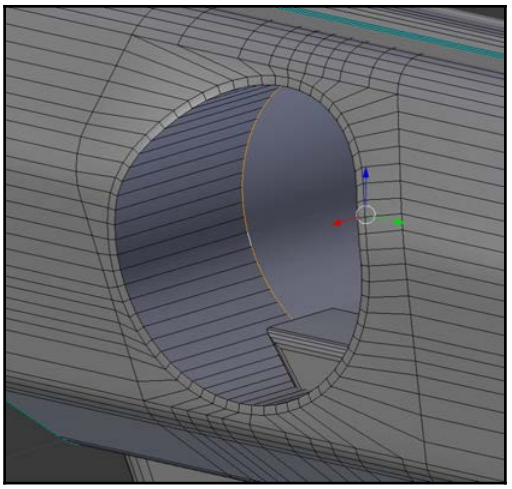


This is a key step, and it often gets overlooked. By adding the extra ring of edges (with the **Inset** tool), we preserve the shading around the edge of recessed areas.

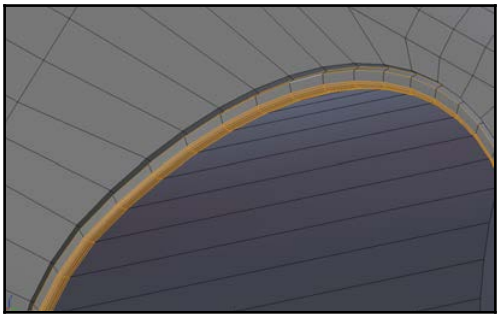
Once we've got the ring, we'll again extrude the circular face towards the center of the gun.



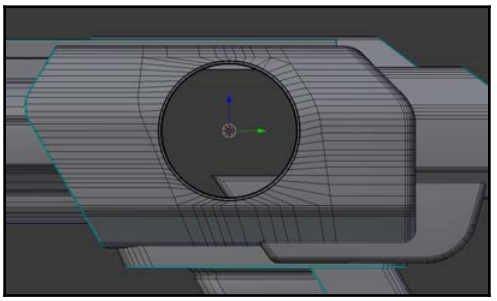
As the face hits the center point of our gun object, the **Mirror** modifier will force it to become flat. Once this happens, delete the face, and you're left with a nice, cylindrical hole through the center of the gun's body.



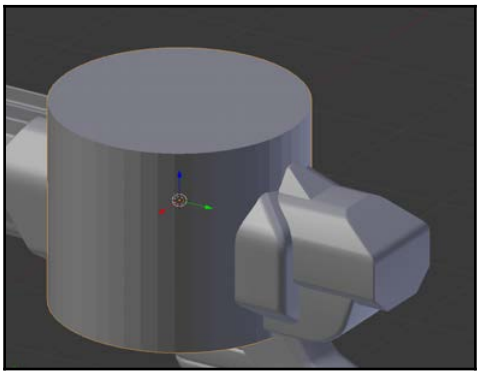
If you'd like, you can now go through and add a little detail to it by beveling the edges.



Once this is done, the gun's body will look like this:

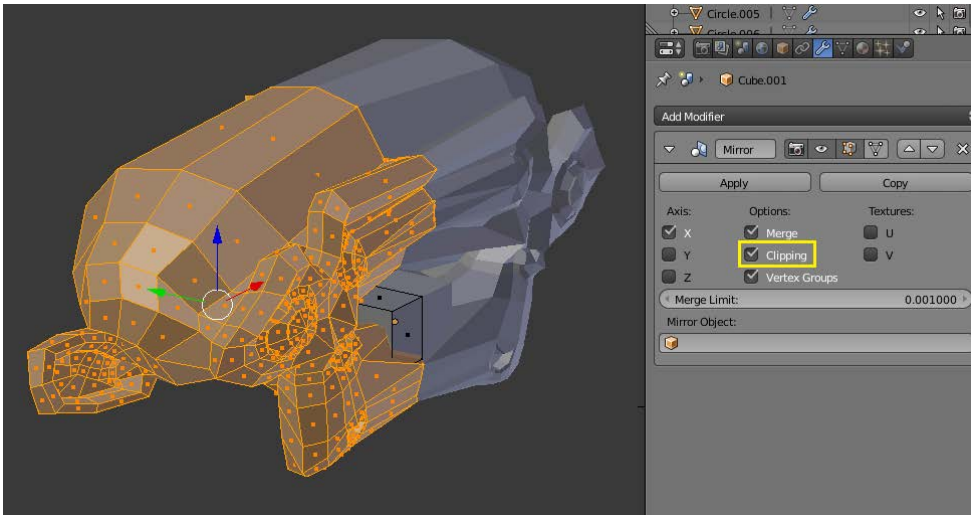


Next, we'll go ahead and add a cylinder in **Object** mode.

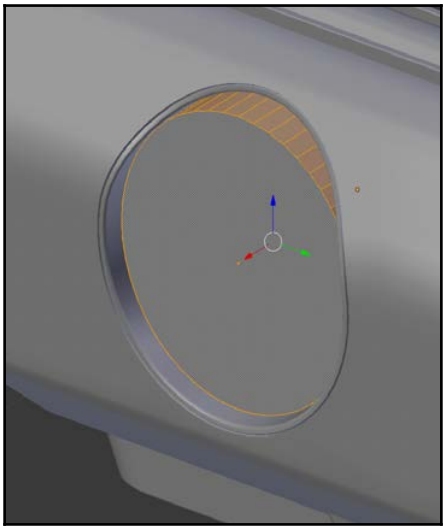


Adding Parts to Mirrored Objects

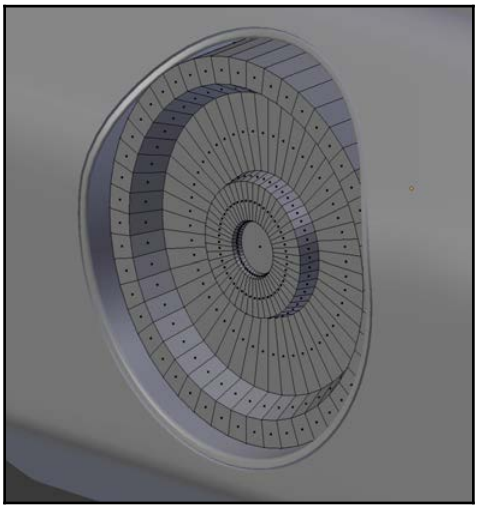
When using the Mirror modifier, it's often convenient to add parts in Object mode (versus adding them in Edit mode to the existing model). When Clipping is enabled on your Mirror modifier, you may accidentally distort your object if it's overlapping the center point.



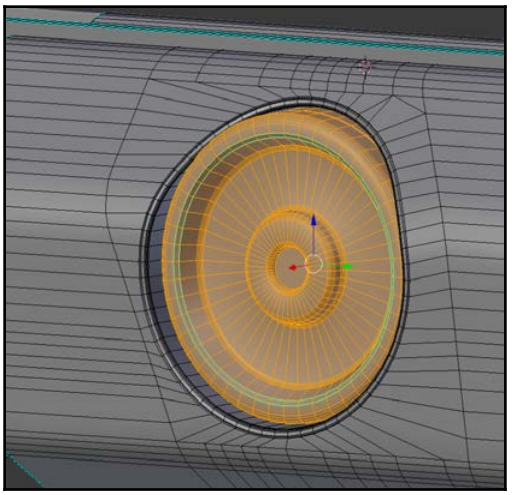
Next, we'll scale and move the cylinder into position:



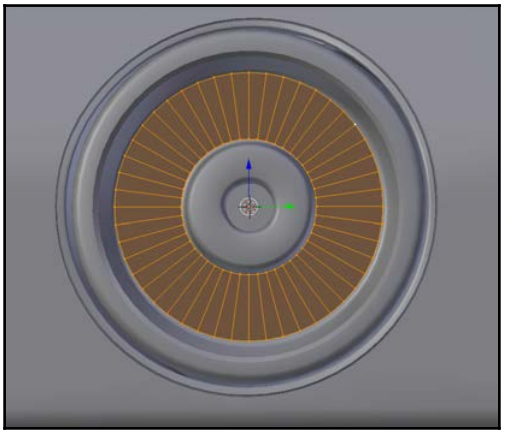
Using the and **Inset Extrude** and **Inset** tools, we'll create some basic shapes:



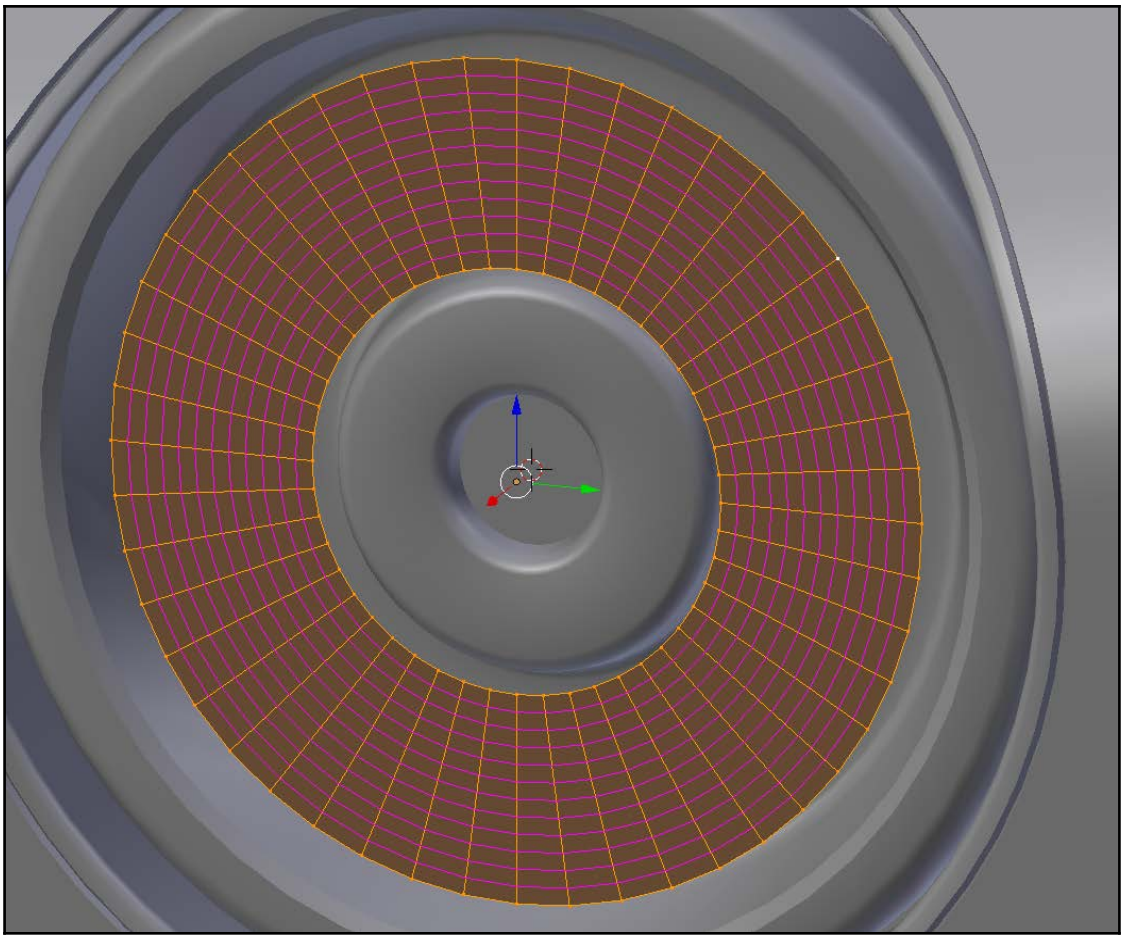
Then, we'll bevel our edges to add some detail. Once the cylinder's in position, you can go ahead and join it to the gun model by pressing *Ctrl* + *J*.



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.



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.



Now that we've done this, let's turn **Proportional Editing** on at the bottom of our screen and select **Sharp**.

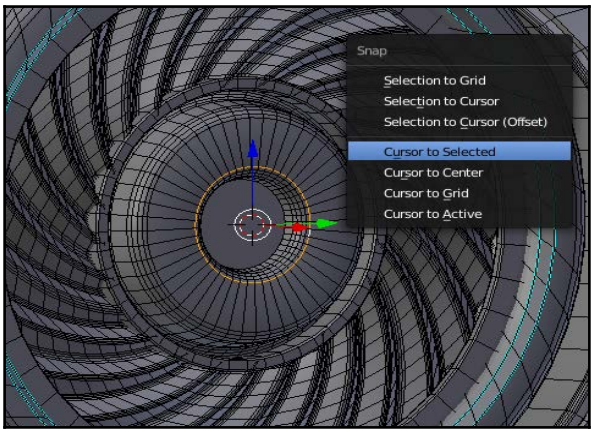
Then, we can rotate the middle ring of vertices, and we can scroll the mouse wheel to adjust the area of influence. Just rotate and adjust until you have given a nice shape to your circle.



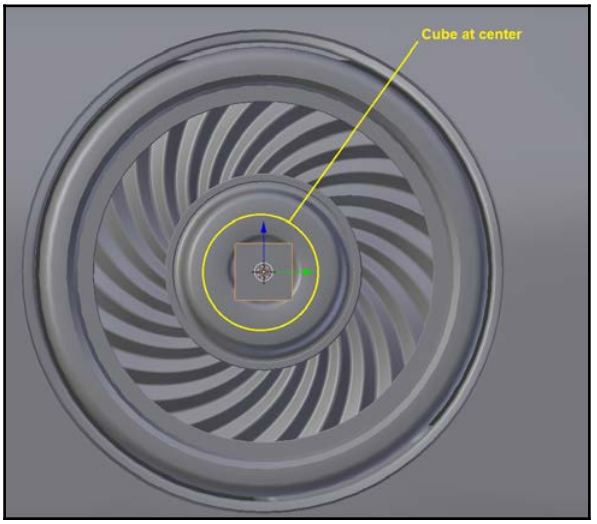
Next, we'll select all other edge loops and push them back towards the center of the gun. Once we have the basic shape of our fins, we can go in there and bevel them. If you'd prefer, you can also delete a few rows of faces to create gaps or holes for heat to escape (this is not shown here).



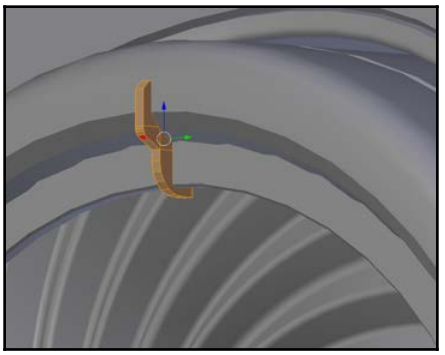
The next thing we'll do is add a little circular detail to the outside of our recessed circular area. Start by placing the **3D cursor** at the center of the circle. An easy way to do this is to pick a ring of edges and press *Shift* + *S*, **Cursor to Selected**.



Then, go ahead and add a cube.

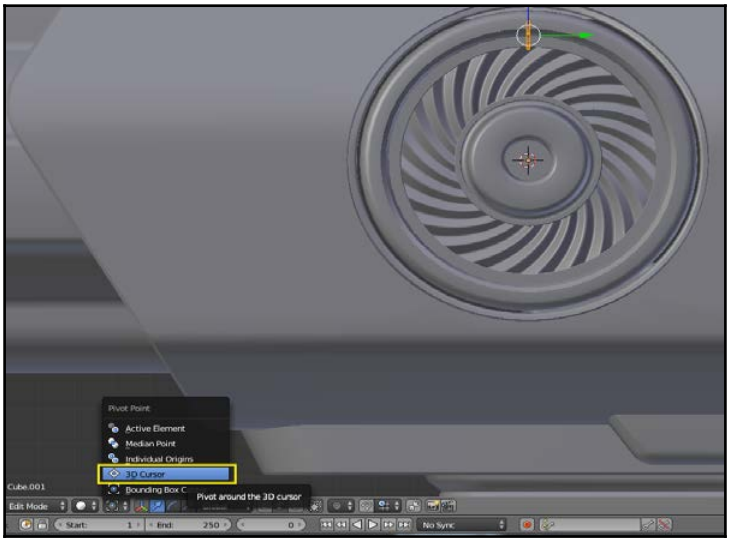


Next, *Tab* into **Edit mode** on the cube and pull it up to the border of our circle. Using techniques already demonstrated, go ahead and add a bit of detail to it.



Next, ensure the cursor is still at the center of the circle. If it isn't, go ahead and put it there.

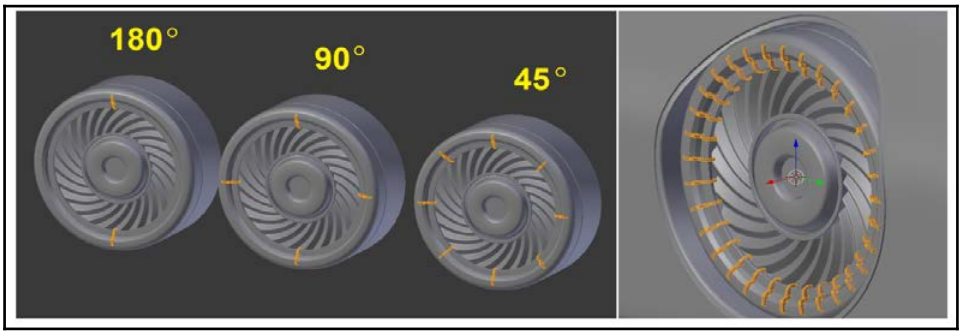
At the bottom of your screen, change the **Pivot Point** around which things rotate to **3D Cursor**.



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.

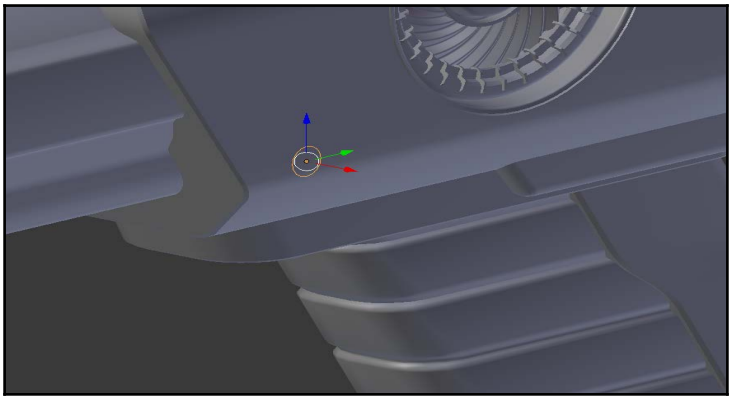
Repeat this process as desired, dividing the angle in half each time. The math will break down as follows:

- 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:

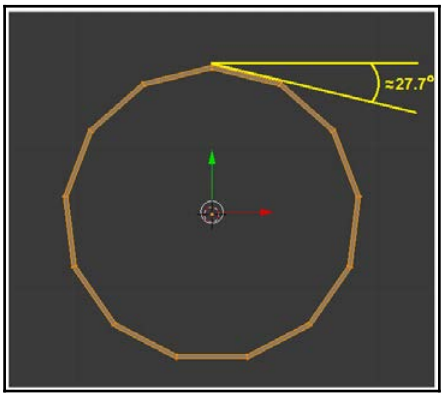


Going into **Edit mode** on your circle and move it around until it's in a location where you'd like a screw/bolt to go. Duplicate the circle and repeat this until you're satisfied with the number/location of holes:



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.

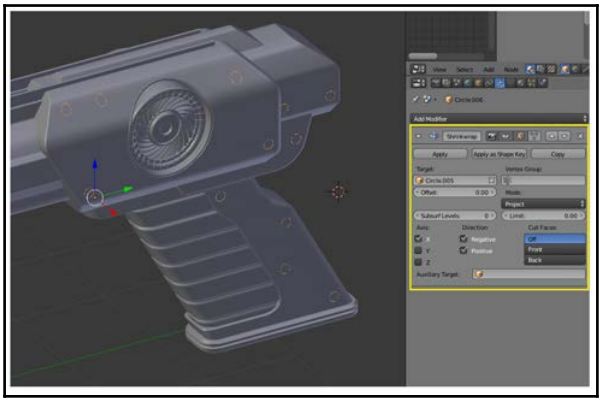


Any fewer sides and your angles are going to break the 30° mark and become sharp. So, 13 is the mathematical minimum. In practice, however, it's not an easy number to work with.

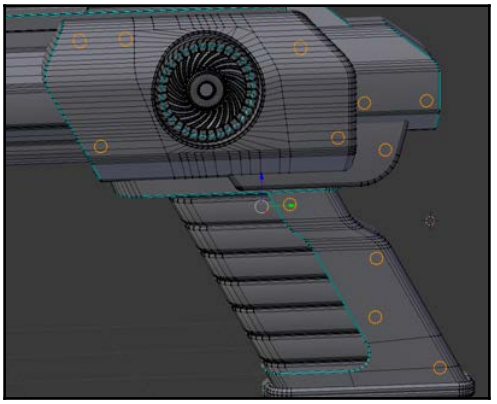
You can't cut it into halves or thirds, and each angle is a long decimal number (27.6923076…). This makes it hard to rotate something (such as a cut out) around a cylinder by "one side".

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.

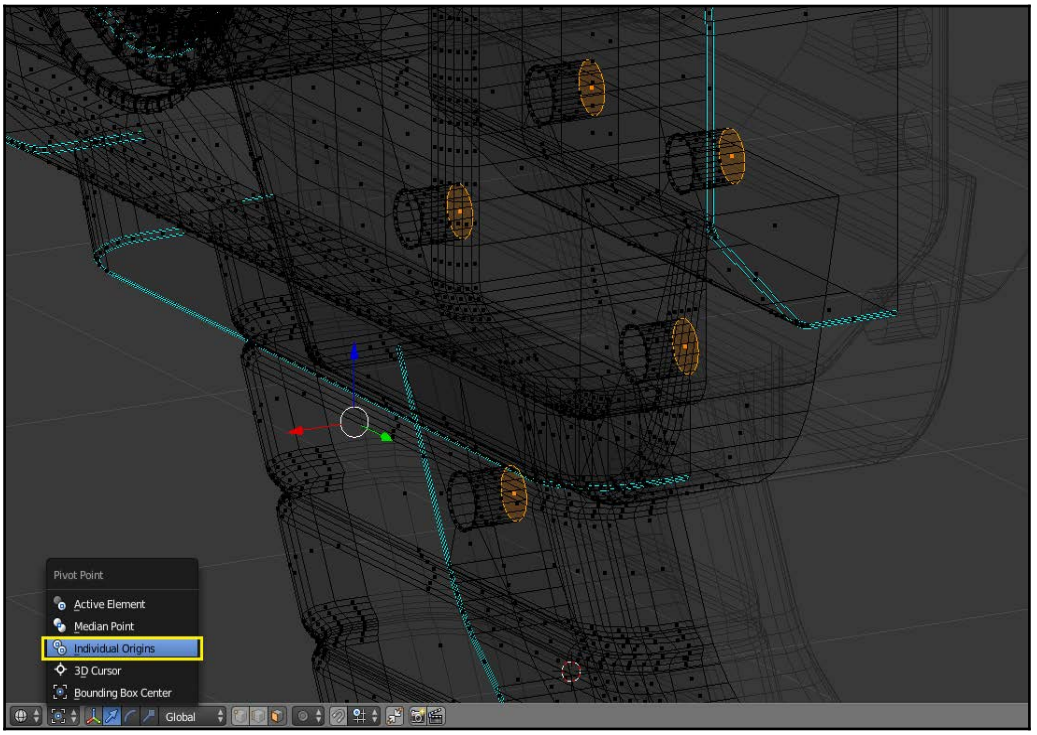


Just like before, apply the **Shrinkwrap** modifier when you're done, and then join the circle object to the gun.



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.

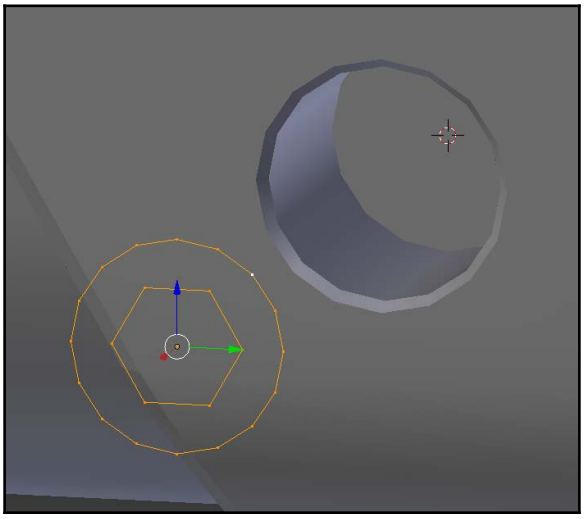
Once this is done, we'll extrude the edges of these circles back. Unlike our larger circle from earlier, we won't take them all the way to the centre of the object. Instead, we'll pull them back just a little bit. Then, we'll select **Individual Origins** for our **Pivot Point**. Once this is selected, 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.



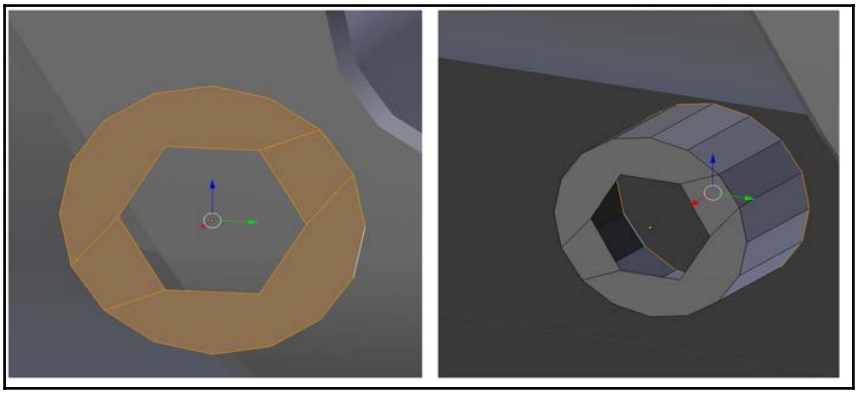
You will now have some good holes for screws/bolts to go in:



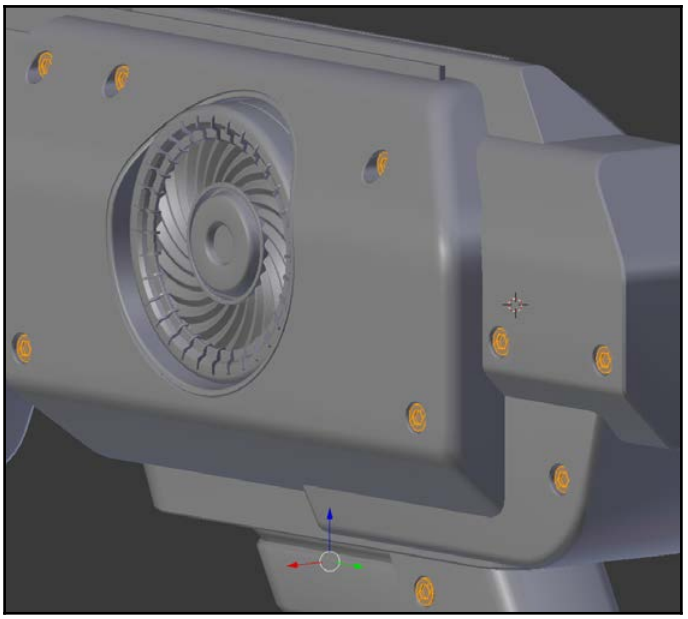
Starting with a circle, go ahead and build a quick bolt/screw:



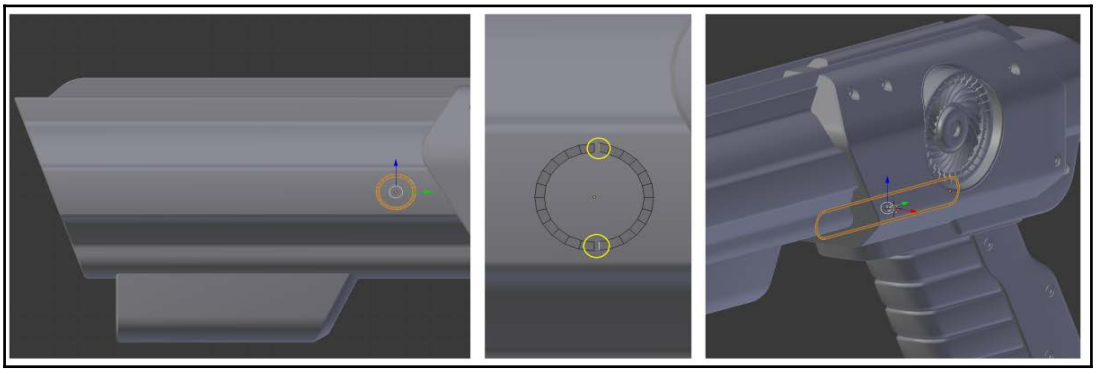
You can also extrude some 2D shapes back to create it:



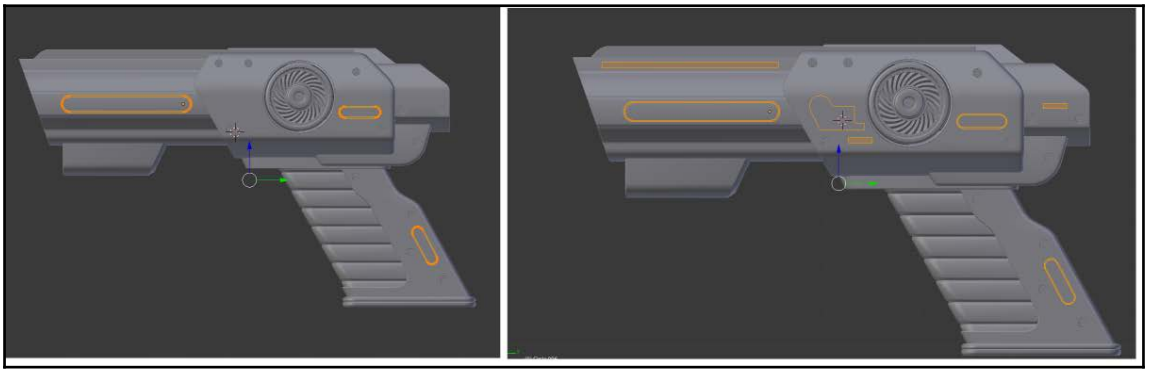
Then, move this into position. Duplicate the bolt/screw, and move the copies around until all of your holes are filled in:



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:



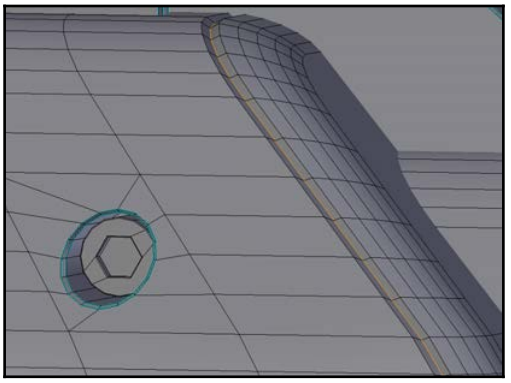
Here's what this gun looks like with all the cuts/holes in place:



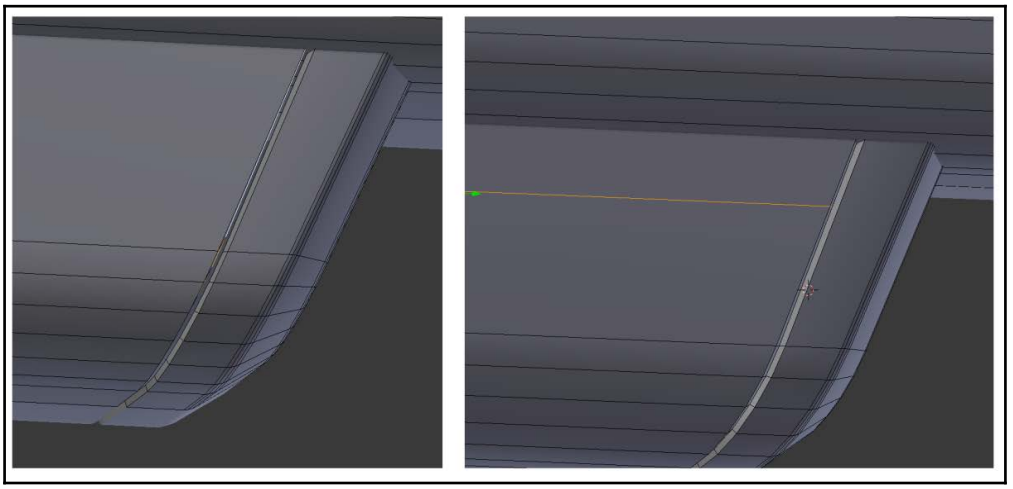
Adding final details

At this point, the gun is really starting to take shape. Let's add a little more detail to finish it up.

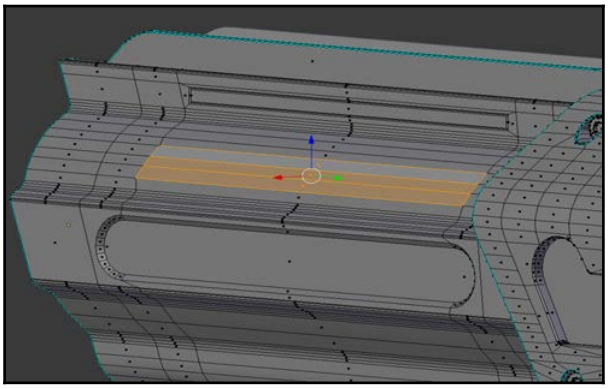
A great way to add detail (without too much work) is to run loop cuts across the existing faces, and then scale these new edges in by pressing *Alt + S*. This gives the appearance of a *panel line* or cut between different the portions of an object:



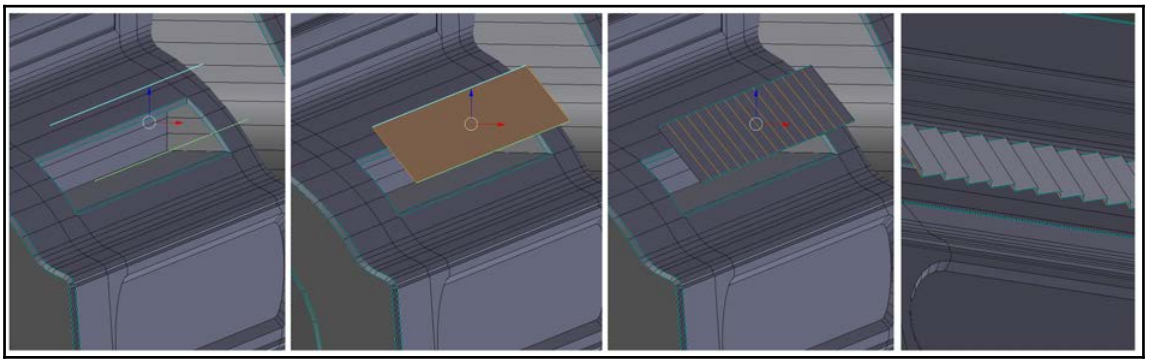
You can also create one such cut, and then rip the center edges. This will enable you to create a second line or cut that doesn't go all the way across the geometry:



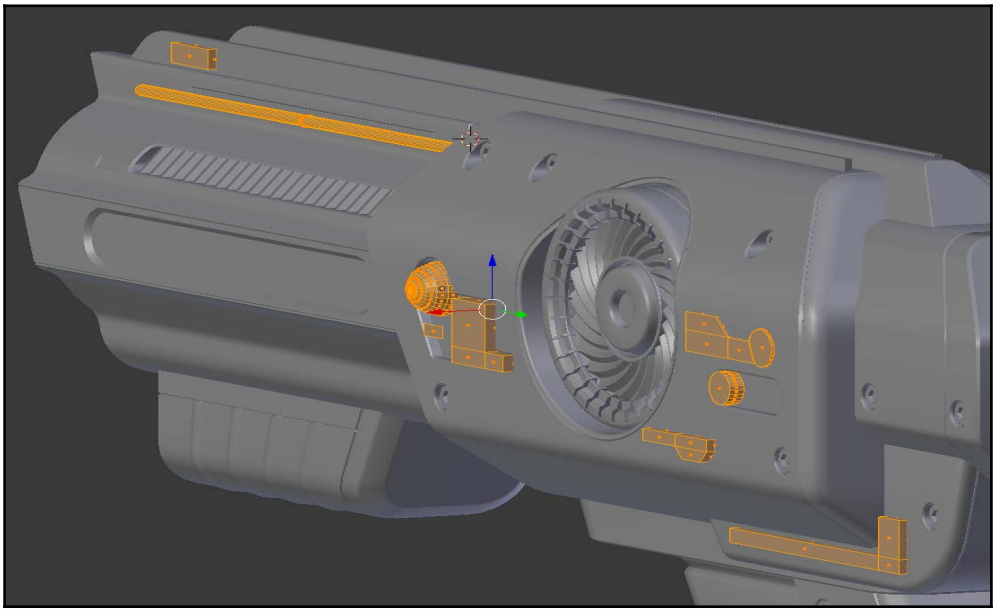
Here, we'll create some additional vents for fins by picking a series of faces on the upper barrel and eliminating them:



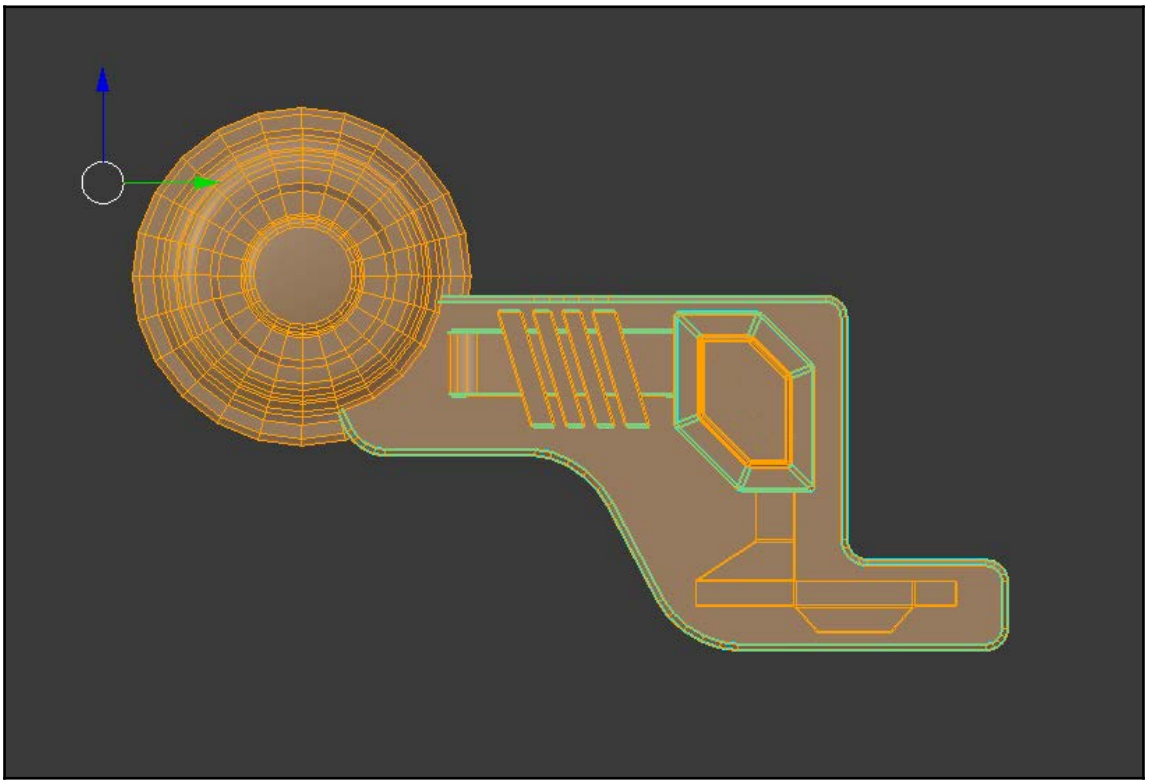
Then, we can pick the two edges at the top and bottom and fill in a face. With a series of loop cuts across the face, we can easily create our fins:



We'll continue the detailing by adding a series of basic shapes to our gun:



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:

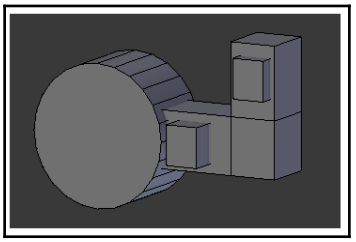


If you'd like, you can look at some real weapons for inspiration. Otherwise, feel free to get creative!

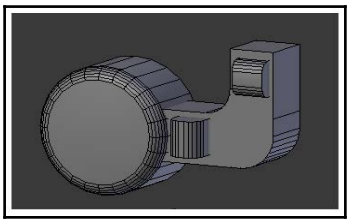
General workflow for detailing

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:

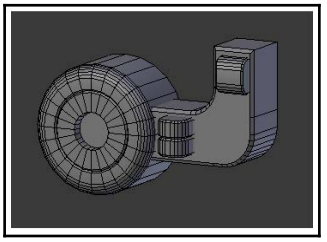
- Add the basic shapes.



- Do your “big” beveling to add curves.



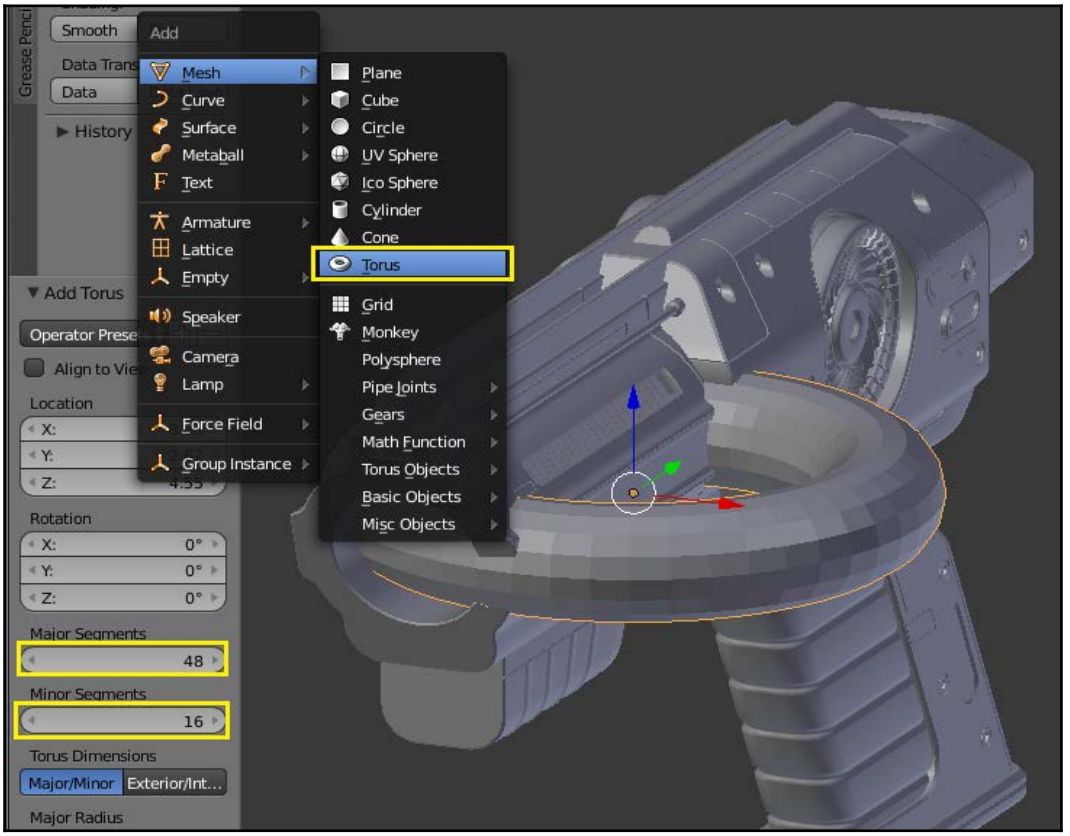
- Do your “small” beveling on the sharp edges.



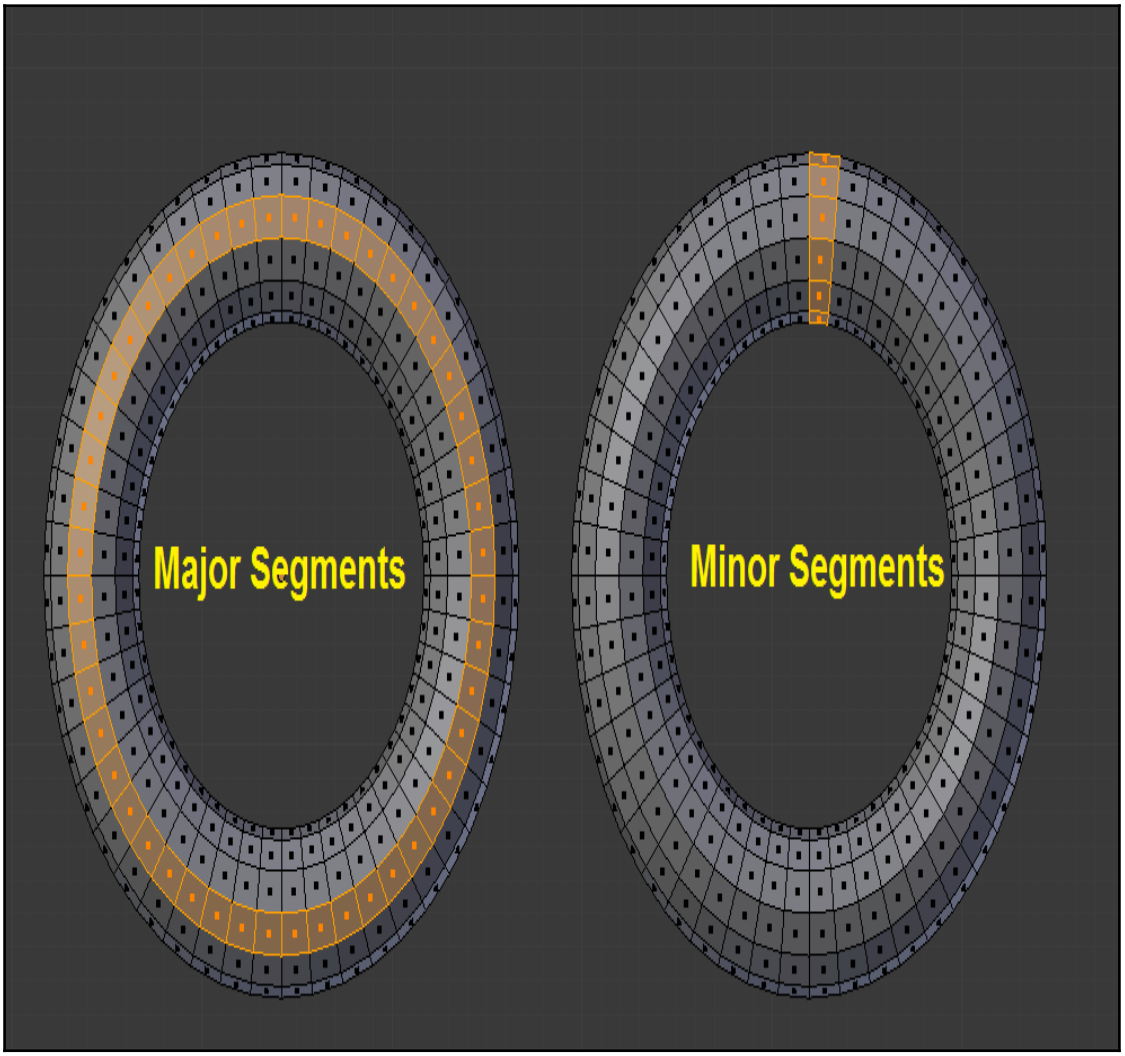
Next, we'll add some pipe-style detailing to our gun. Torus objects are fantastic for creating pipes.

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.

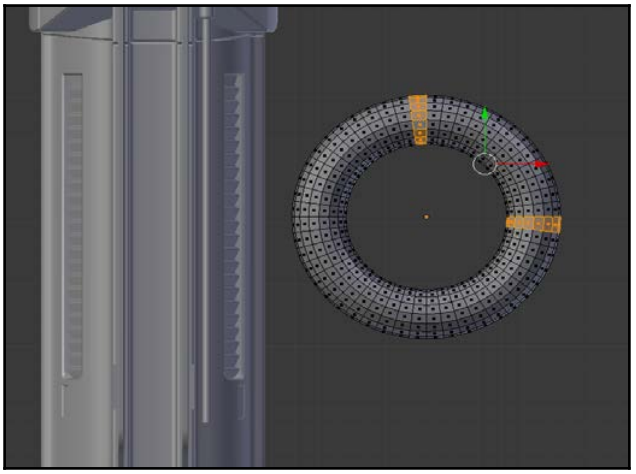
To do this, let's first add **Torus** to the scene. We'll use **48 Major Segments** (by default) and **16 Minor Segments**.



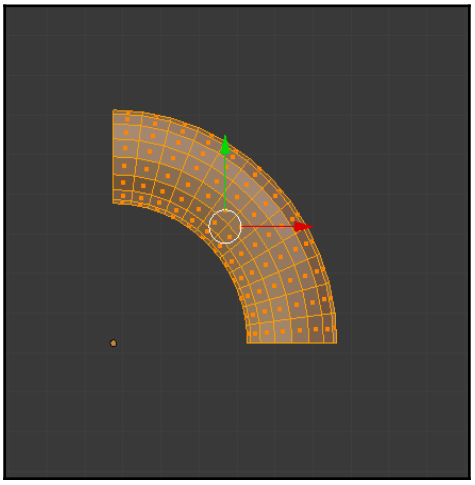
Major **Segments** is the numberof edges around the larger circle. **Minor Segments** is the number of edges around the *tube* that makes up the torus.



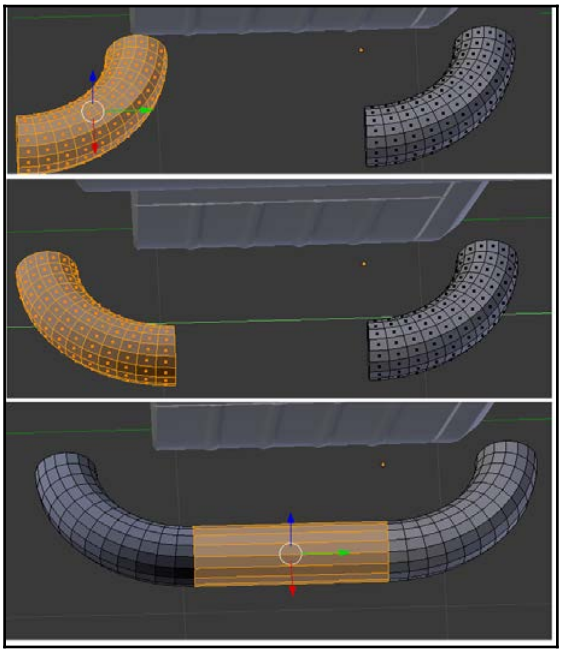
Let's start by deleting three quarters of the torus:



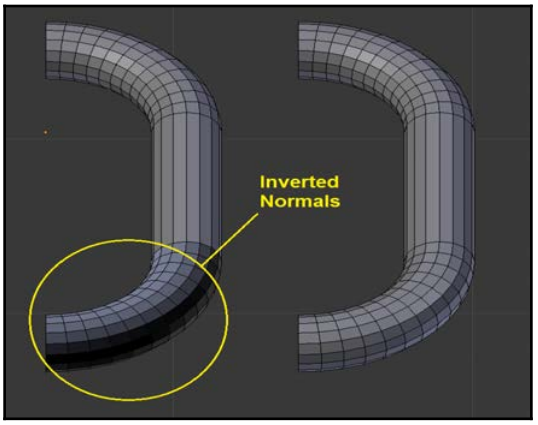
This will leave us with the small section that we need:



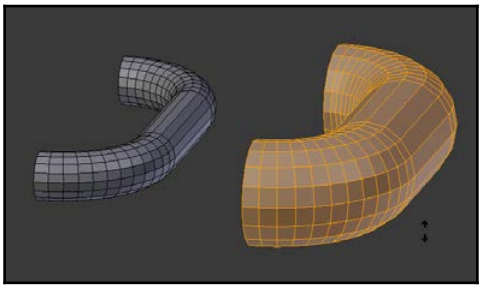
You can then duplicate this and flip it by pressing *Ctrl* + *M*. Then, you can fill in the faces:



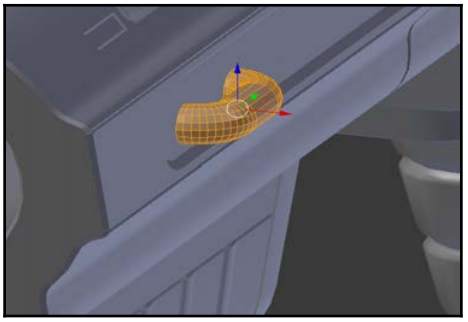
The only downside to doing this is that when you **Mirror** an object (*Ctrl* + *M*), you invert the normals. Go ahead and fix them by selecting everything and pressing *Ctrl* + *N:*



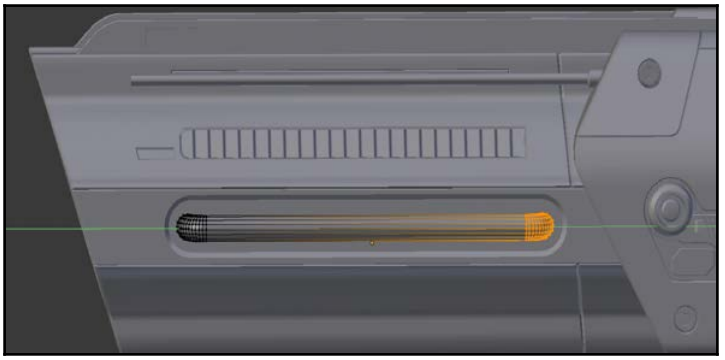
You can also change the width of your torus by scaling along the face normals (*Alt* + *S*):



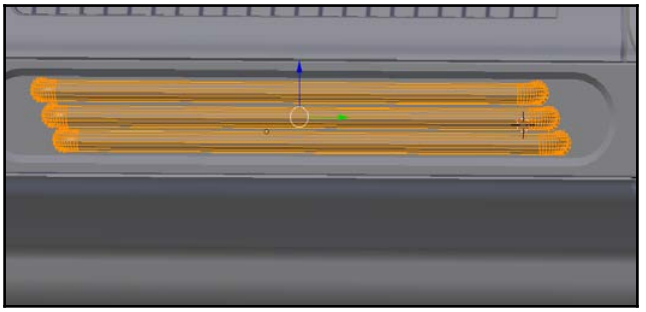
When you have your torus the way you want it, go ahead and move it into position:



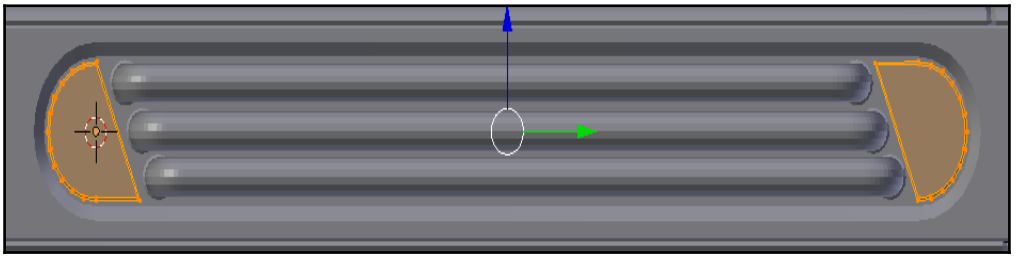
Then, you can select the back half of it and drag it back along the *Y* axis (or whichever is appropriate):



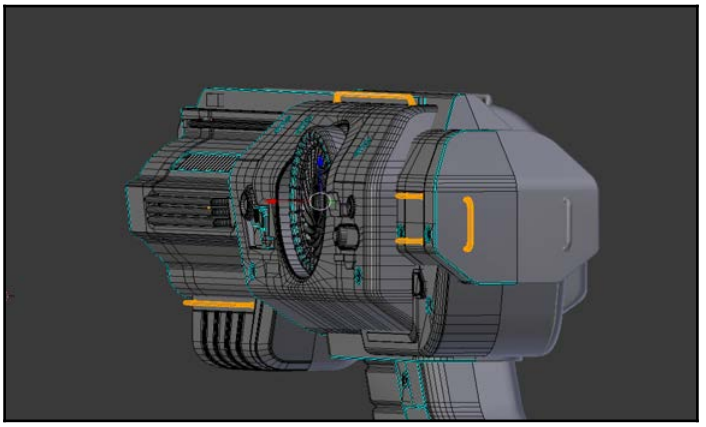
If you'd like, you can add two or more pipes…this is purely a stylistic choice:



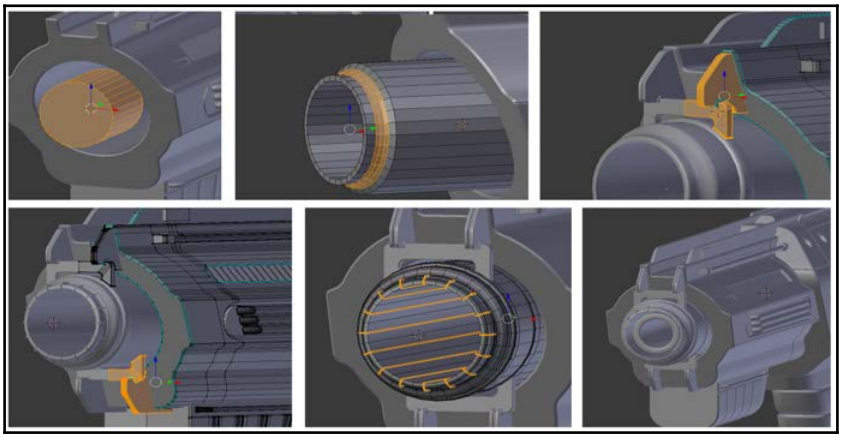
You can also add some detail to the ends of your pipe:



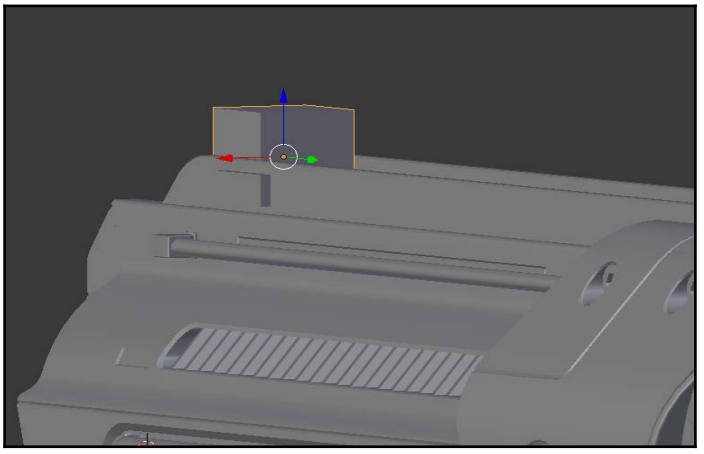
Once you've created one pipe section, don't hesitate to duplicate it, change the diameter by pressing *Alt* + *S*, and move it to other the areas of your model:



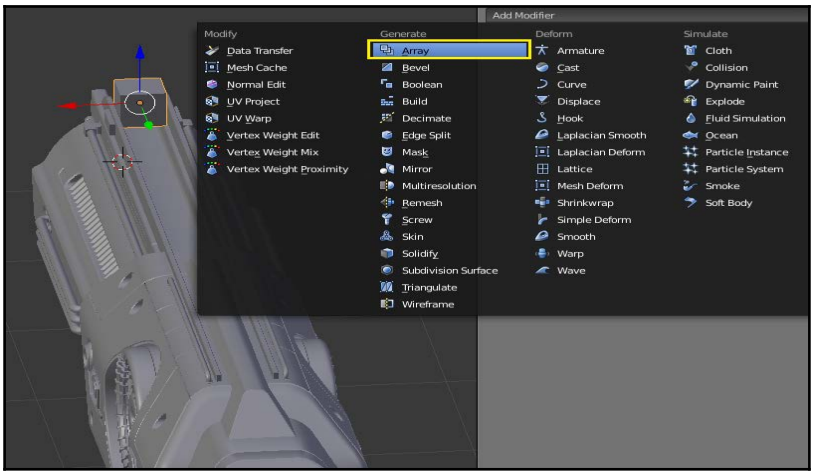
Next, we'll add some detail to our gun's barrel. We already have a nice circular hole in the front of the gun, so we'll just fill it in with a cylinder. Then, using techniques previously discussed, we'll add a bit of detail to the front of the gun:



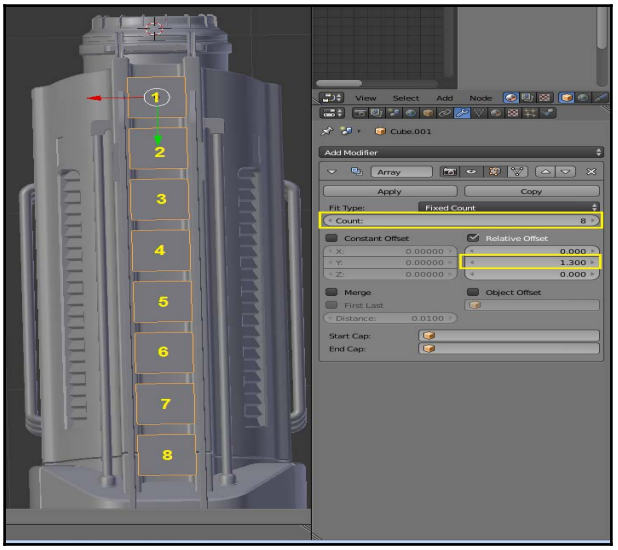
Then next thing we'll do is add some detail to the top with the **Array** modifier. First, we'll add a cube at the top front of the barrel:



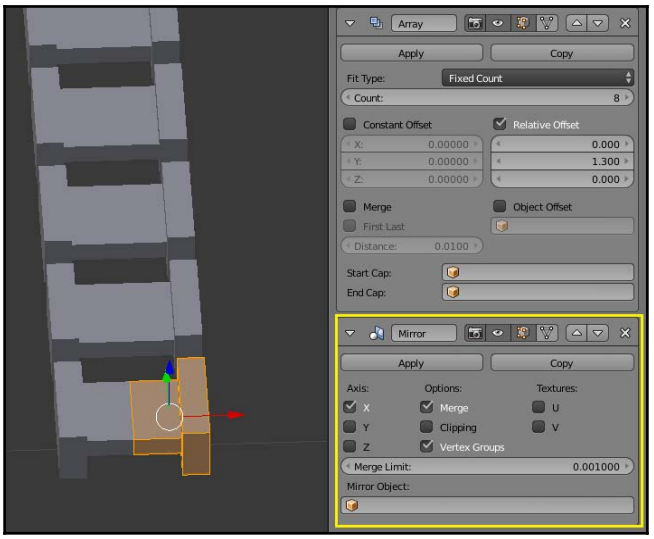
Then, we'll add the **Array** modifier to it. The **Array** modifier creates multiple copies of an object in a line (or "array") with an offset in position (that you specify):



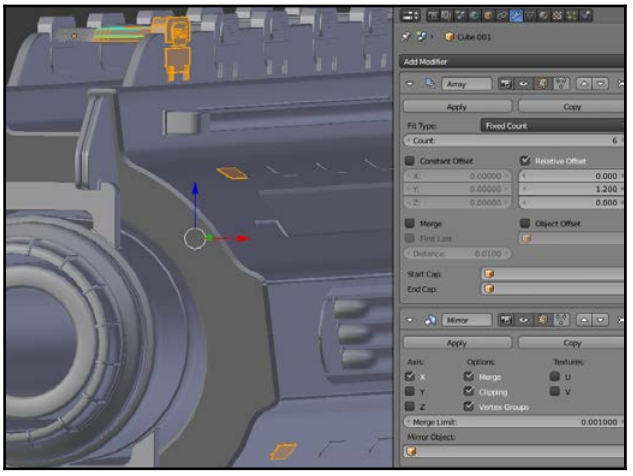
Adjust the **Count** and **Relative Offset** until you're happy with the result. You may need to change these settings as you begin to add detail to the arrayed object:



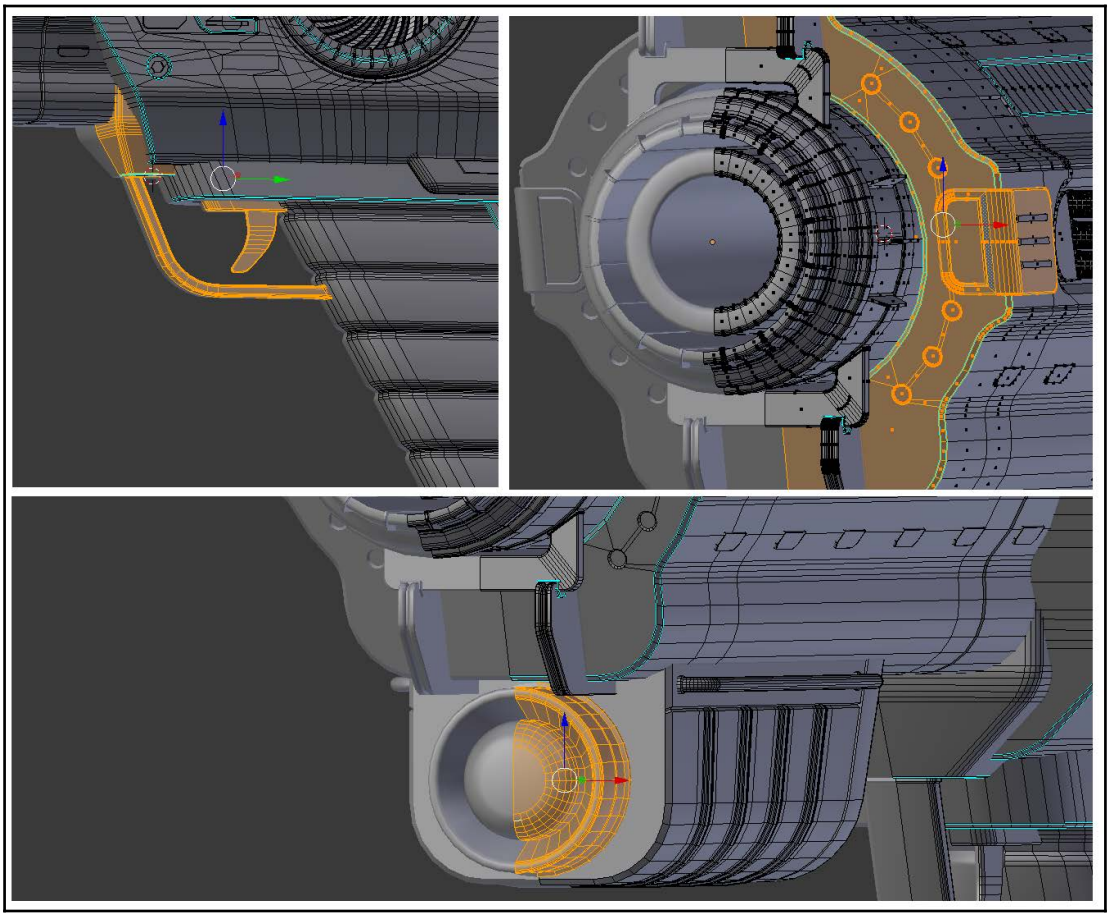
Of course, you can also delete half of your arrayed object and add a **Mirror** modifier to it as well:



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:



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.



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.

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:



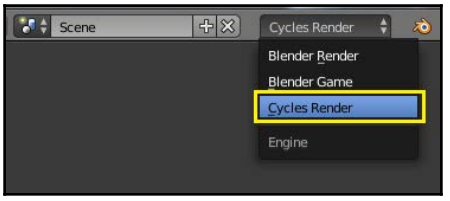
Of course, this is just what my gun looks like. Once we're done with this chapter, we will have no trouble adjusting various materials and settings to make it look however we'd like!

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.



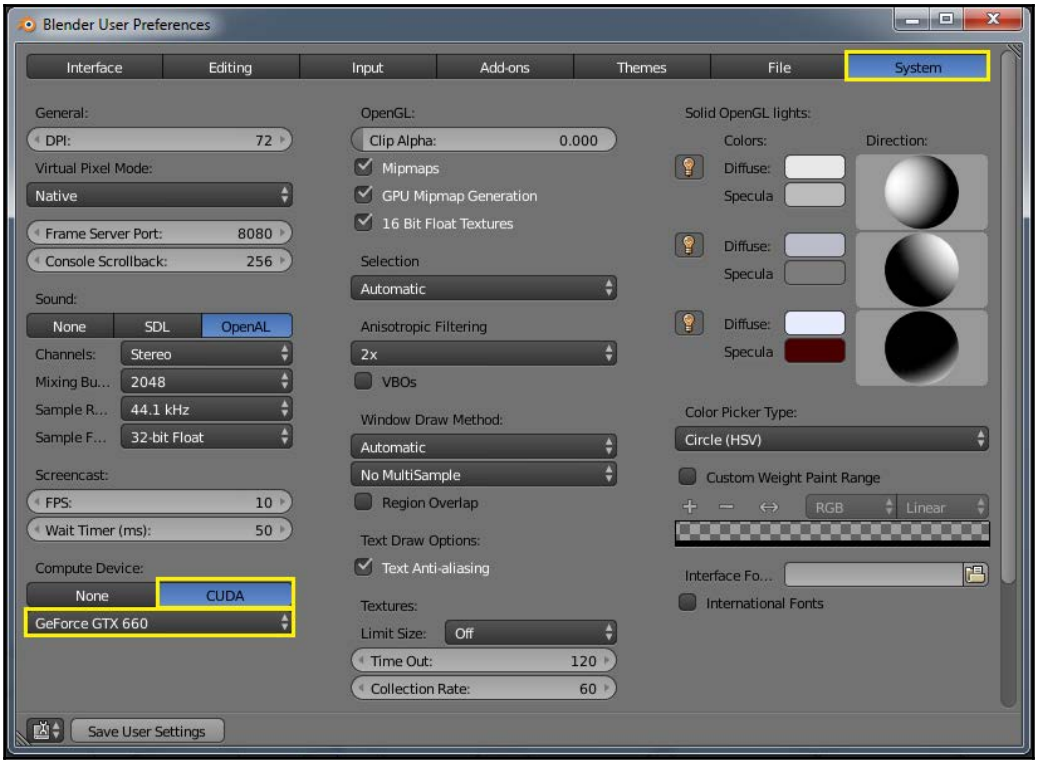
The first thing we need to do is make sure that we're using the Cycles Render engine. This can be selected from the drop-down menu at the top of your screen:



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.



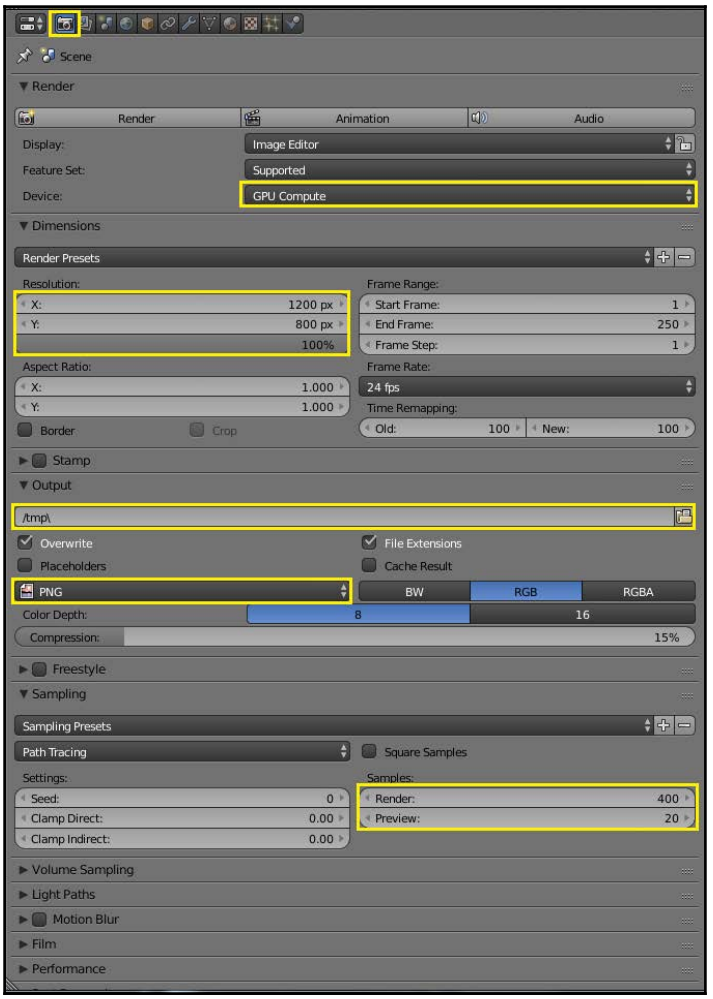
You can also set your **Resolution** at this point. This is how big the rendered image will be. The bigger the image, the longer it will take to render.

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.

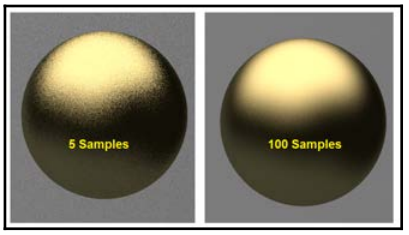
The **Preview** setting tells Blender how many samples to use in the 3D window when you're in the **Render Preview** mode (more on that in a minute). For now, we'll change these settings to **400** and **20** respectively.



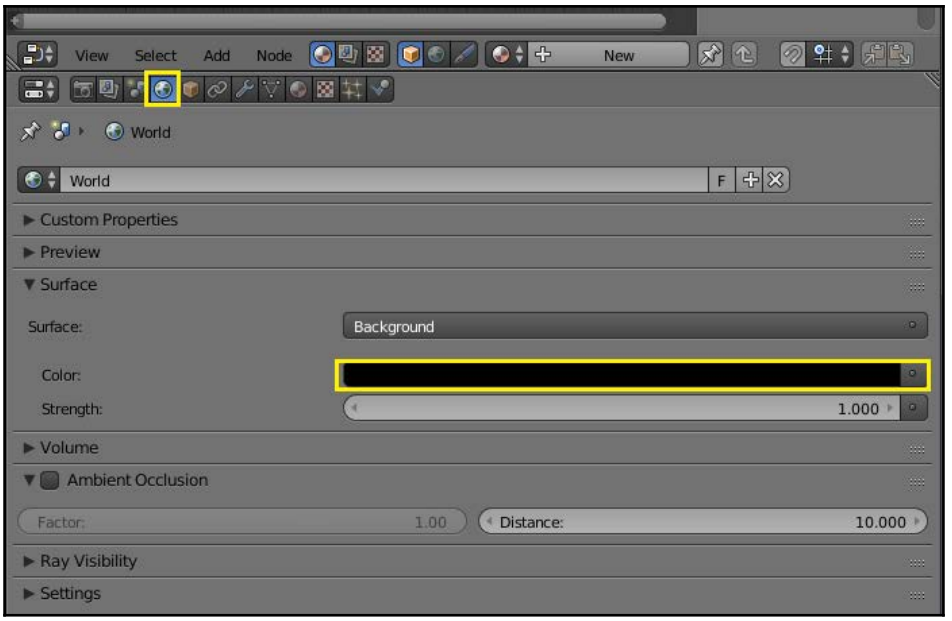
Sampling in Cycles

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:

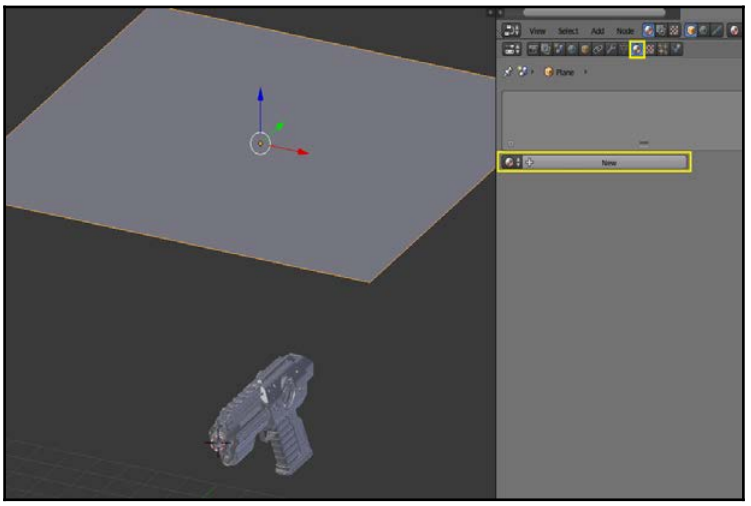




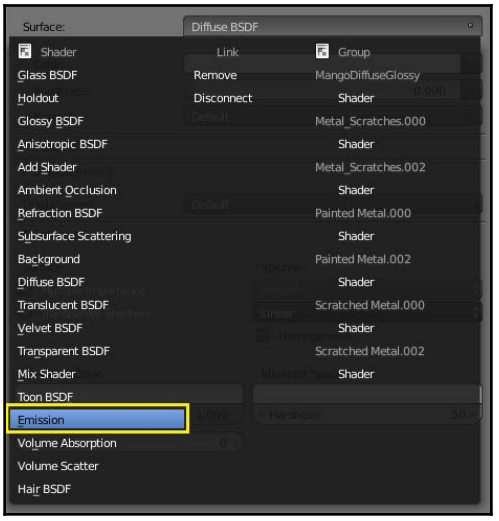
Next, let's go into our **World** tab and turn our ambient light all the way down. Do this by either changing the **Color** to black or the **Strength** to 0.



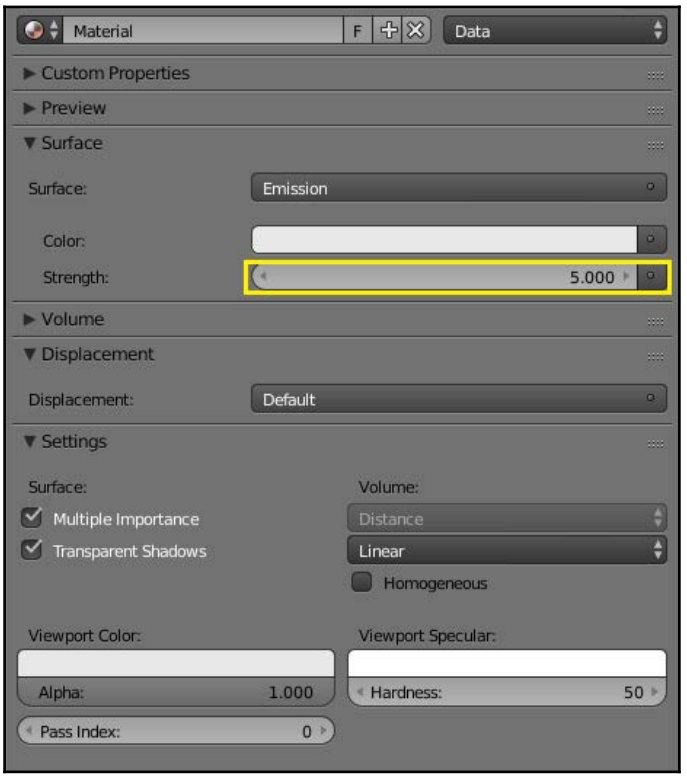
Next, we'll add a basic plane in our scene to function as the overhead light. Then we'll add a new material to it under the **Materials** tab:



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.

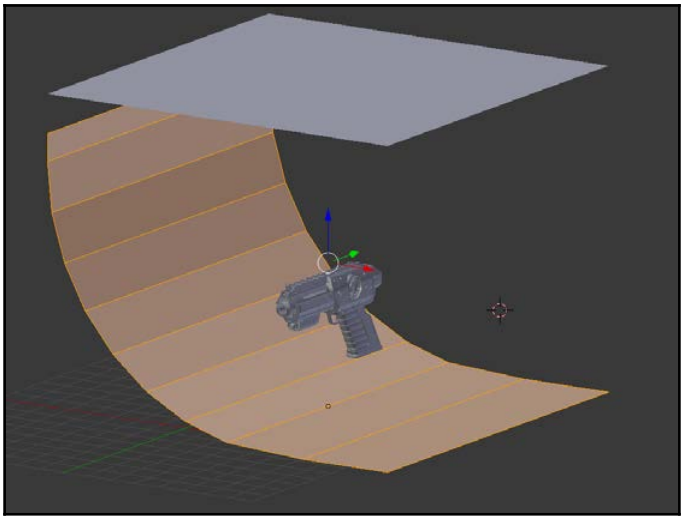
If you'd like to quickly switch back to **Solid Shading**, which is the default view, you can press the *Z* key twice or *Alt* + *Z*.



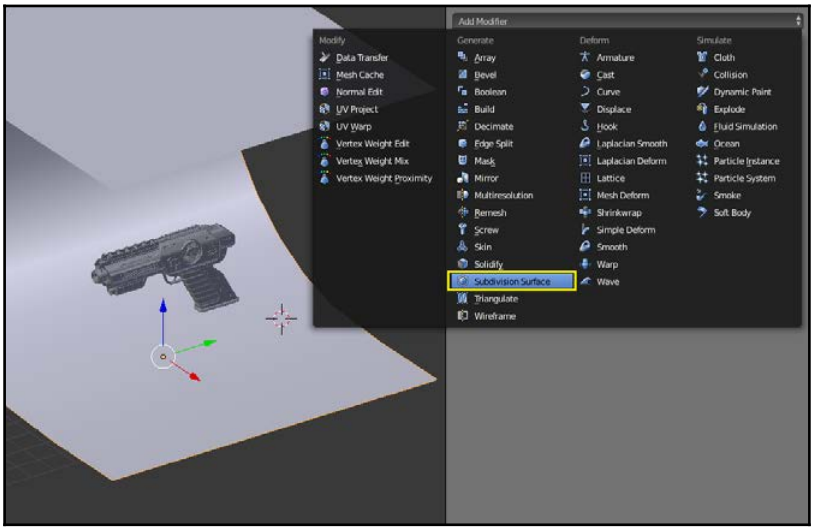
This only works to switch back to solid shading from something else.



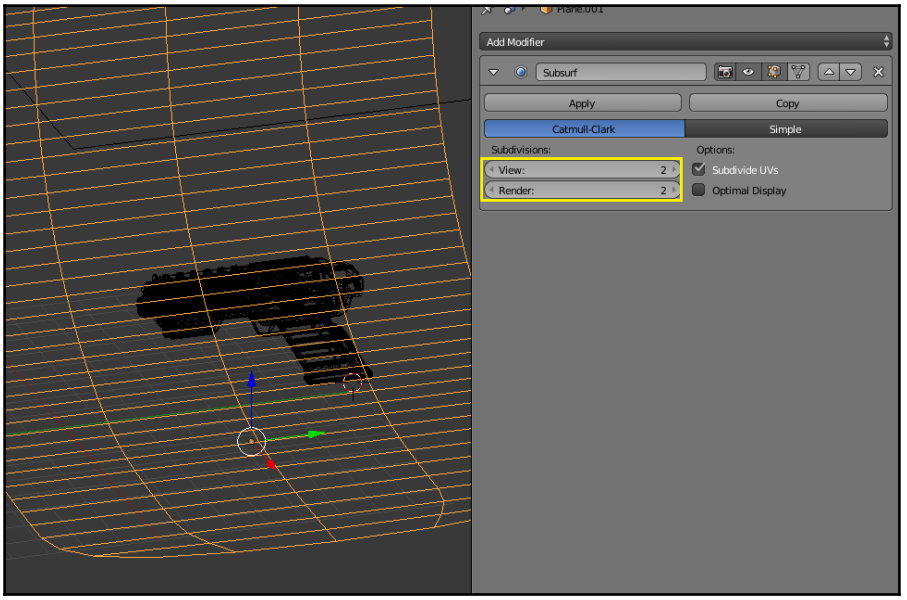
Next, we'll add a curved background plane. You can add this using any of the techniques that we covered in the previous chapter. It can be a portion of a cylinder, a part of a beveled cube, or so on.



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.

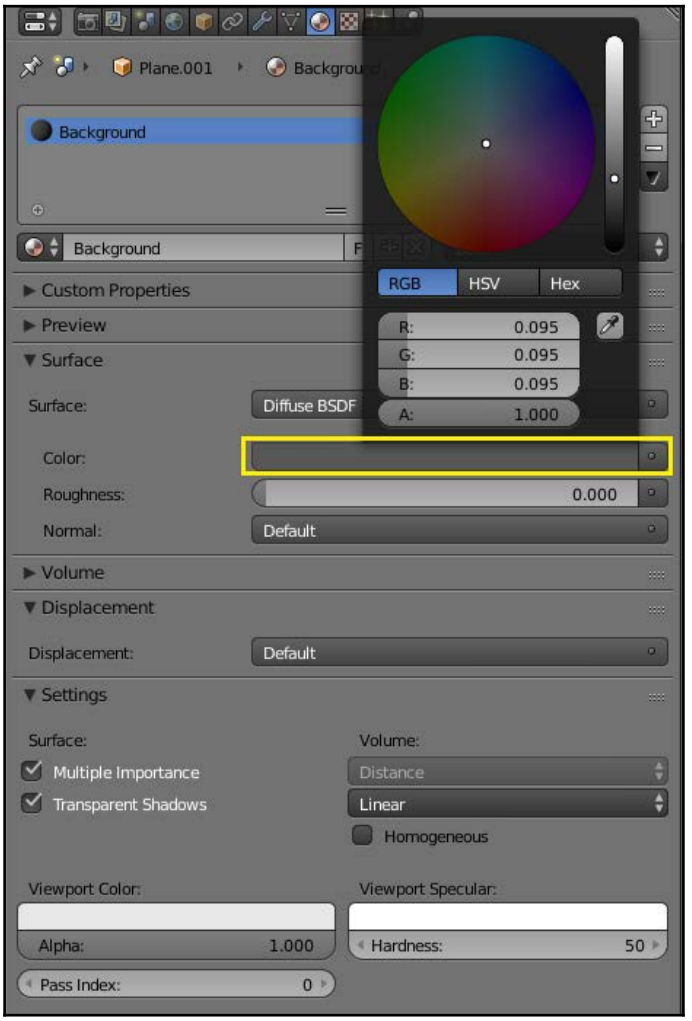


When you toggle into Wireframe mode (press the *Z* key once), you can see what the **Subdivision Surface** modifier has done for you.

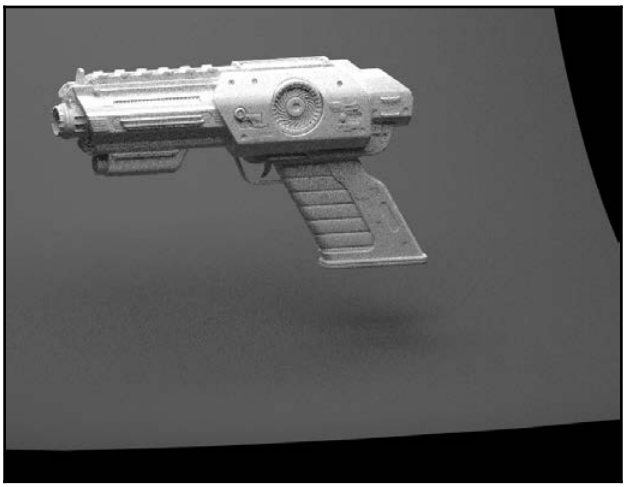


Let's add a basic **Diffuse** material to our background plane. A **Diffuse** material is

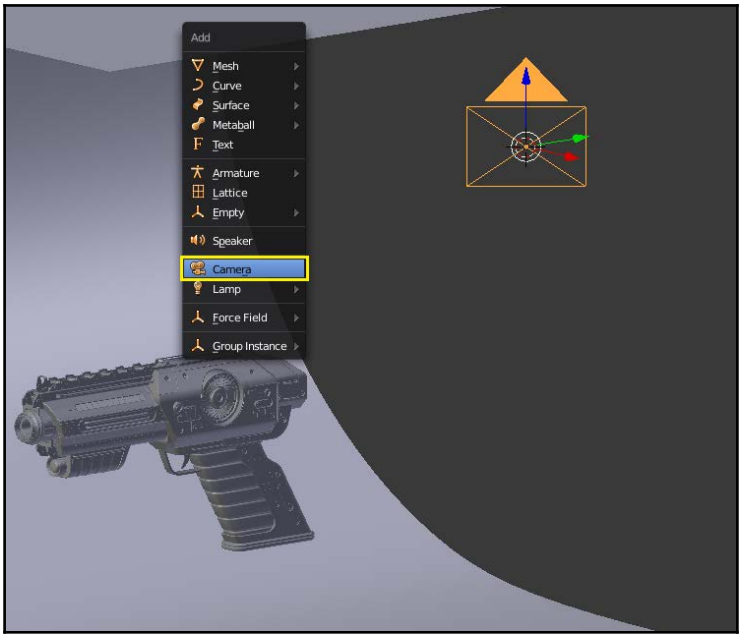
essentially a flat (not shiny) material that gets its name by diffusing light. We'll take a look at this in a minute.



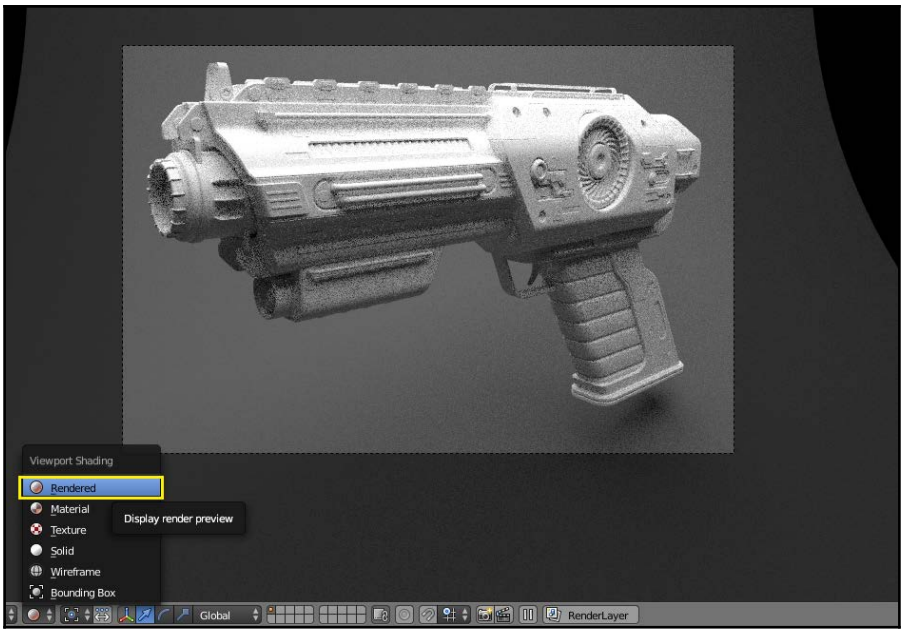
Take another look at our rendered preview to see how the background plane looks.



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:



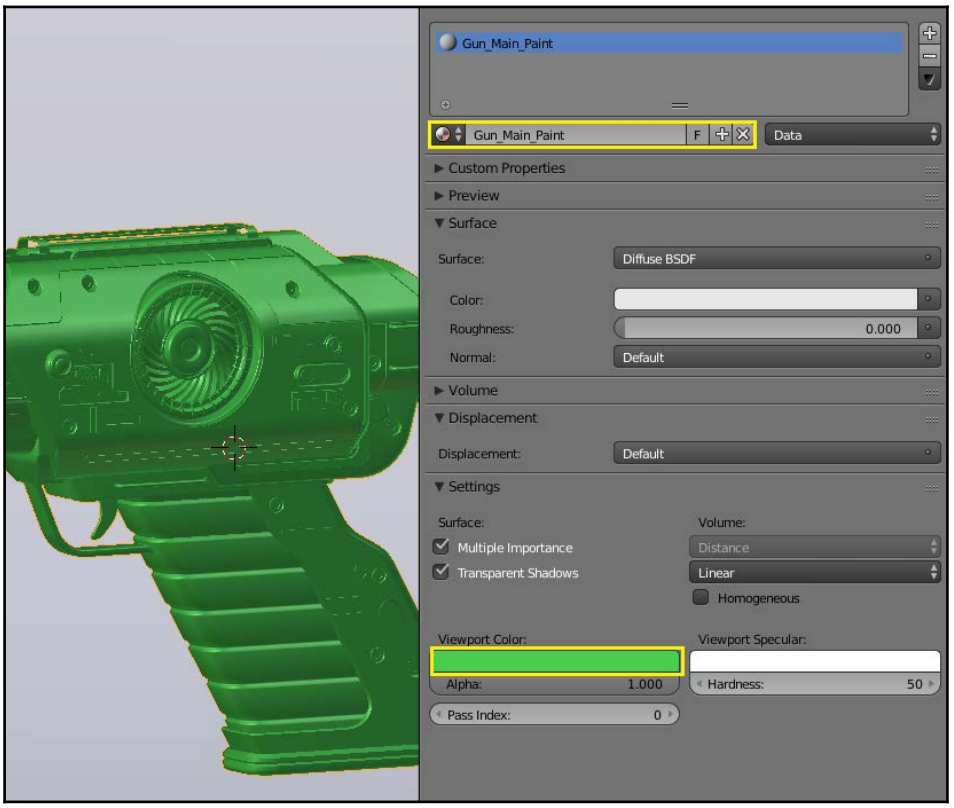
Adding materials and textures

Now, it's time to add some materials to our gun!

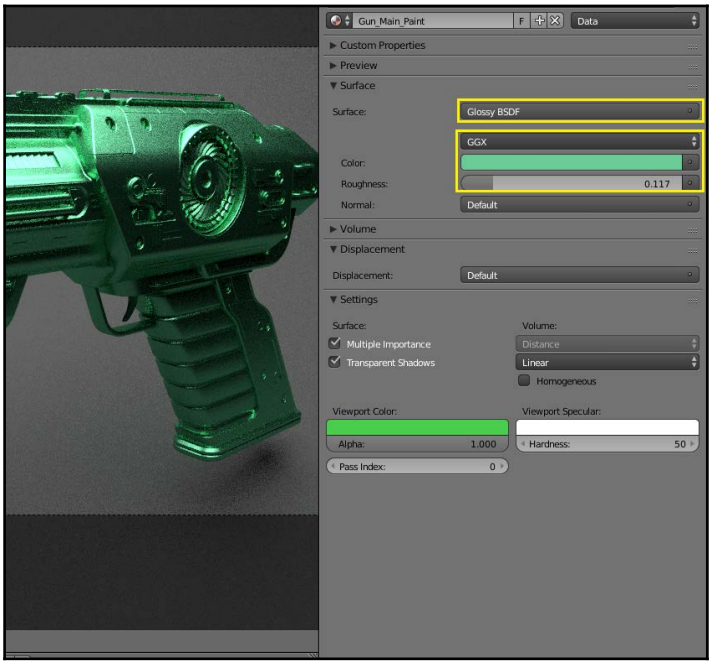
We'll add a basic one first and call in **Gun_Main_Paint**. One thing we want to do is adjust the **Viewport Color** setting so that we can see which materials are assigned to the various parts of our model. Because it's the only material on our gun right now, the entire gun will change its color to match what you've selected.



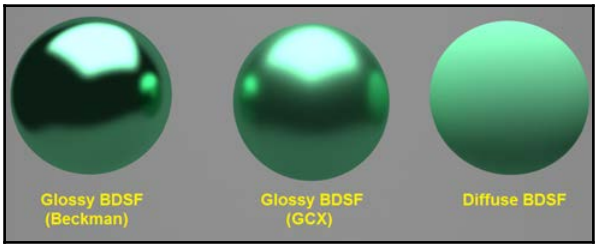
Changing the **Viewport Color** does not affect the actual color of the material (at the render time). It simply changes what shows up in your 3D window.



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.



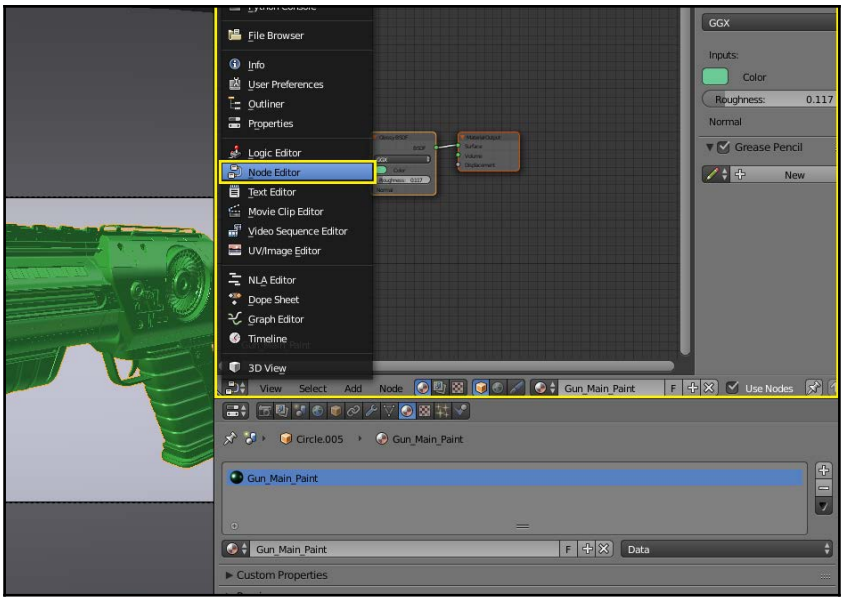
You also have the option to switch between the **Beckman**, **GCX**, **Sharp**, and **Ashikhmin-Shirley** types of shading. Each of these reflects light somewhat differently and you must feel free to experiment. For this model, however, we'll probably only need Beckman and/or GCX shading.



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.

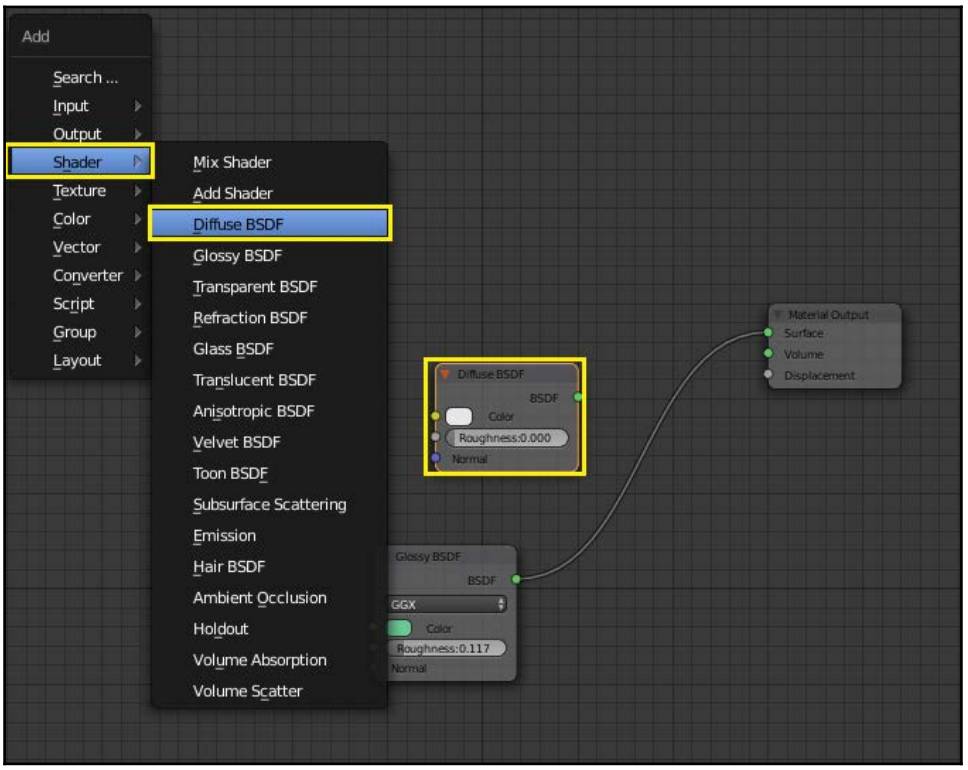
We'll open up **Node Editor** in Blender. It's easiest if you have it set up right above your **Properties panel** and next to your main 3D window. Of course, this is really a personal preference.



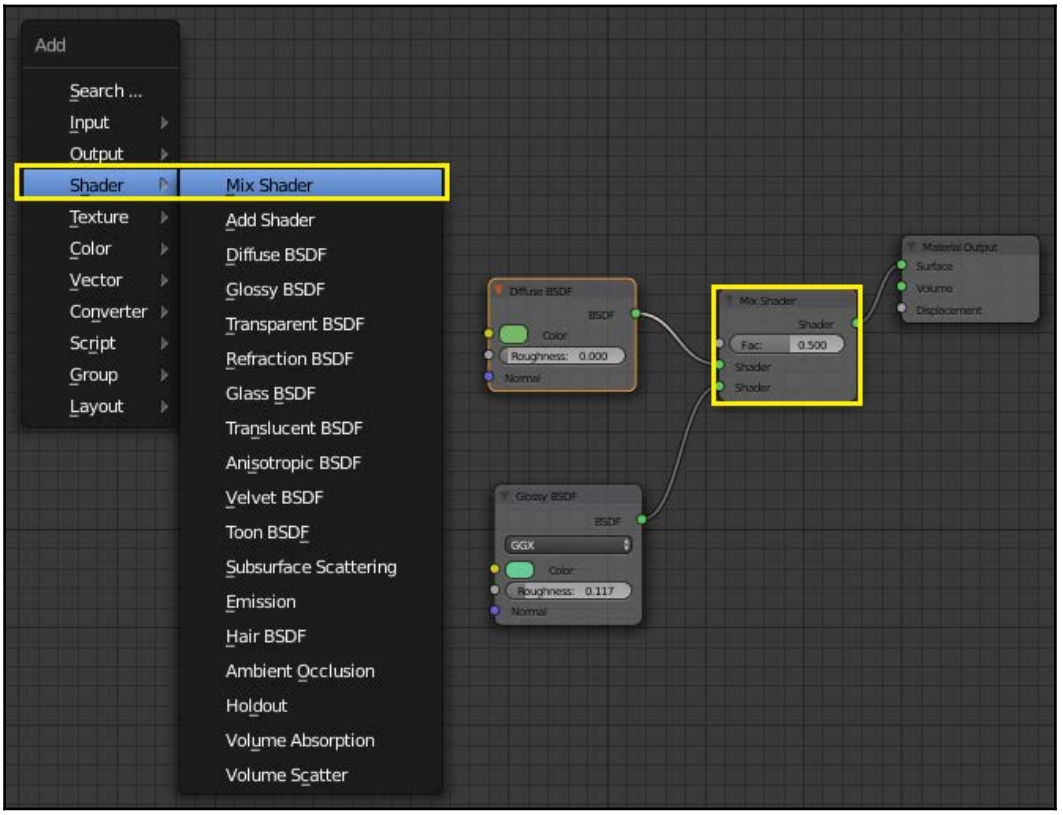
By looking at our **Node Editor** window, we can see that the **Glossy** shader that we selected is connected directly to the material's **Surface Output**.

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.

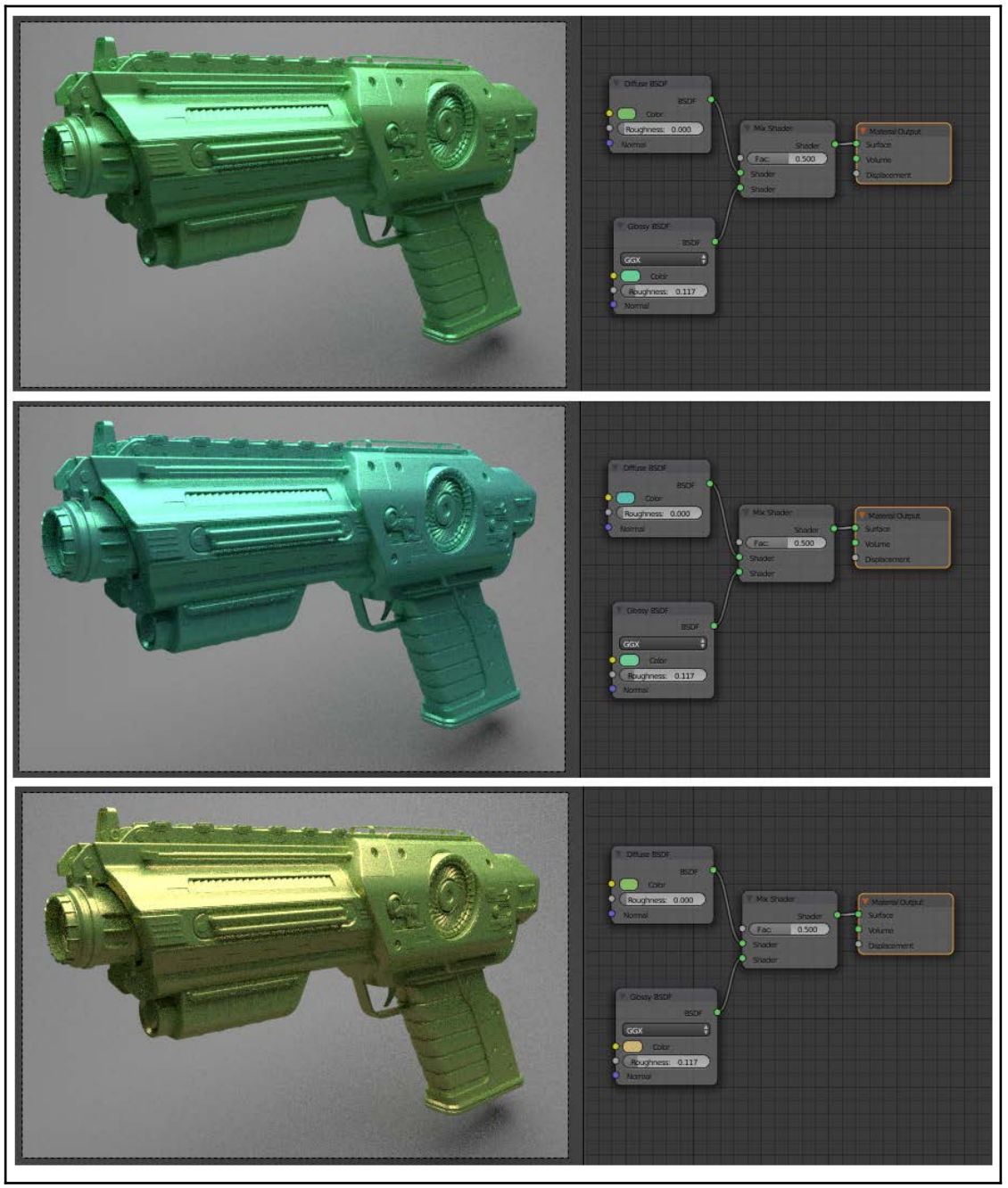


You can adjust the color of that shader to match the glossy shader that we already have. In order to mix these two, we'll next need to add a **Mix Shader**.



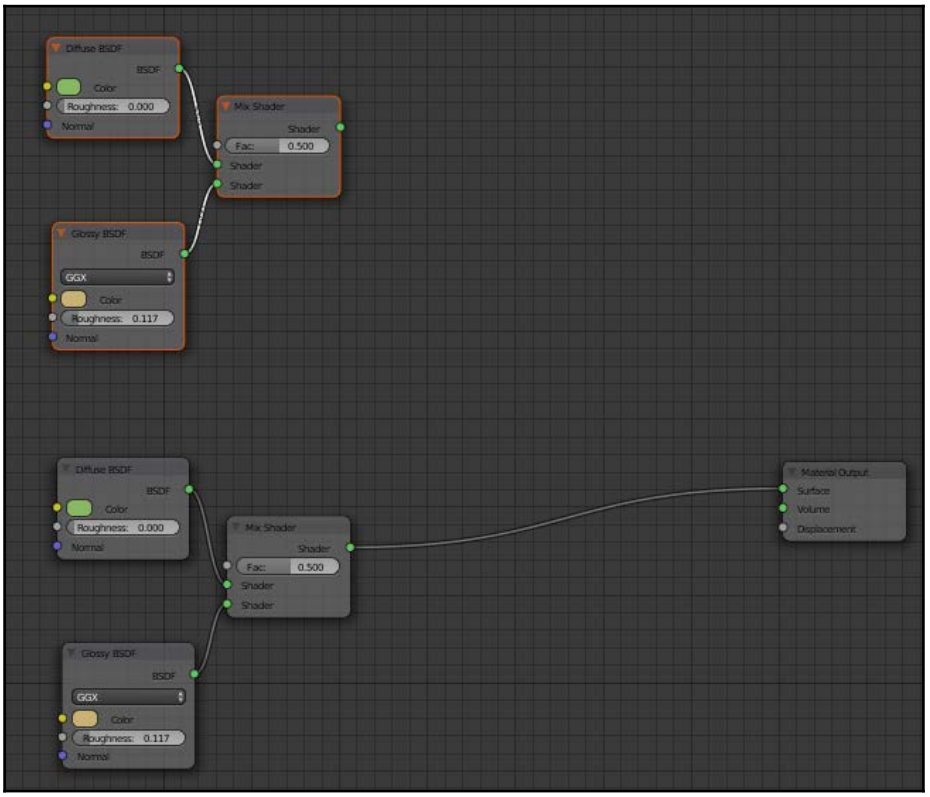
Connect the outputs of both the **Glossy** and **Diffuse** shaders to the **Mix Shader**. Then, connect the **Mix Shader** output to the **Surface Slot** of the **Material Output**.

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.

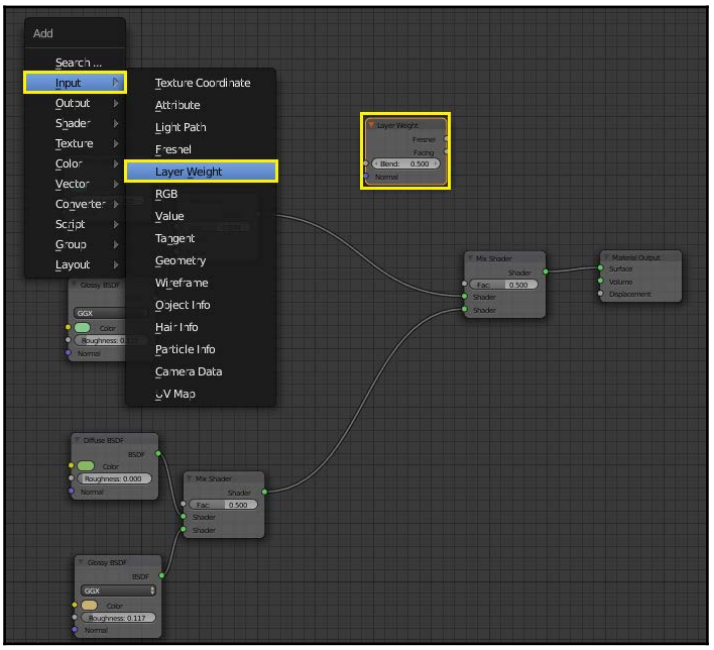

Diffuse BSDF
BSDF
Color
Roughness: 0.000
Normal
Mix Shader
Shader
Fac: 0.500
Shader
Shader
Material Output
Surface
Volume
Displacement
Glossy BSDF
BSDF
GGX
Color
Roughness: 0.117
Normal

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.

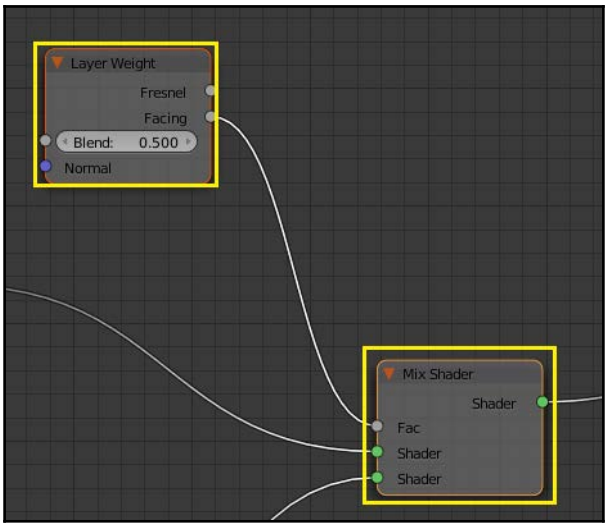
Go ahead and make a few changes to your duplicate set of nodes. Change the color or roughness a bit. We just want to make it a bit different from the first set.



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**.

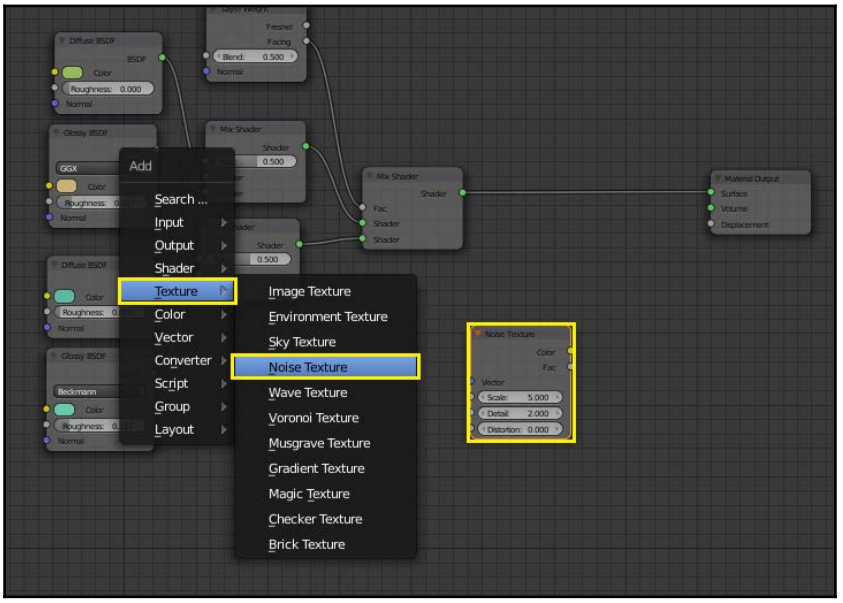


Since we're using solid (metal/ plastic) types of materials, it's best to use the **Facing** output of the Layer Weight shader. If you have a material where light penetrates the surface (such as glass or translucent plastic), you can also experiment with the Fresnel tab. For now, we'll stick with **Facing**.

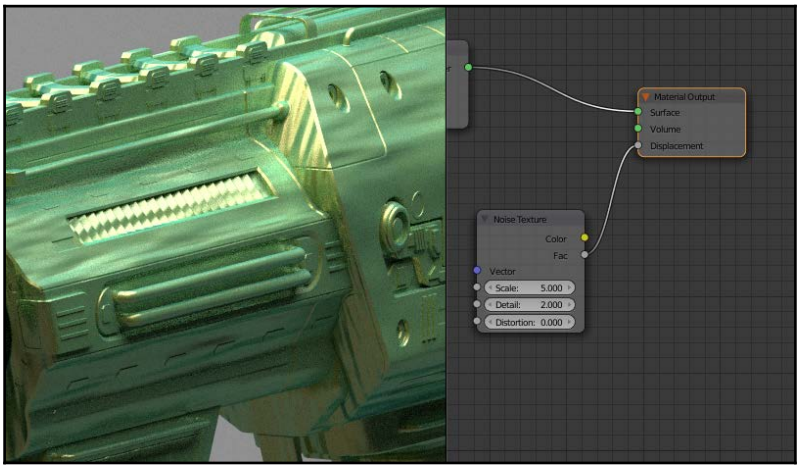
Now our gun has a nice fade between its different colors.



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.

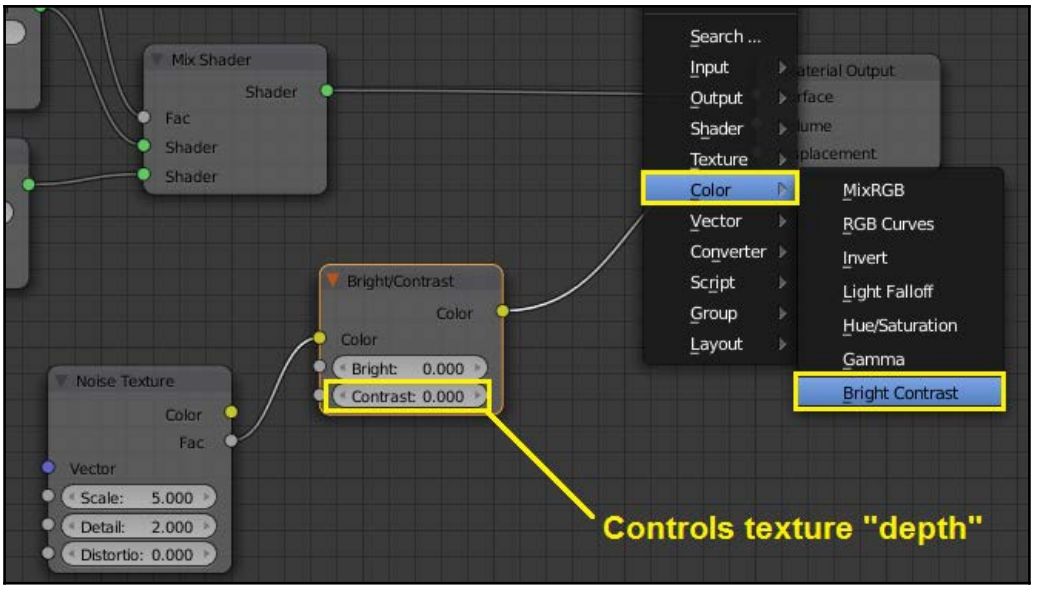


When you plug this node into the **Displacement** input of our **Material Output** node, you can see that the surface of the model is affected:



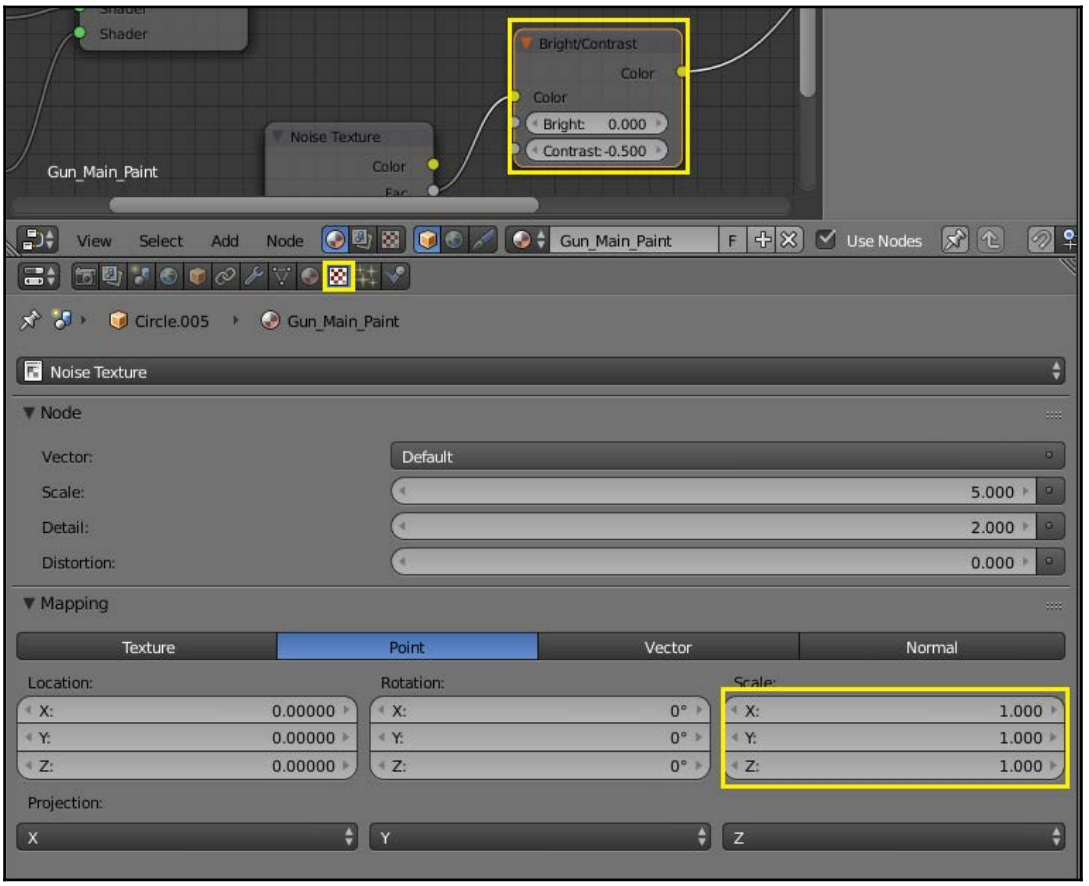
It appears to have some *hills and valleys* to it. Note that this does not change the mesh at all; it's just a material's effect.

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.

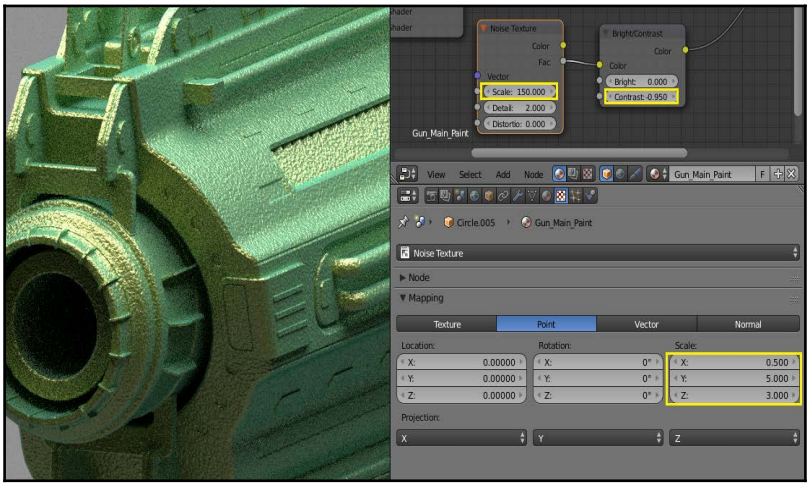


We also need to change the **Mapping** of the texture. Because our gun is longer than it is tall, (and taller than it is wide), the texture appears stretched. In order to fix this, we'll select the **Texture** node and move it to the **Textures** tab of our **Properties** panel.

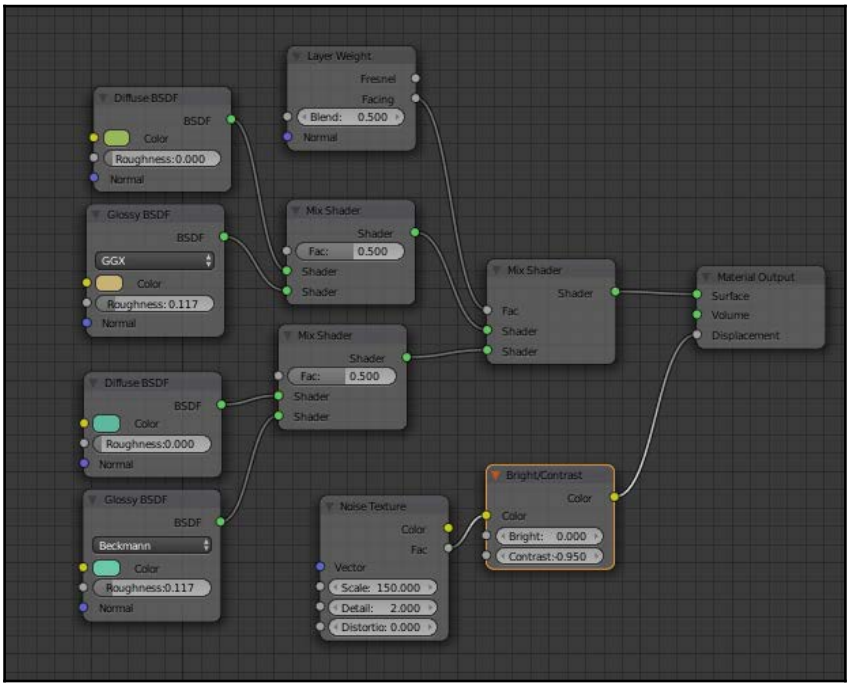
Here, you can see that the texture is being mapped to the gun evenly on all three axes. Let's adjust these numbers to reflect the different length, width, and height of our gun.



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.

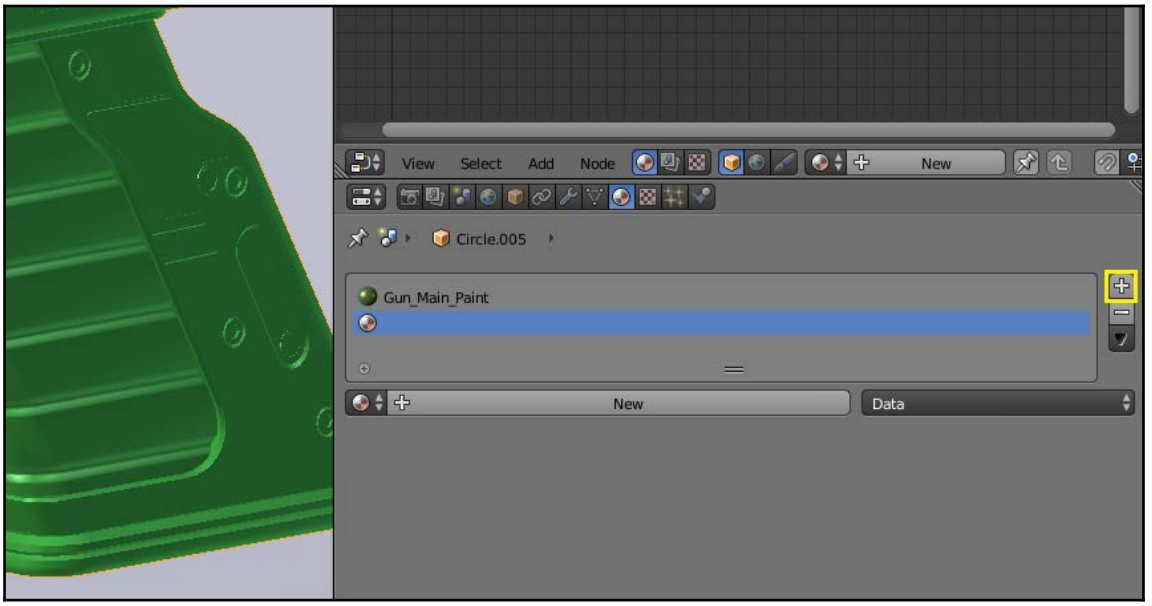


This looks pretty good for our basic paint material. Before we move on, here's what my material looks like in the **Node** editor:

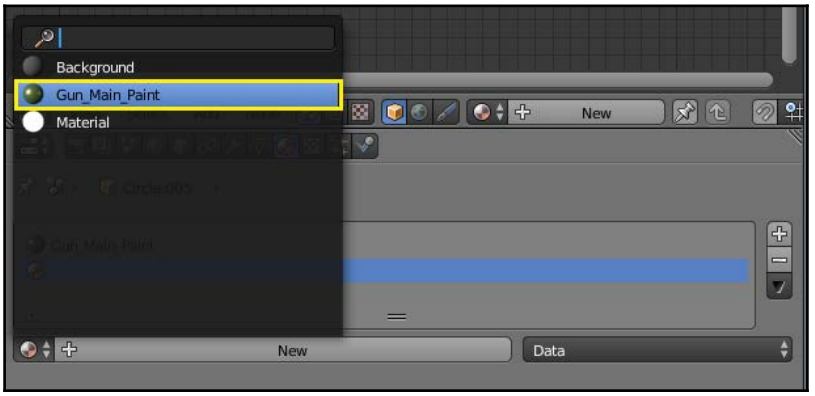


You can feel free to use these exact settings, or you can adjust them until they look good to you.

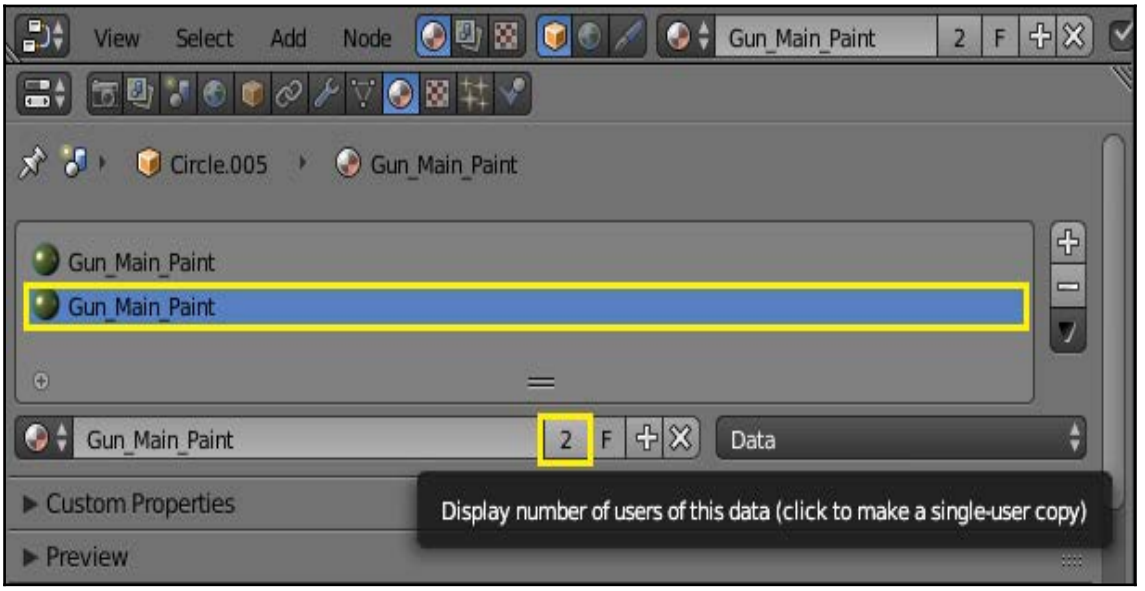
Next, we'll add a second material to our gun. This will look very similar to the first material, so we'll duplicate it and make a few adjustments. In order to do so, first add a new material to the gun with the small plus tab next to the materials list.



Then, select the **Gun_Main_Paint** material that we already have in place.

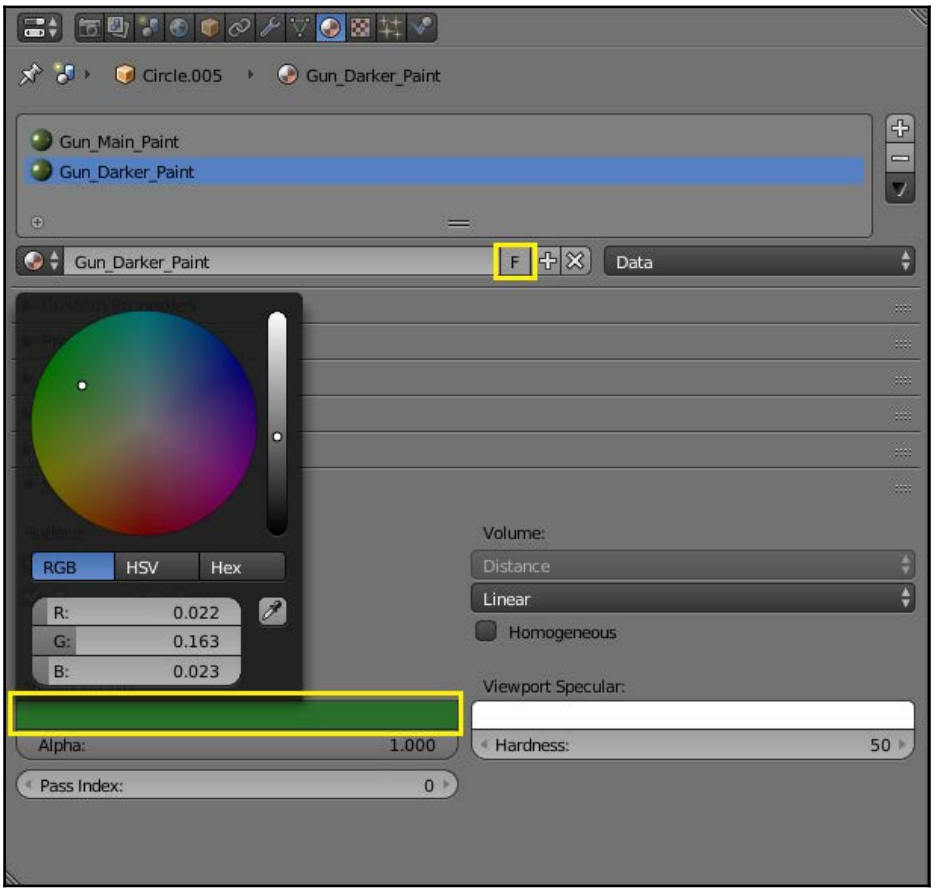


Because it's being used twice, you can see that the number **2** appears just below the materials list. We need to click on this to make it a **single-user**. In this way, we can adjust our second material without affecting the first one (right now, we just have two copies of the same one).

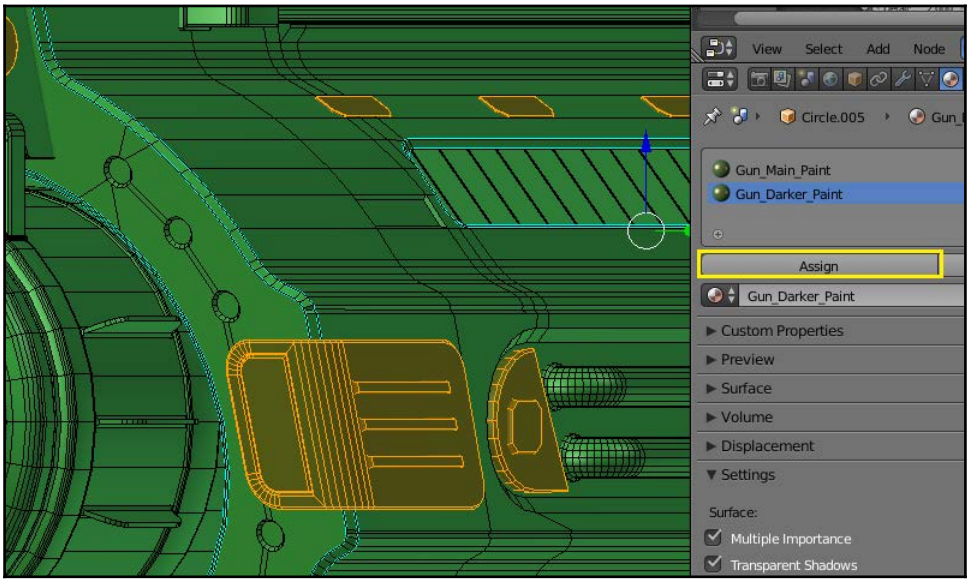


When you click this button, the letter **F** appears. You've now created a second material that's an exact copy of the first one.

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 you 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.

4

Spacecraft – Creating the Basic Shapes

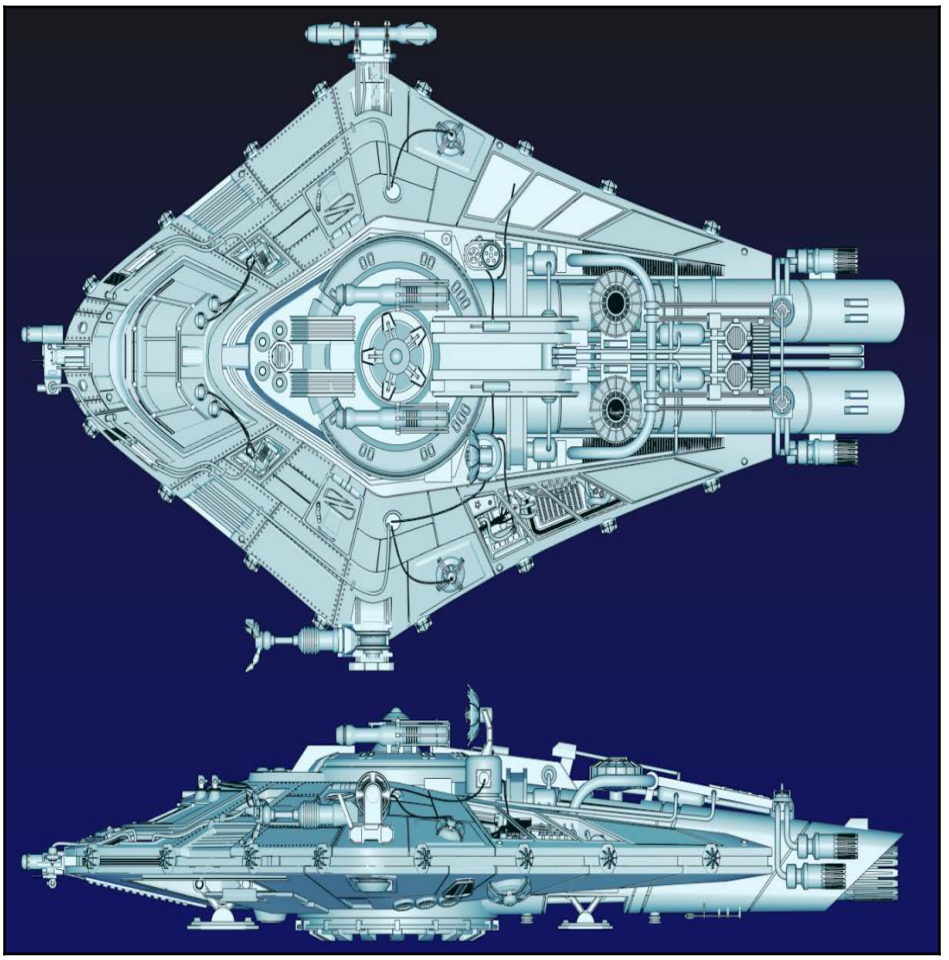
In this chapter, we'll get started on a detailed spacecraft. We'll lay a good foundation here by reviewing our goal, and then creating the basic shapes. That way, we can take it further in subsequent chapters. The following topics are covered in this chapter:

- 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!

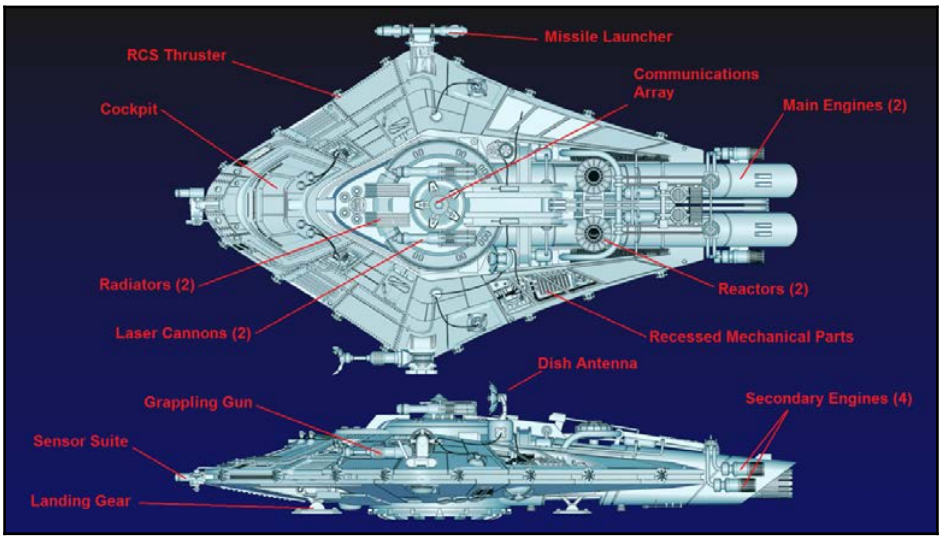
First, let's take a look at our design:



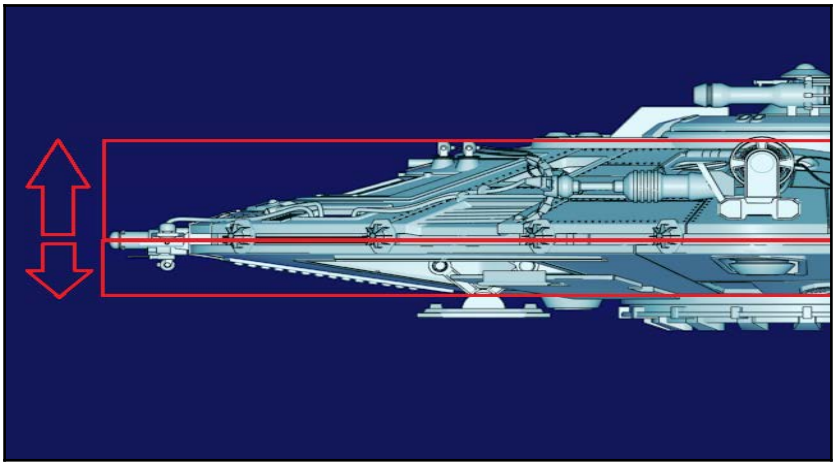
If this looks familiar, that's because it was one of my first major Blender projects. I released it under the Creative Commons Non-Attribution license, which means anyone can use the model (or the design) for whatever they want.

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.

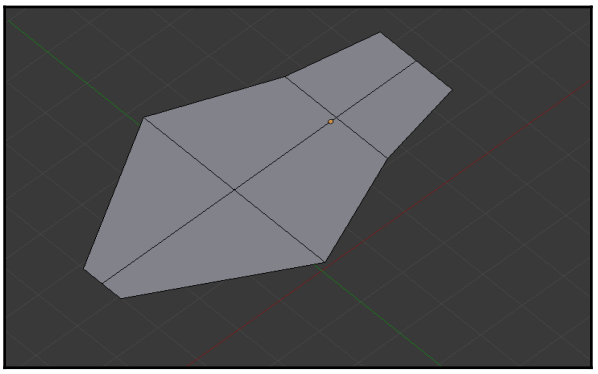
From the image, we can identify several key components and details that we'll want to pay attention to:



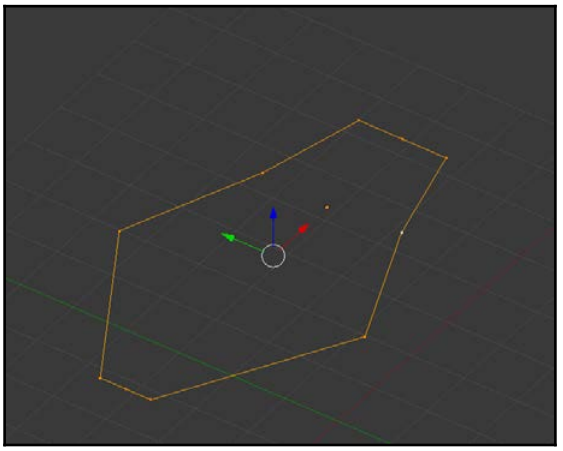
The first thing to focus on is the basic shape of the hull. You can see that the top and bottom sections are very similar, but not identical. The bottom section is significantly shorter than the top:



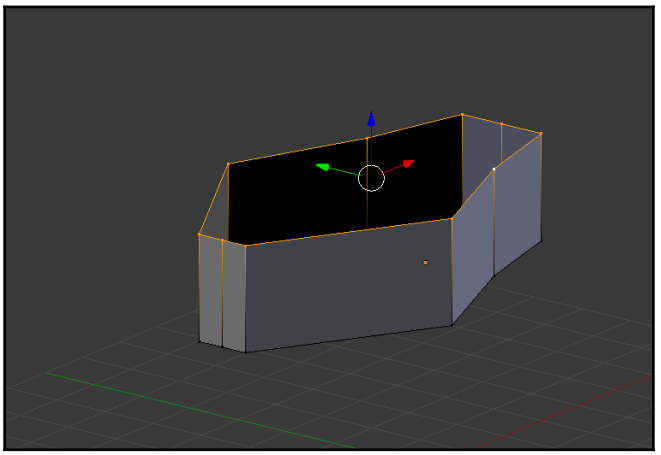
Rather than modeling them separately, it's probably easier to just create one section (the top, for example), and then copy and modify it as required. So, let's tackle that first. We'll open up a new scene in Blender and start with a basic plane. We'll then subdivide it twice with loop cuts, and move the vertices around to the approximate position that we want:



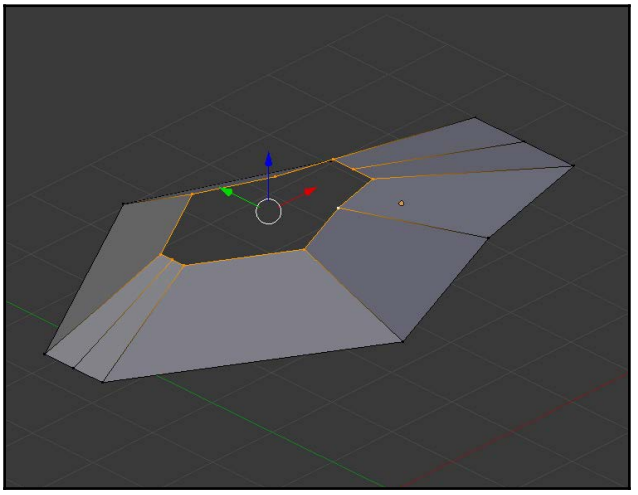
Next, we'll delete the inner vertices, leaving just a ring:



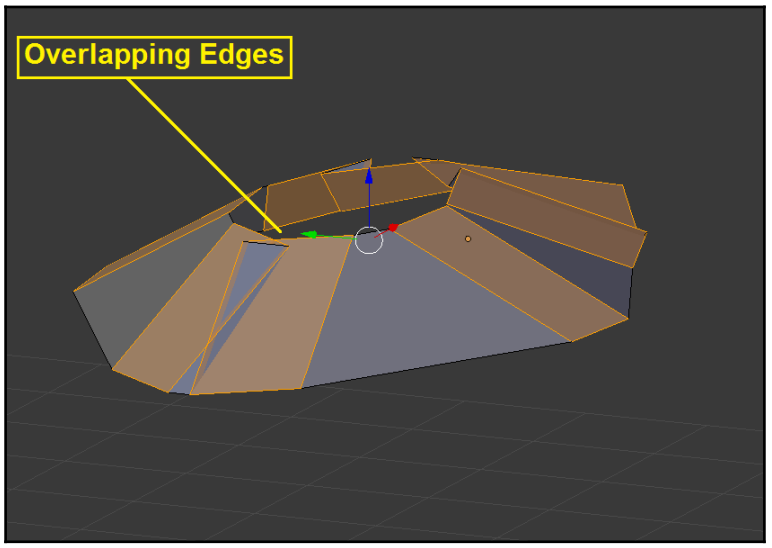
Then, we'll extrude those edges straight upwards:



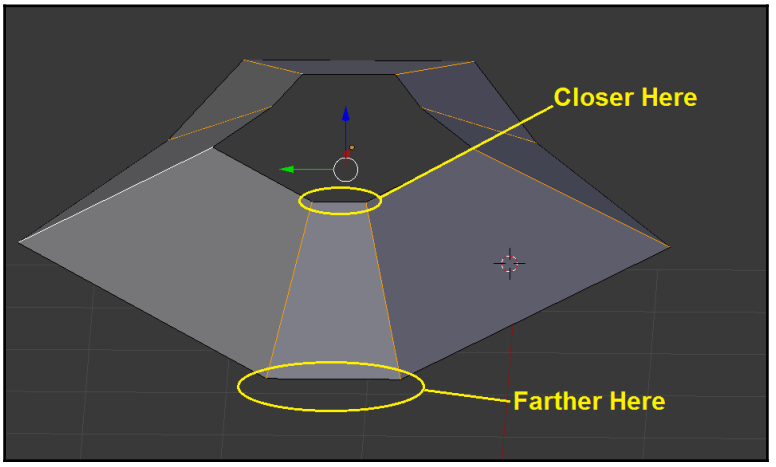
Obviously that's not how it looks in the design, but we're going to be beveling this to round out the shapes. If you'll recall from `Chapter 1`, *Sci-Fi Pistol – Creating the Basic Shapes*, that can cause problems sometimes. Let's say that we scaled down the top vertices to the approximate position that we wanted them to be:



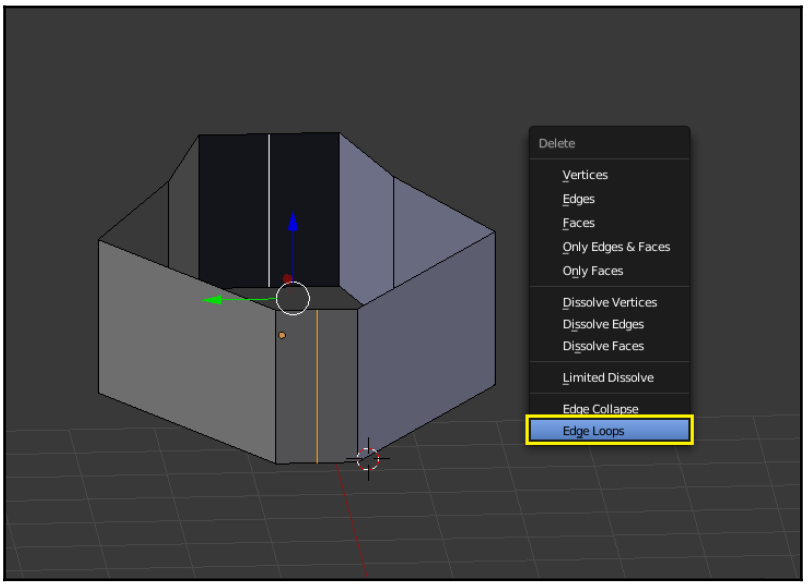
Now if we tried to bevel the edges, we'd end up with something like this:



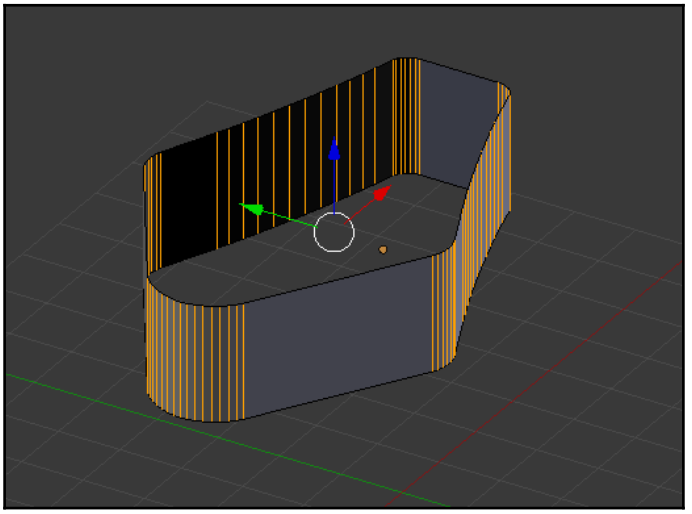
That's kind of a mess, isn't it? This type of overlap tends to occur when the vertices are closer together at one end than at the other:



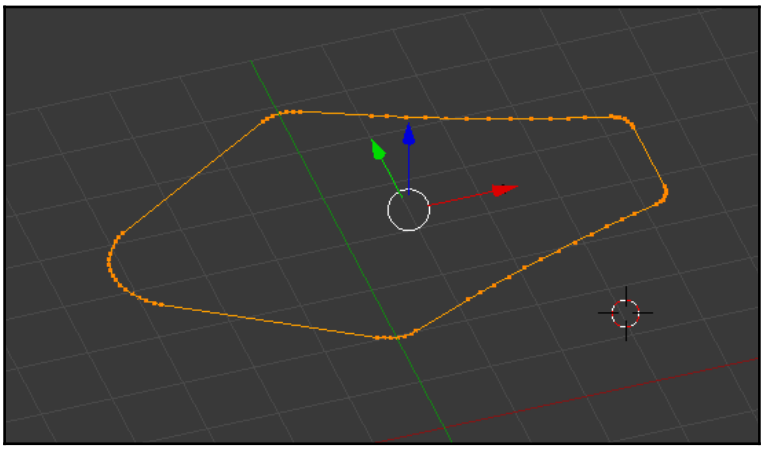
So to avoid this problem, we'll first extrude our edges straight up. Then we'll delete the two-edge loops on the ends so that we can bevel further:



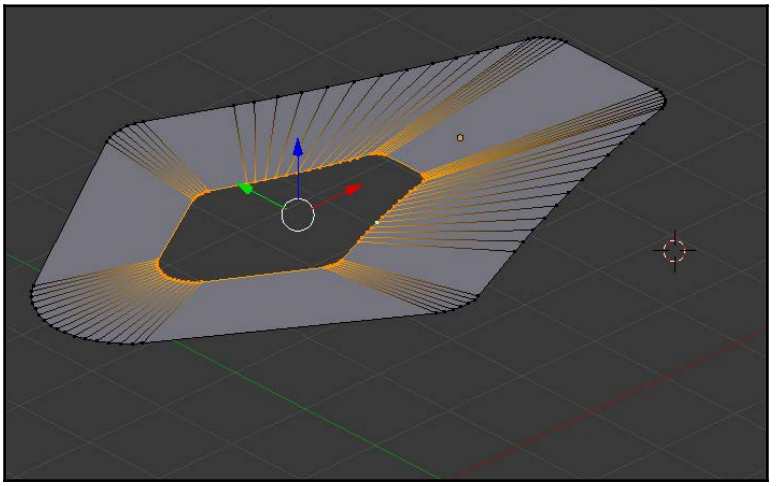
Then we can bevel it and get a nice smooth shape:



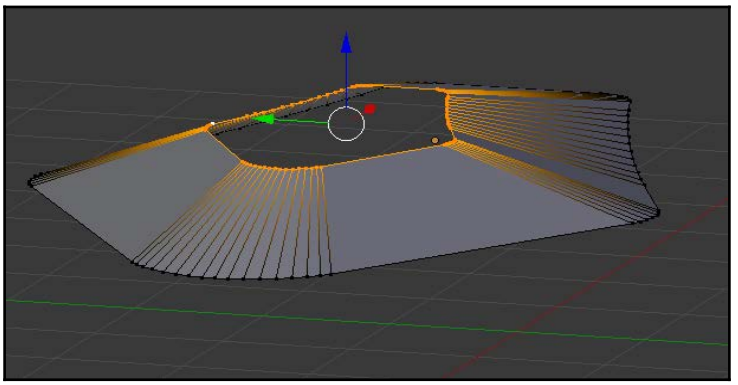
Once that's done, we can delete those top vertices:



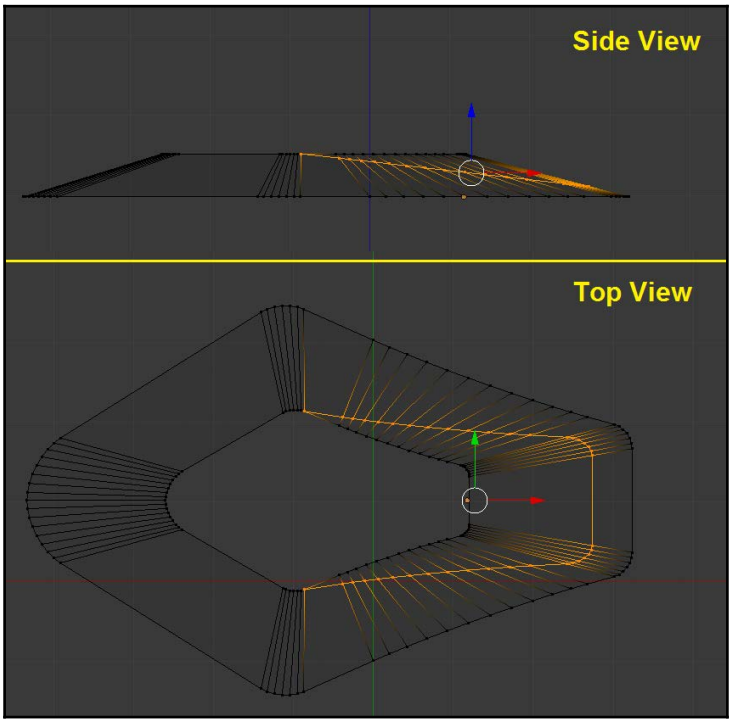
Then we can extrude the bottom ones in and move our inner ring to approximately the correct position:



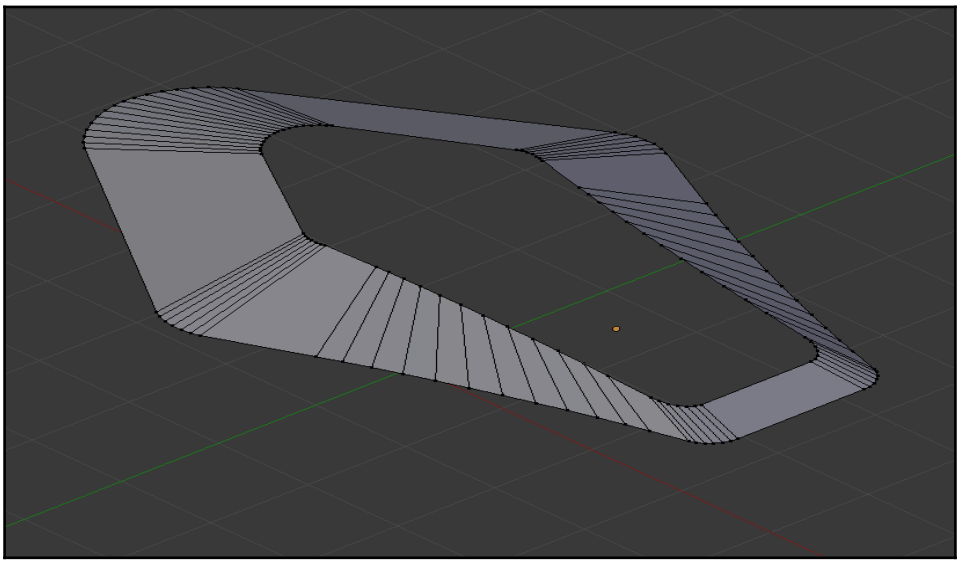
Now, we can pull the whole thing up without running into any trouble:



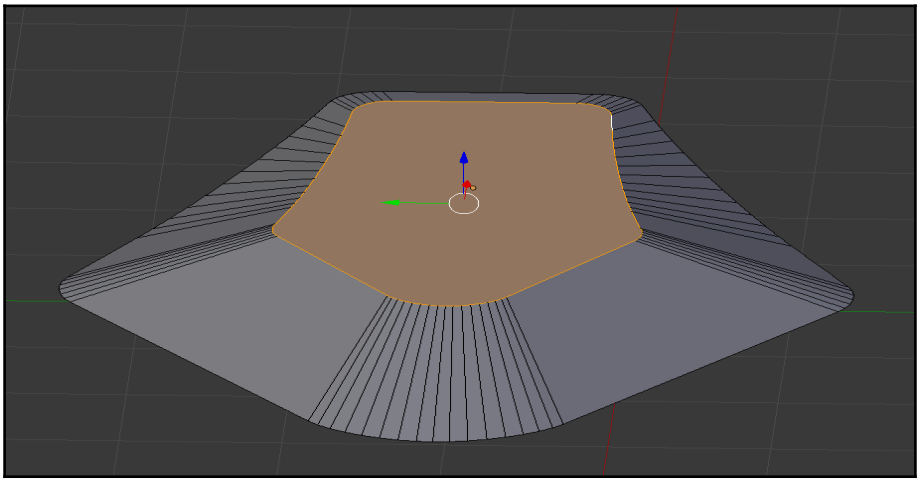
We'll then use our Knife tool from the side to cut the shape at an angle:



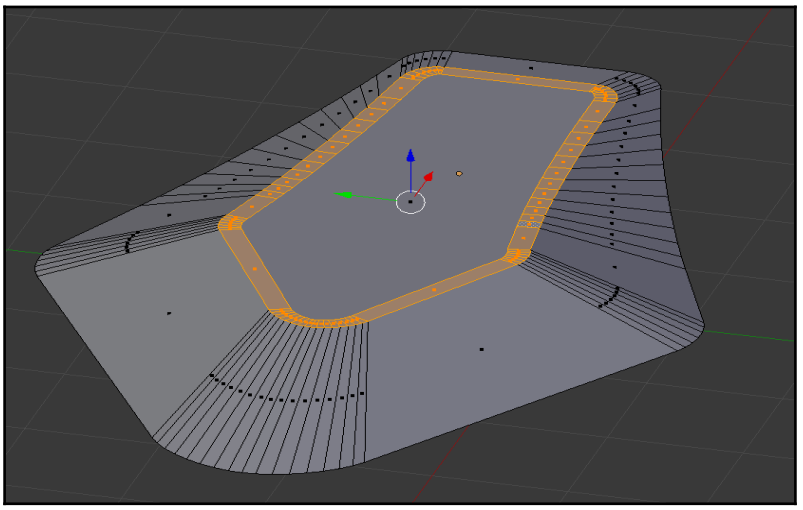
Delete those extra faces that we don't need, giving us the basic shape of the hull:



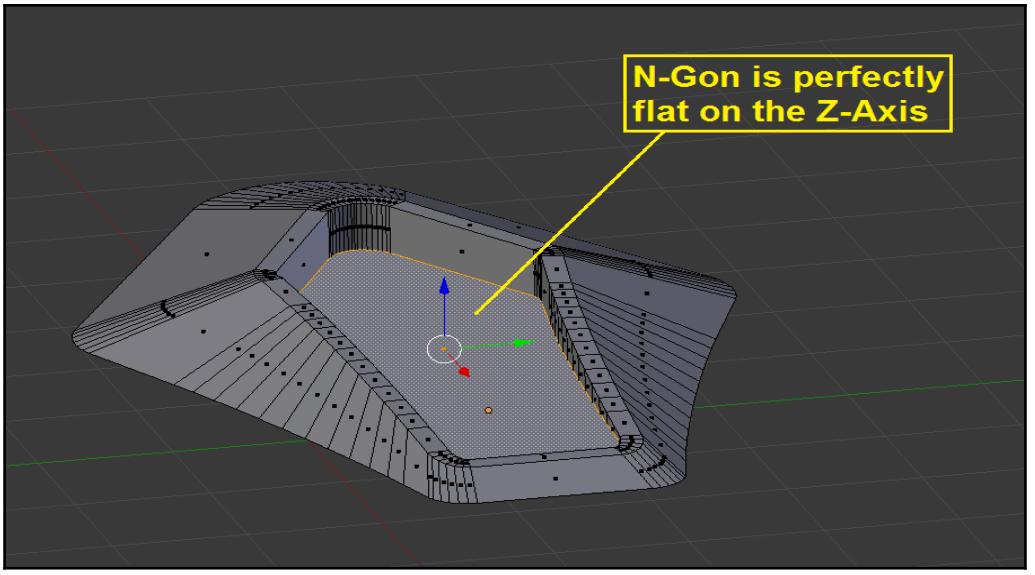
Let's create a temporary N-Gon here at the top. It looks pretty ugly, but that's okay; we'll fix it in a moment.



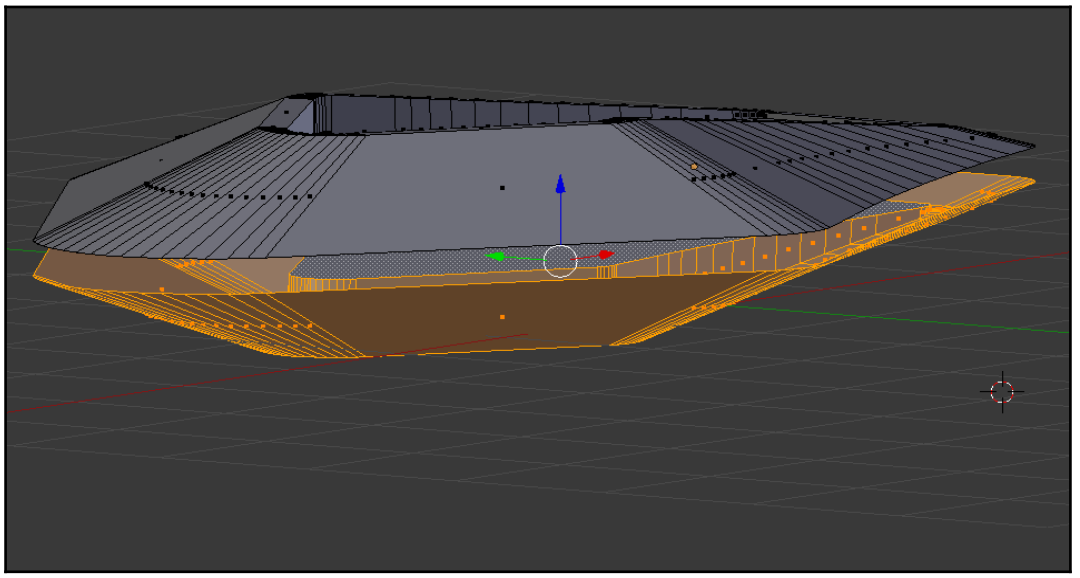
Next, we'll use Inset tool to create a nice even border around the top:



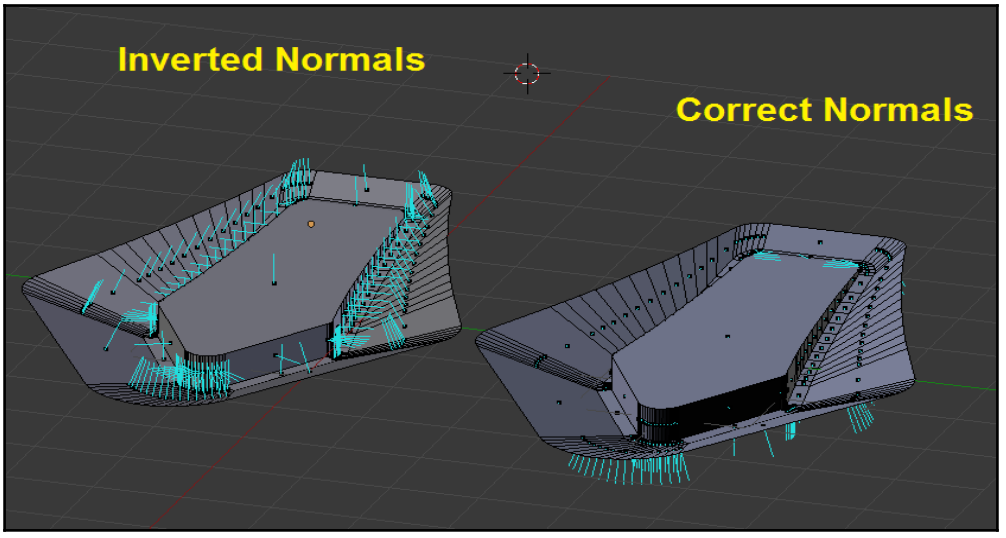
Then, we'll extrude that N-Gon down slightly into the hull and scale it to zero on the *Z* axis (S, Z, 0):



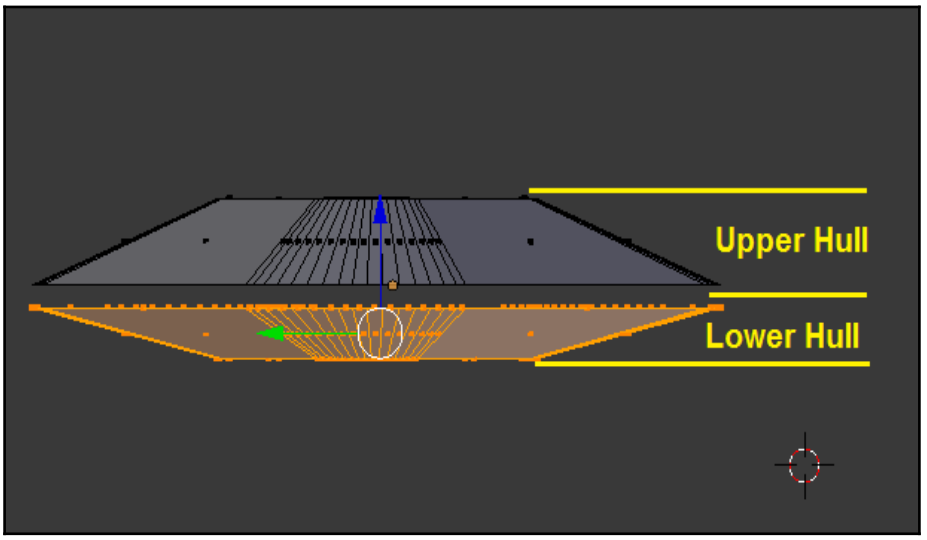
Before we go further, let's duplicate the upper section and flip it down across the *Z* axis (using *Ctrl* + *M*):



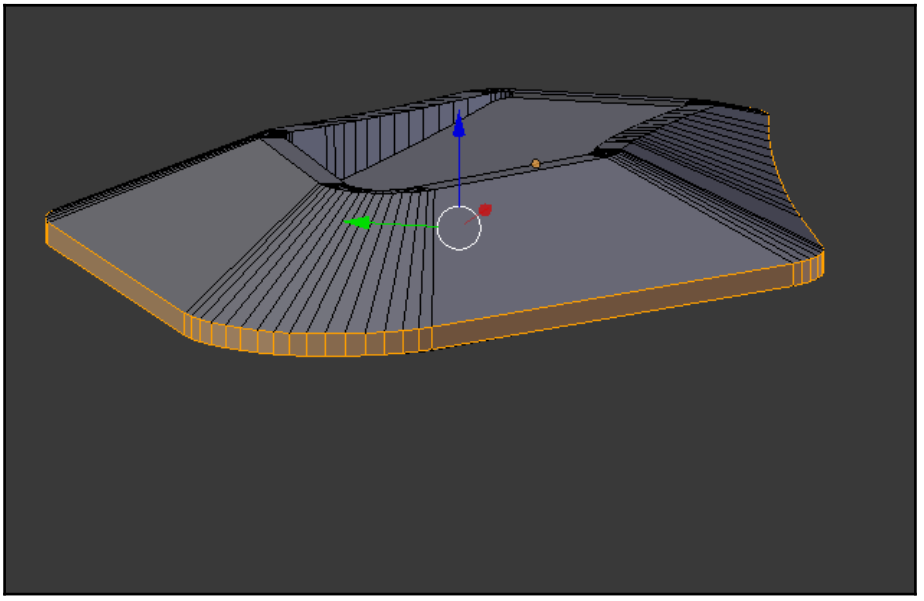
Don't forget to fix the **Normals**!



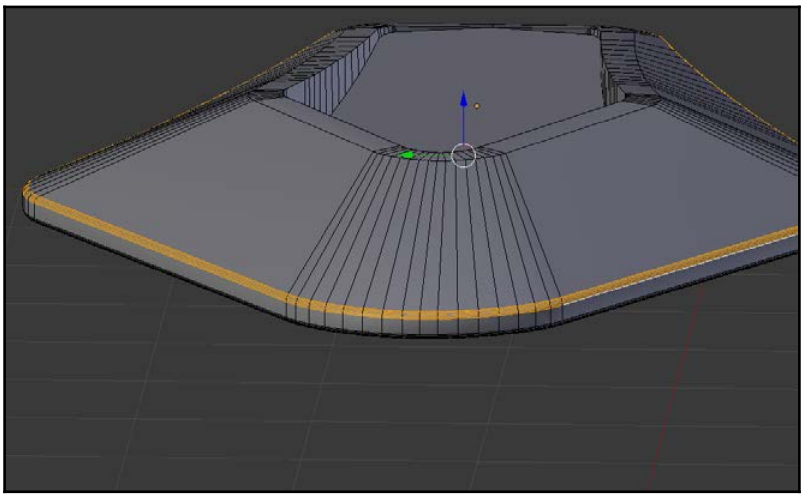
Let's scale the lower section down a bit to match the design:



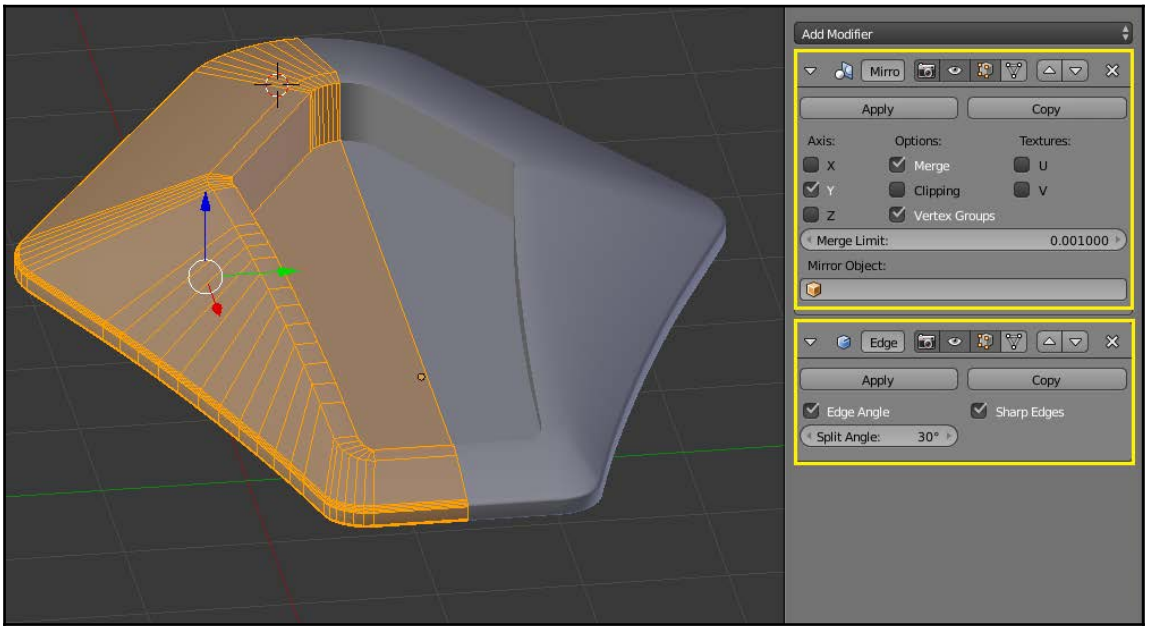
We can now select both the upper and lower edges and bridge the edge loops:



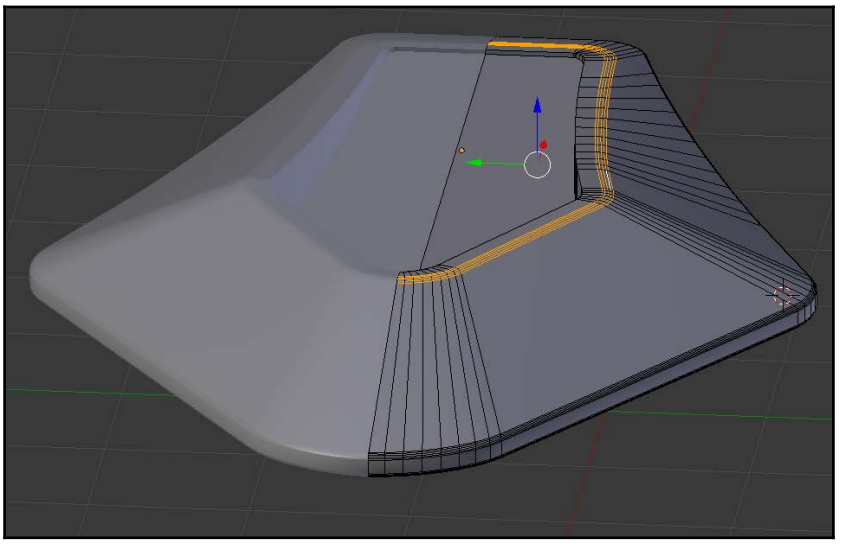
Now, we can bevel these edges at the top and bottom:



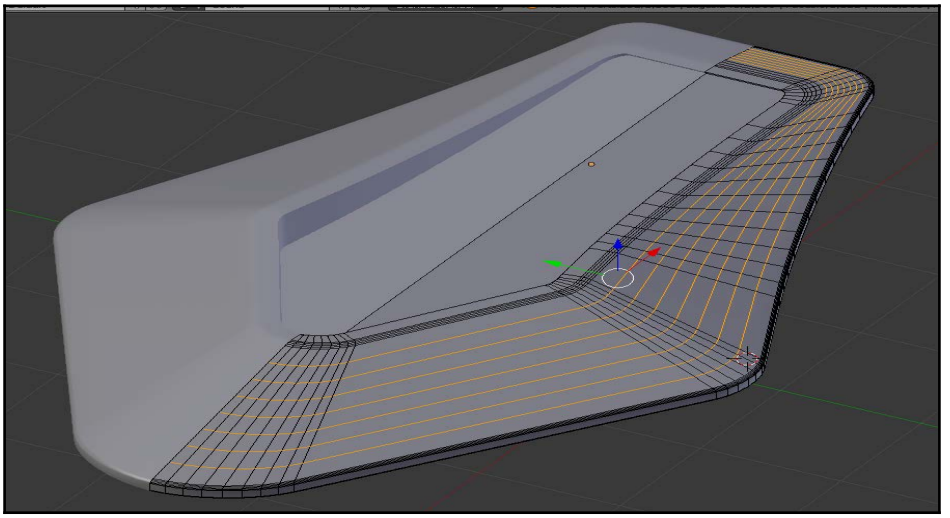
That gives us the basic shape of the body. Before we go further, let's delete half of it and add both a **Mirror** and an **Edge Split** modifier:



Then, we'll add a slight bevel to both the top and the bottom:



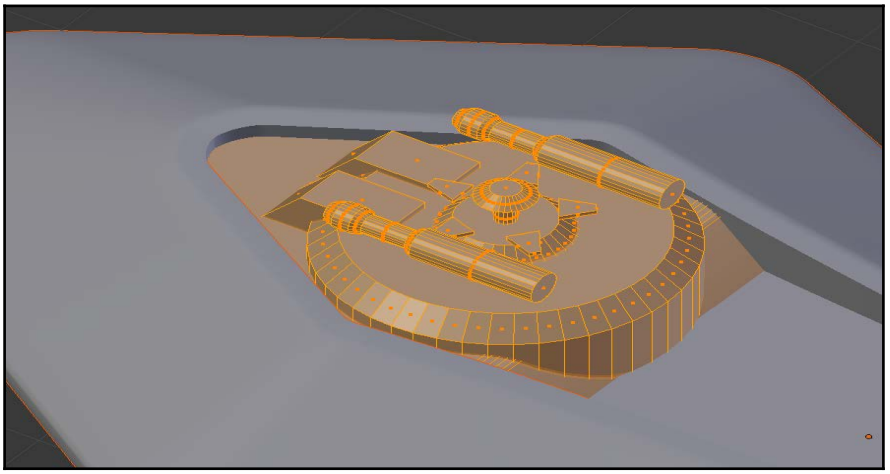
Now, we'll add a few more loop cuts around the side, just to give us more options when we make cuts in the mesh later on:



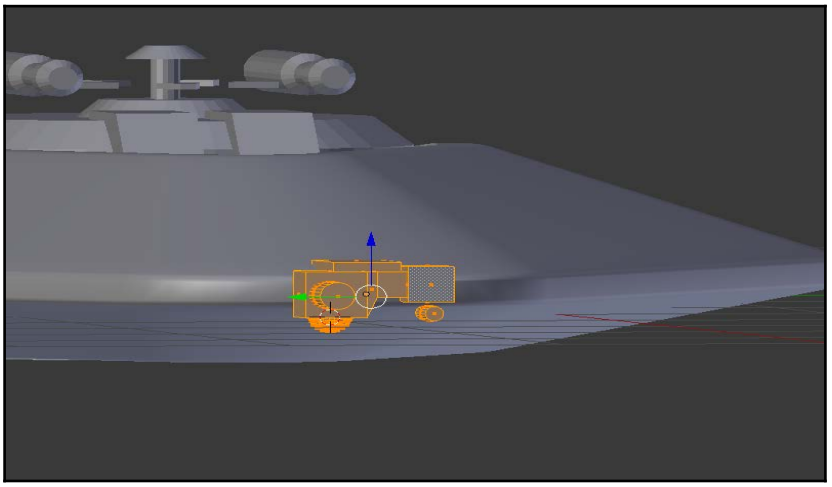
That wraps up the basic shape of the body. Wasn't too difficult, was it?

Creating the accessories

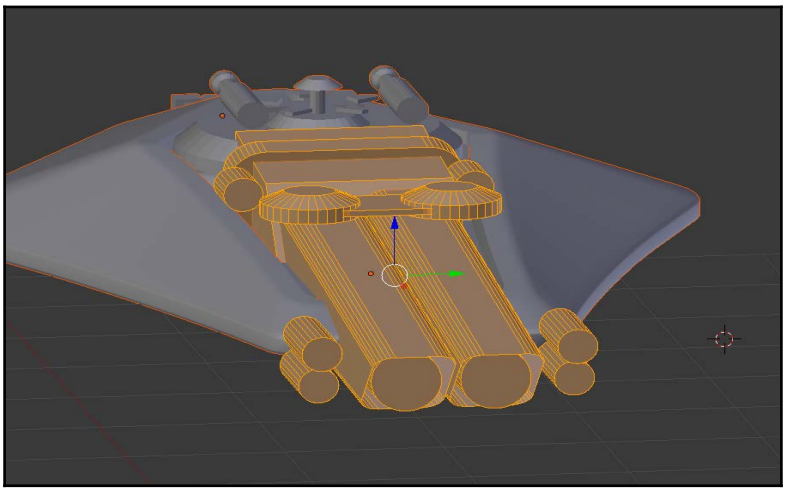
Next, I'll add in a couple basic cylinders and cubes at the top to fill in the hull and represent the equipment up there:



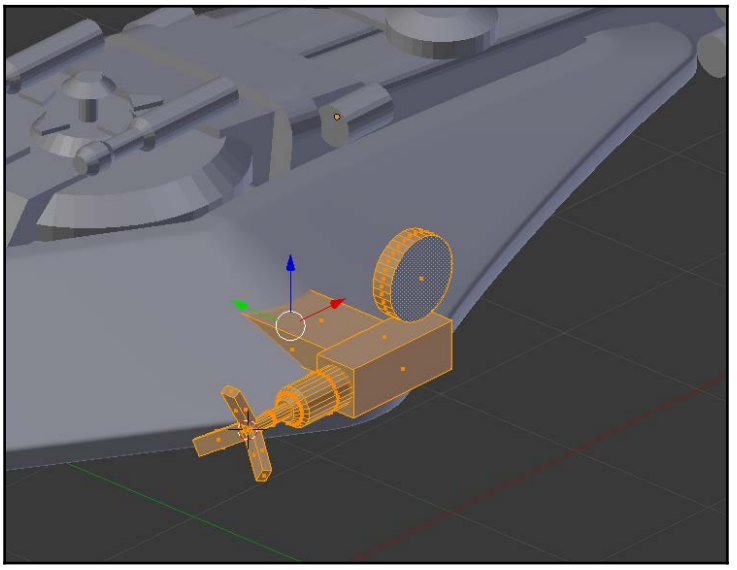
Then, I'll do the same for the navigation array at the front:



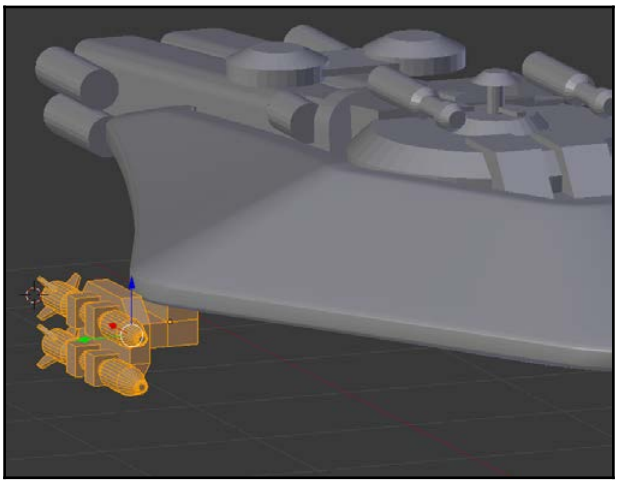
At the back, I'll add a couple of beveled cubes and cylinders to represent the engine area:



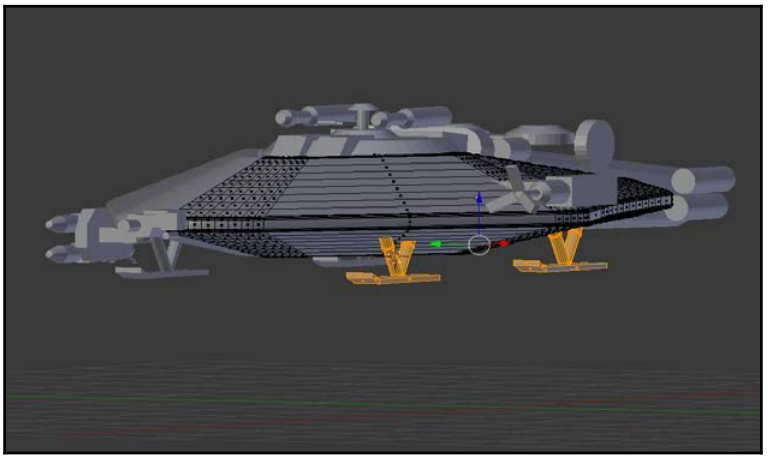
I'll also add some basic shapes in for the grappling gun on the port side:



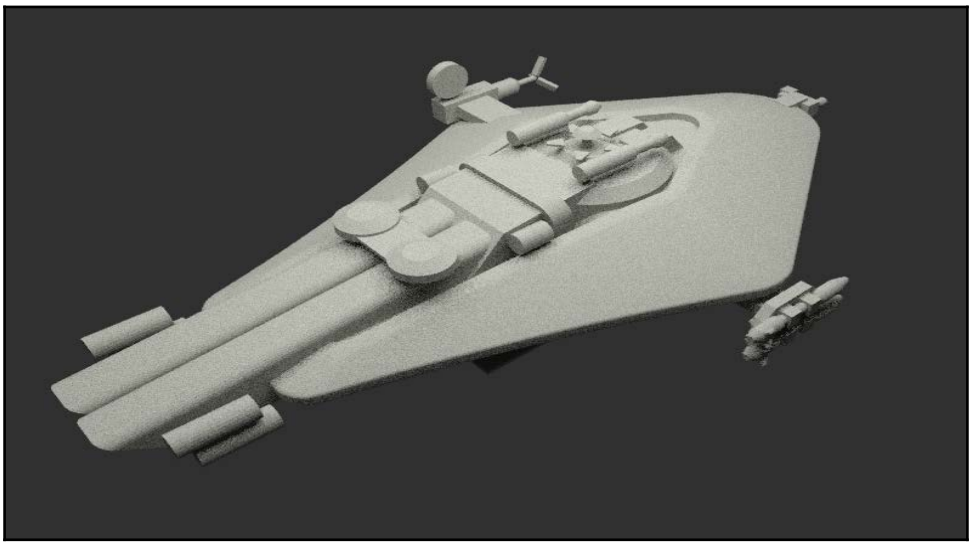
And the missile launcher on the starboard side:



I'll block out some basic landing gear (skids) on the bottom:



That really wraps up the basic shapes of our ship:



It's a 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!

5
Spacecraft - Adding Details

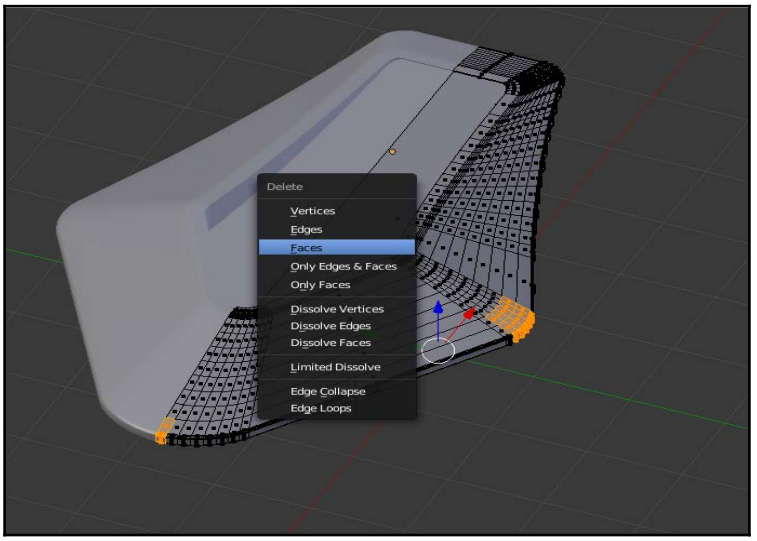
In this chapter, we'll finish modeling our spacecraft. As we do, we'll cover a few new tools and techniques. Mostly, however, we'll be expanding on what we have already learned, and applying things in different ways to create a final, complex model:

- 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

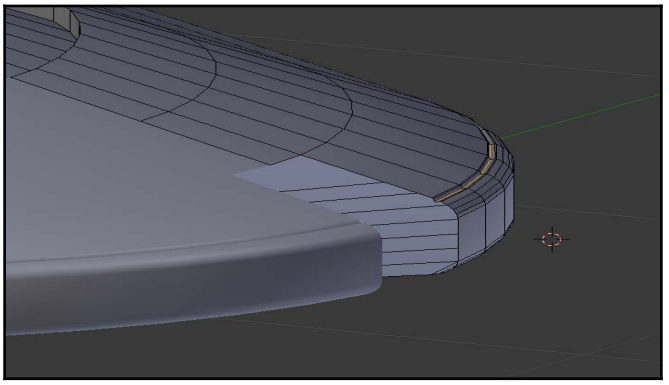
Refining our ship

We'll work though the spacecraft one section at a time, adding detail until it's finished.

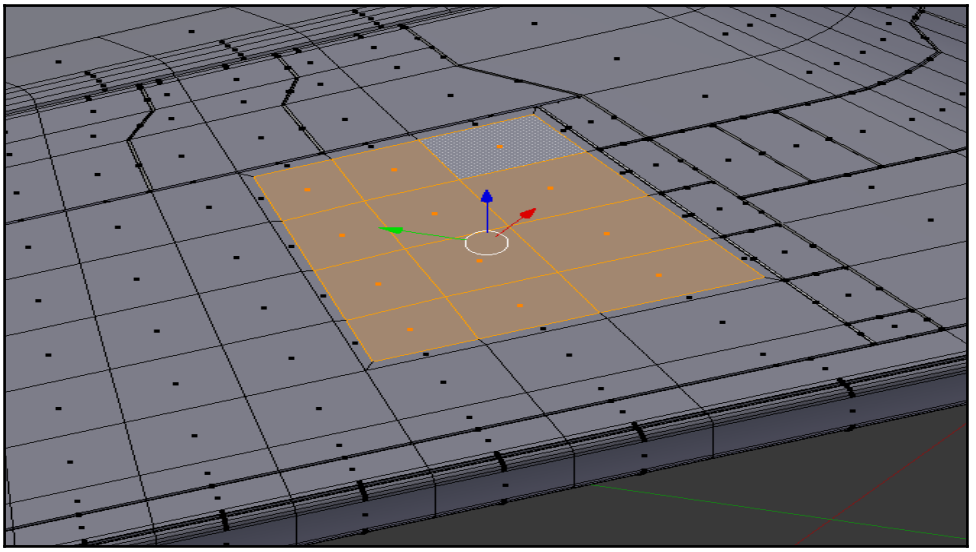
Let's start by moving everything but the main body to another layer. Then, we'll select the portions of the body that we want to remove (to create gaps in the hull) and delete them:



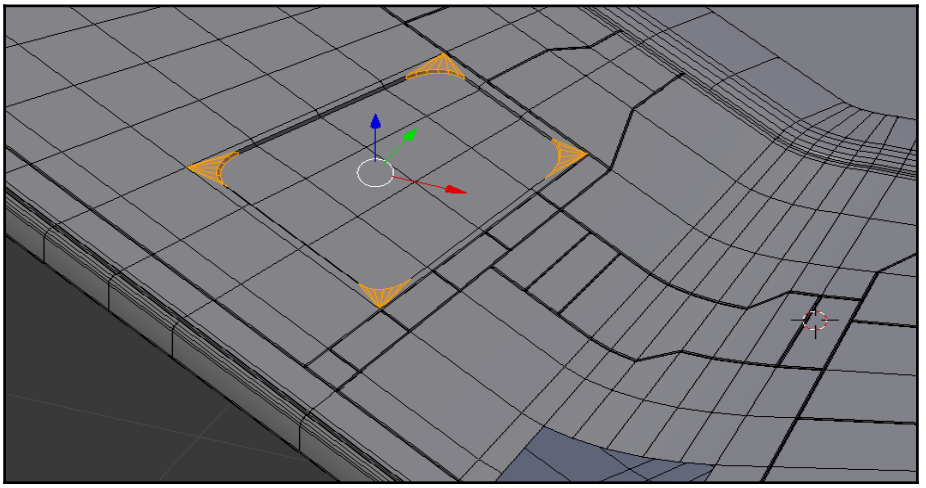
We can use *Alt* + *S* to scale in some edges, then rip them with the *V* key:



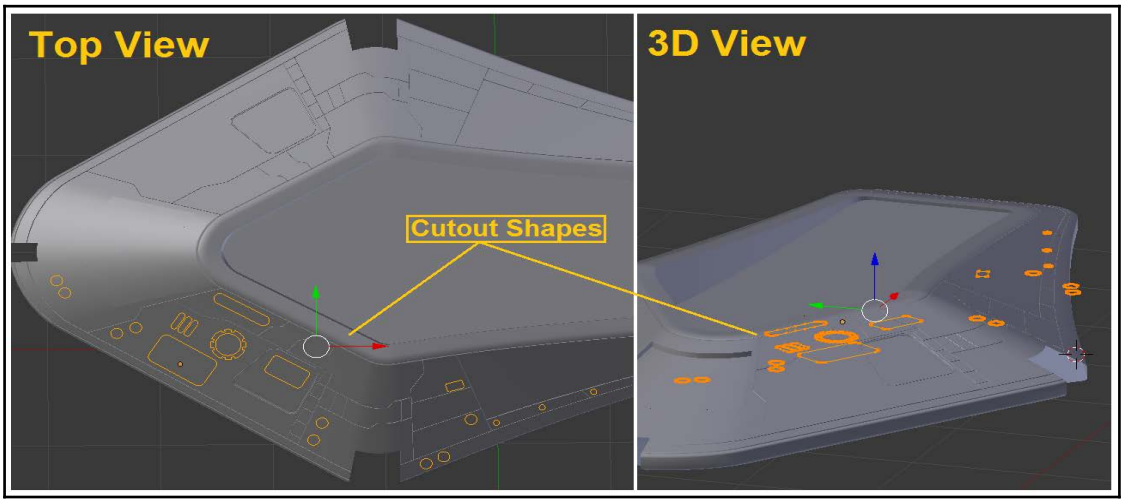
I'm also going to use the Inset tool to create a door (or hatch) on the side of the ship:



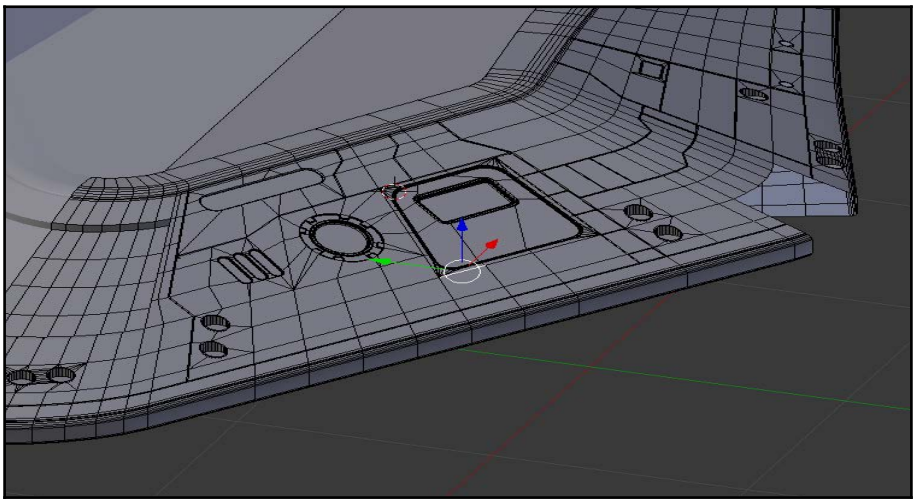
Then, I'll just bevel those edges so that they're more rounded:



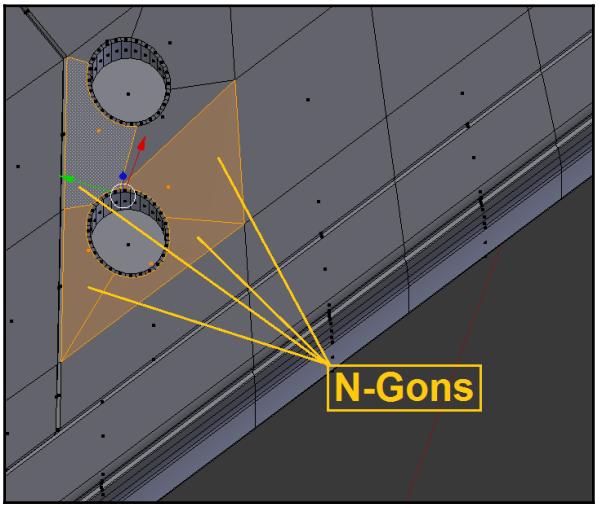
Next, we'll create a series of cutout shapes to use for the top of the hull (just like we did on our gun model):



Using the same techniques from `Chapter 2`, *Sci Fi Pistol – Adding Details*, we'll cut those pieces into our hull:

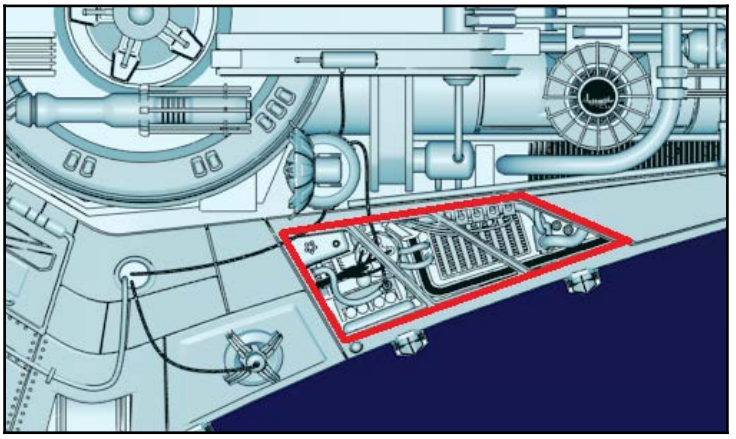


I'd like to point out that I've cheated a little bit here. Earlier in the book, I said you should never use N-Gons in curved areas. That's an excellent rule of thumb, but there are times when you can get away with it. For instance, I've done it here:

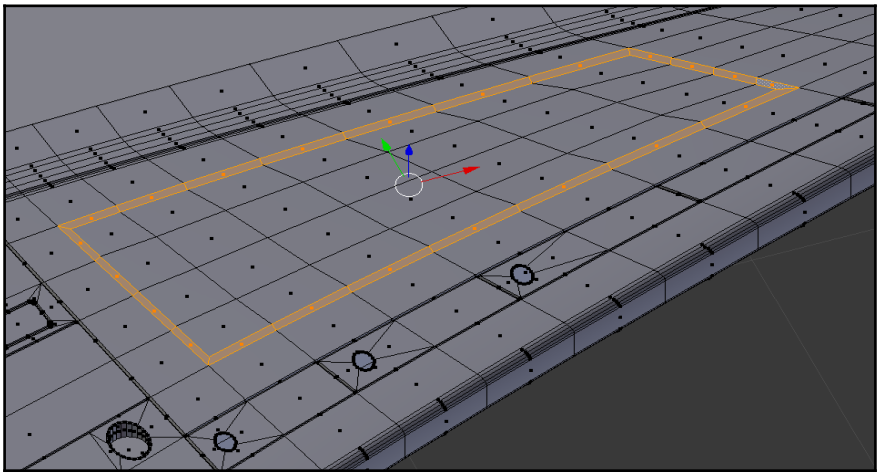


That hull area is technically curved, but it's so close to being flat and the faces are so small that there's no noticeable distortion-it certainly saves a lot of time. This is really a judgment call.

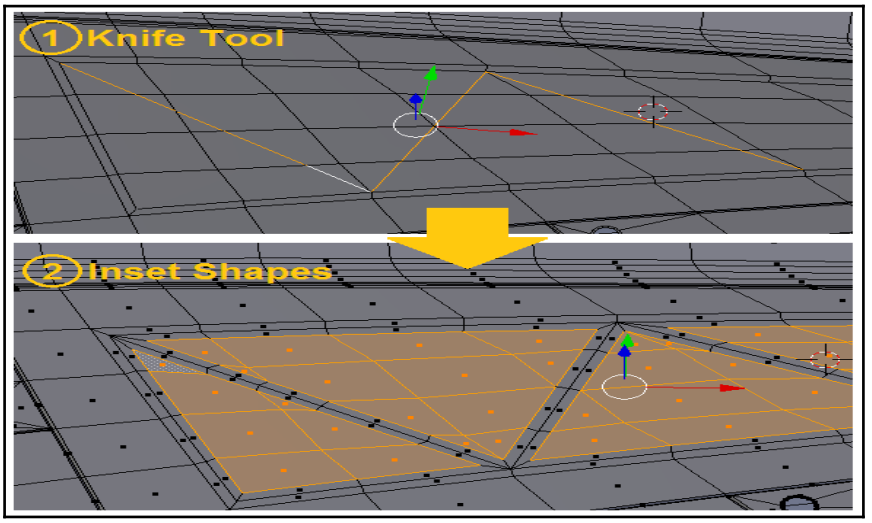
Next, we'll create some space for our mechanical area near the back:



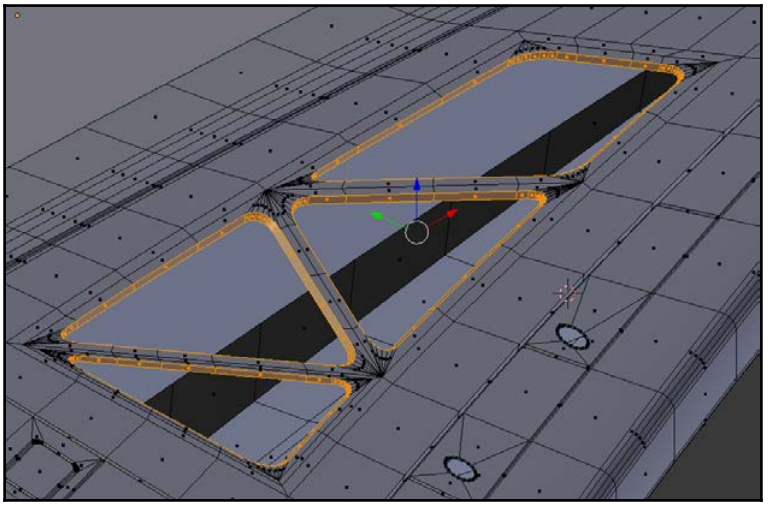
We can do this by just grabbing a good area of faces and using the **Inset** tool again:



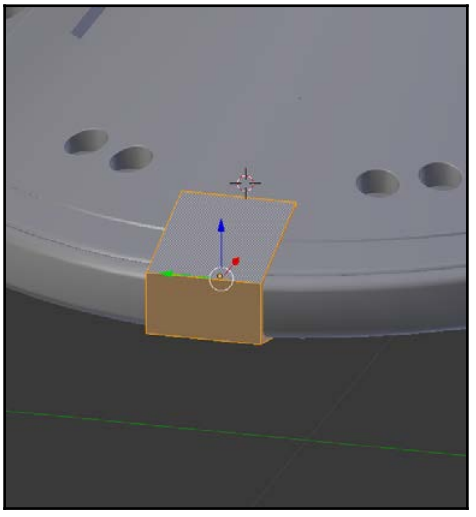
With my Knife tool, I can now create some nice diagonal cuts in those panels:



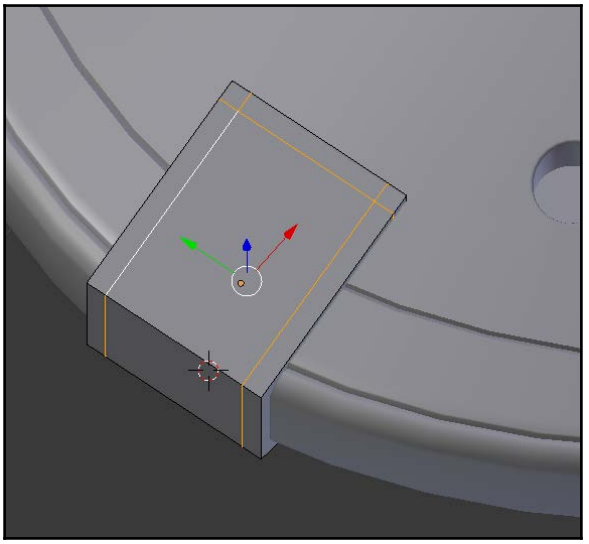
Using beveling again, I'll round out those edges:



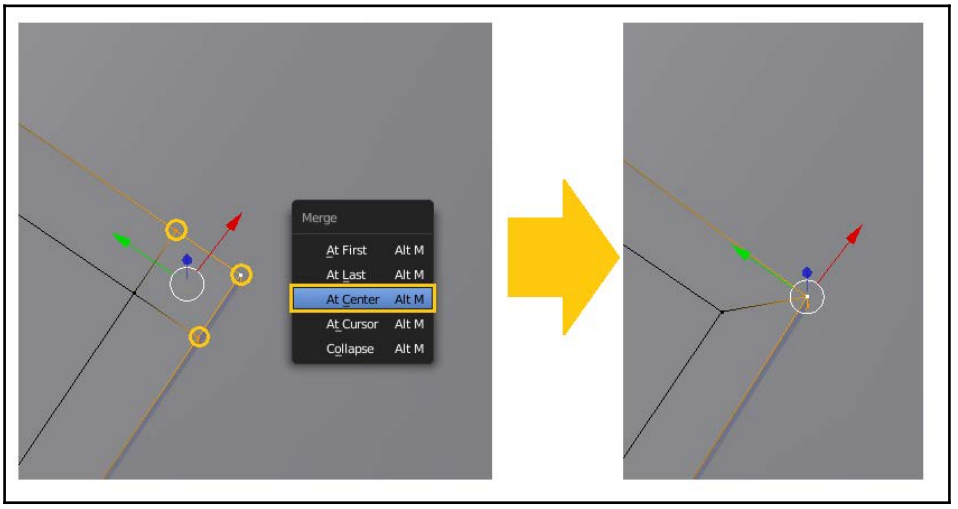
Next, we'll fill in those open gaps for the nose and side-mounted equipment. Start by adding a new **Cube** object and moving it into position:



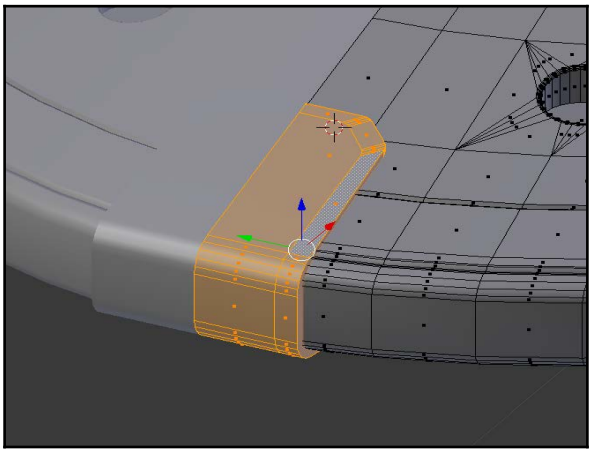
Then, we can run some loop cuts where we want the borders to be:



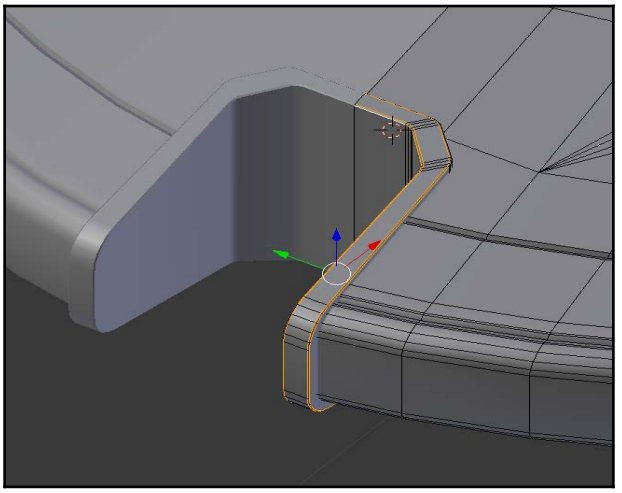
We can then merge those corner vertices, so we get a nice slanted line for beveling:



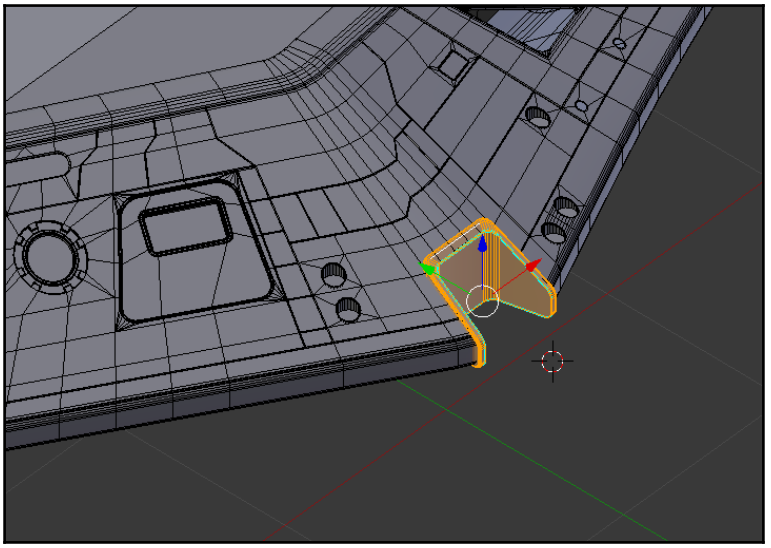
Now, we can use the bevel tool to create the shape we want. Then, we'll delete half of it (to make use of the Mirror modifier) and join it to our body:



We can then delete the faces created inside the loop and bridge the top and bottom edge loops to create our final shape:

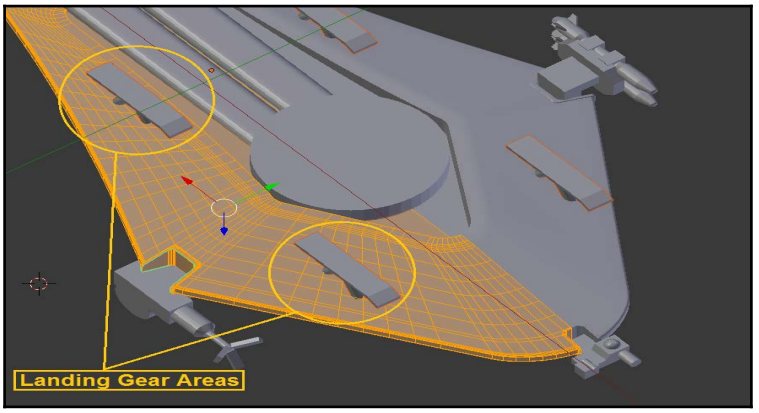


We'll use the same technique on the sides of the ship:

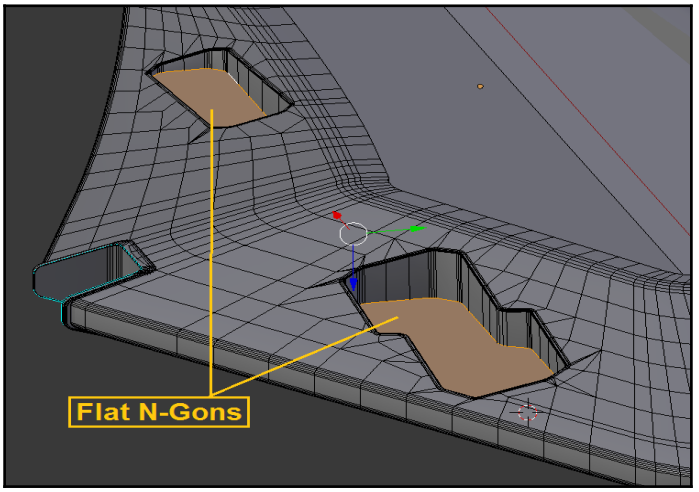


Even though the missile launcher is different from the grappling gun, I'm going to make these cutouts in the hull identical (just for simplicity's sake).

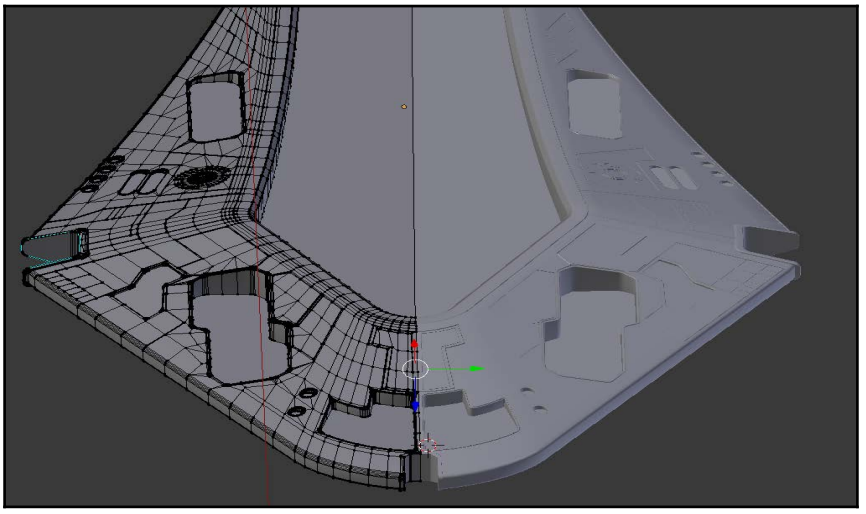
Next, we'll create some cutouts on the bottom for our landing gear:



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:

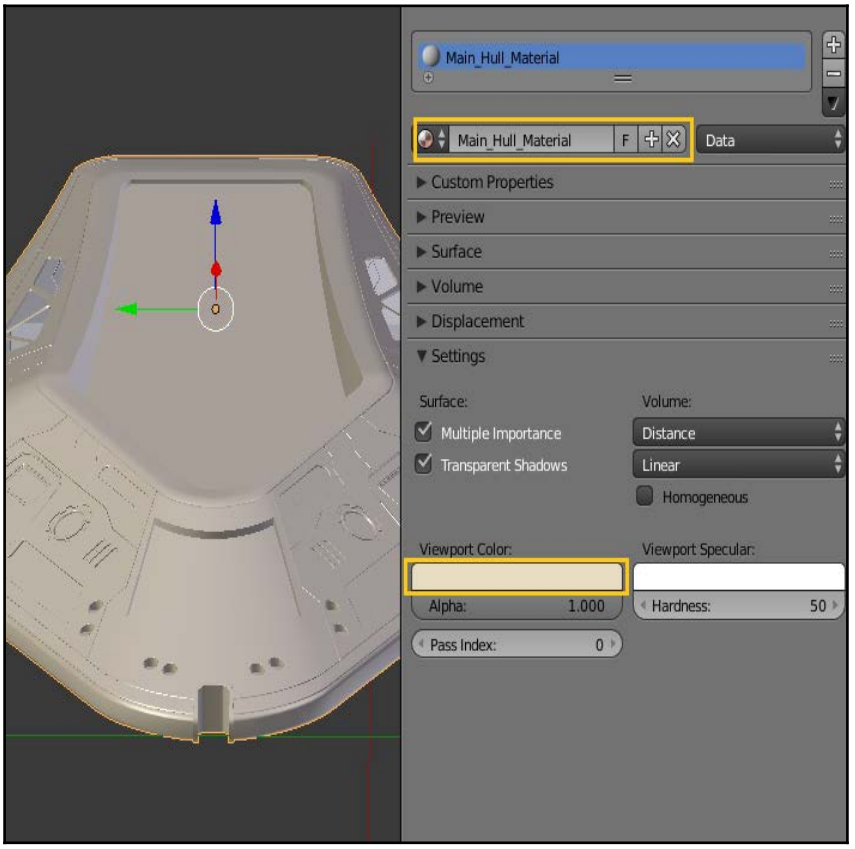


Using the same techniques from before, I'll create some cutouts in the bottom of the ship:



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.

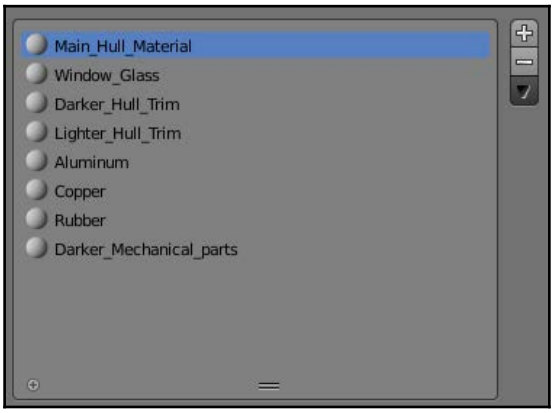
Here, I've just created one called **Main_Hull_Material** and assigned it a viewport color:



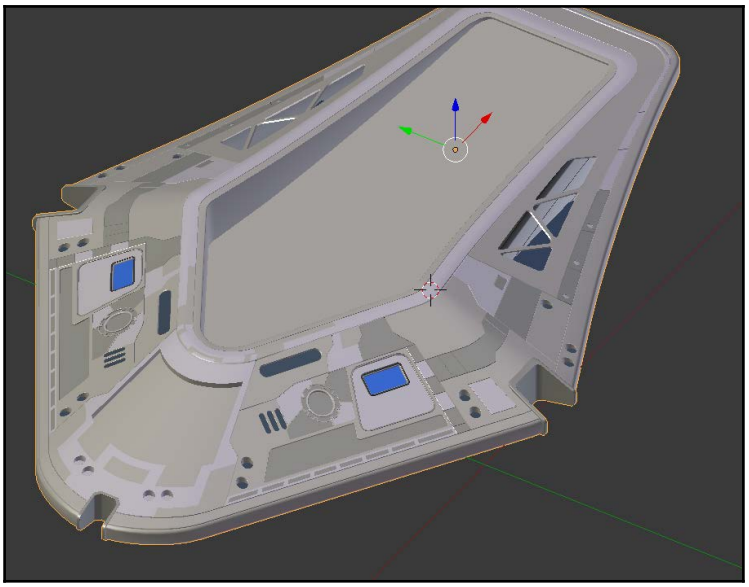
This way, I can assign materials as I model.

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.

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:



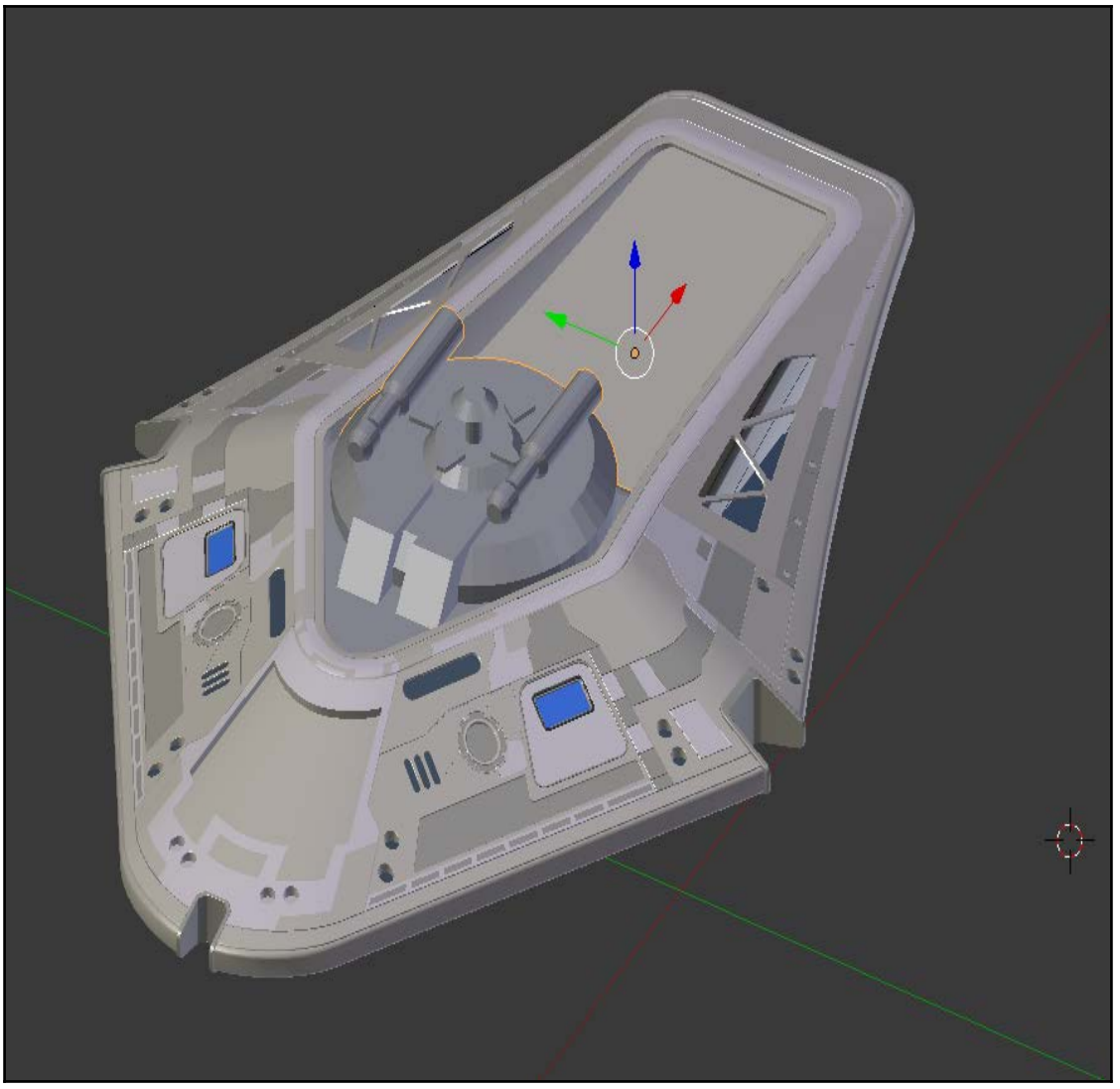
I've given them all different viewport colors so that we can tell them apart:



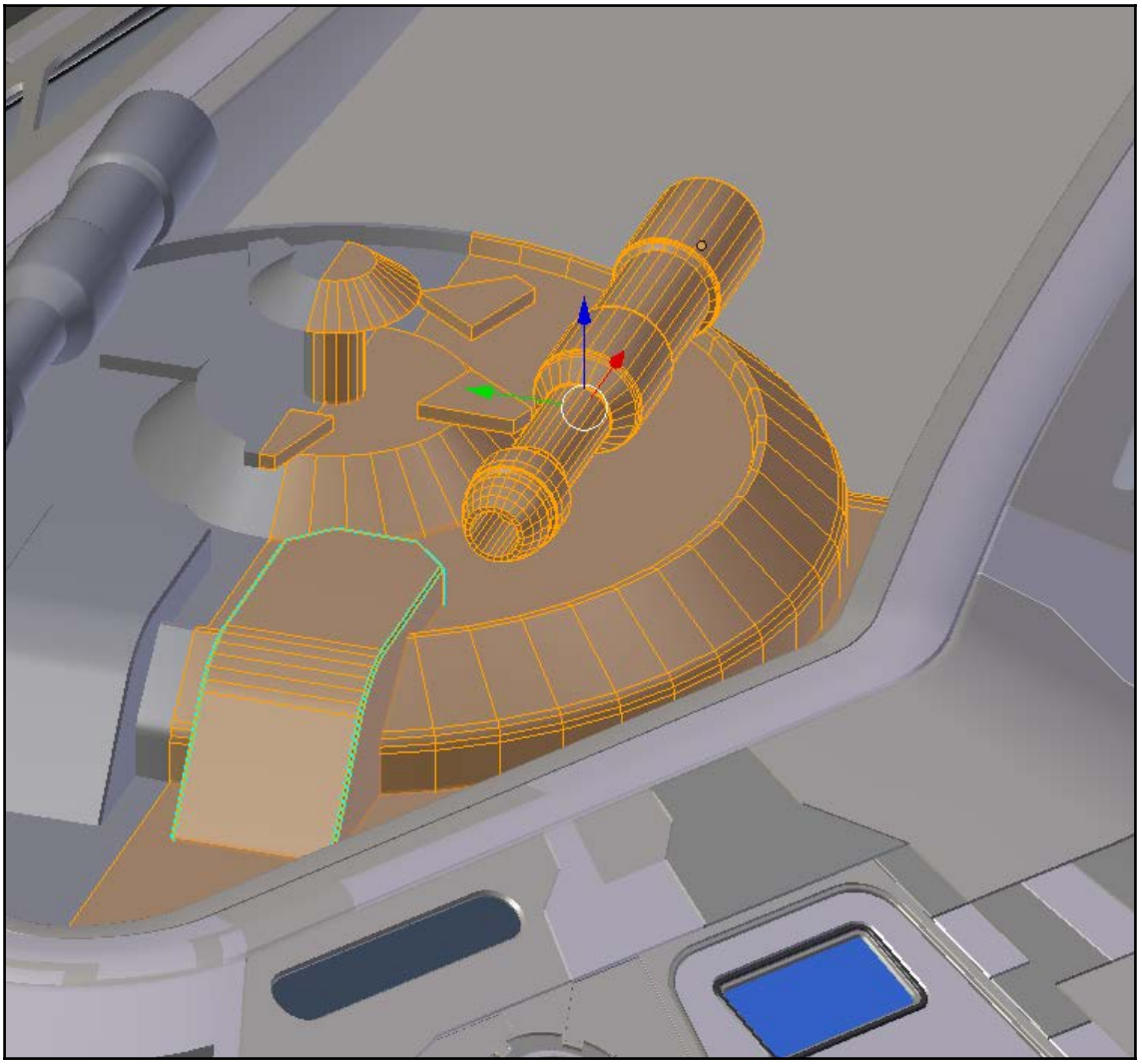
I think we're at a good stopping point now with the body. We'll probably want to come back to it and work on it later, but I'd like to get the rest of the model up to this same level of detail.

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.

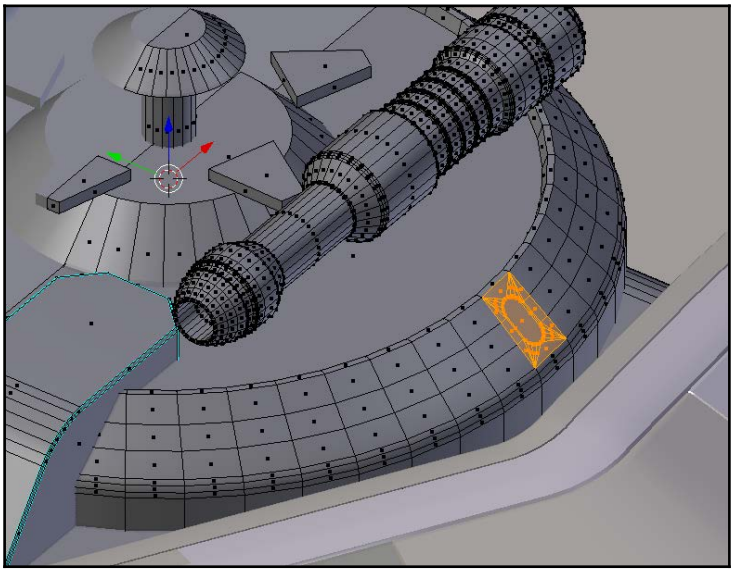
So, let's go to our other layer (layer 2, in my case) and bring that top section back to the same layer as our body:



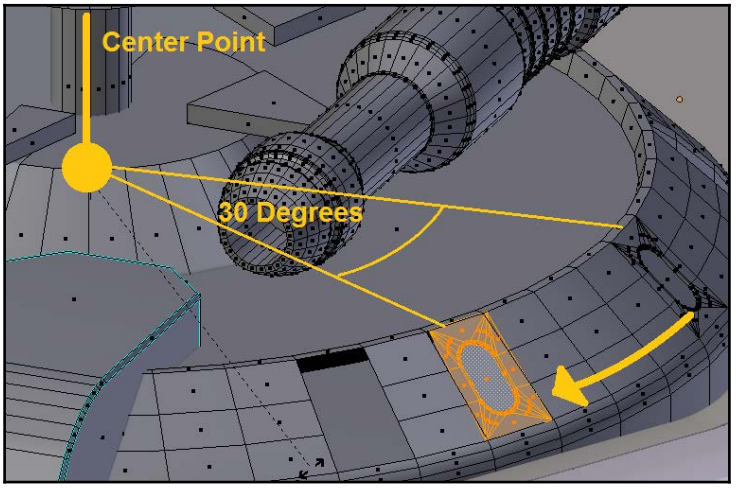
First, I'll just do some basic beveling to refine the shape a bit:



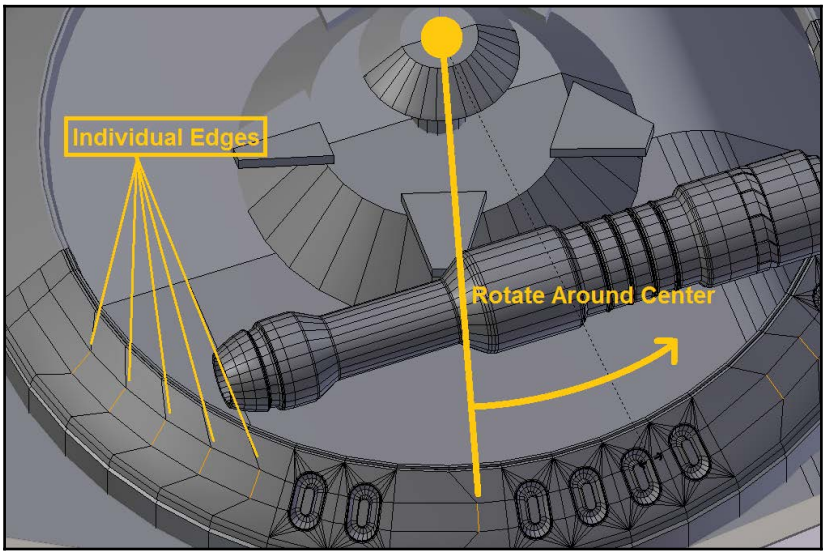
Again, we'll use a **Shrinkwrap** modifier to add cutouts. In this case, I'll just start with one:



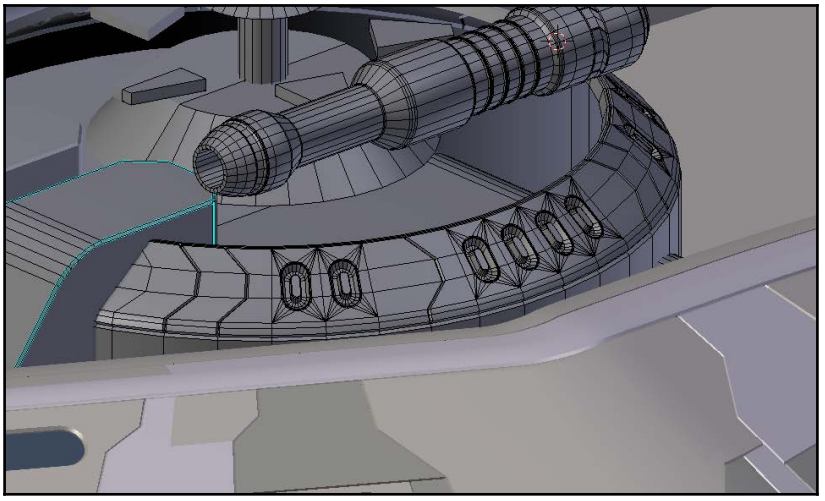
You can then set your **3D Cursor** to the center of that circle and rotate the cutout piece into position (you'll have to delete the original geometry first). Once you do that, of course, you'll need to remove the doubles you created. This saves you the trouble of cutting all those shapes in one at a time:



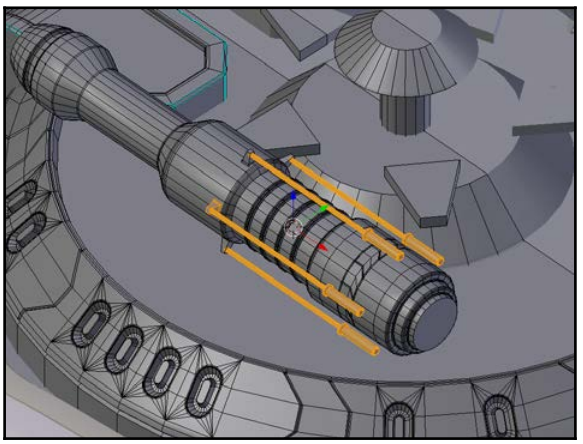
Just to add a little detail, I'm going to grab some individual edges and rotate them around the center point as well:



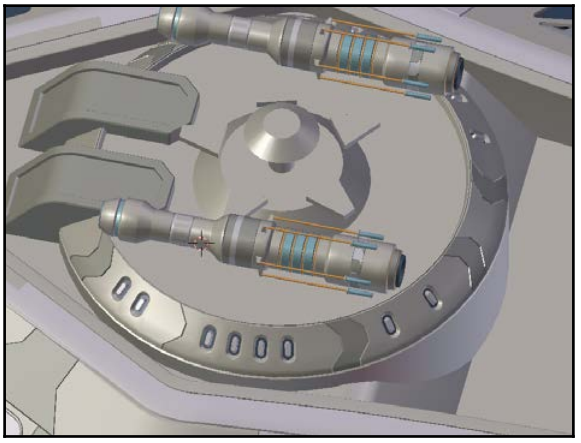
Once we bevel them, I think it looks pretty neat:



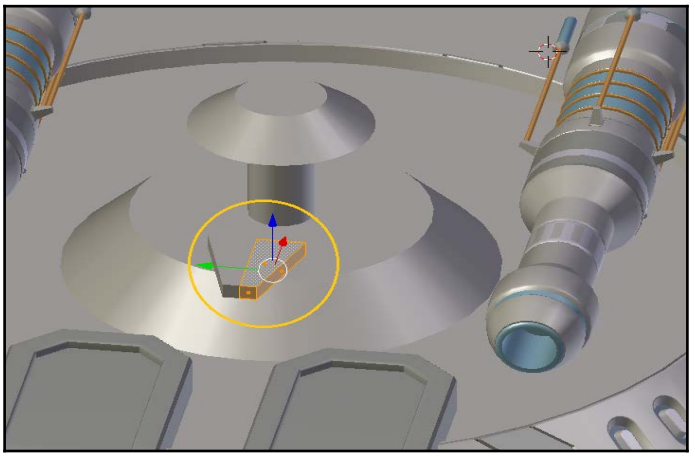
Next, I'll just add a little detail to the main cylinders on top. I'm not really sure what these are. They could be weapons, antennas, tractor beams, or something else altogether. It's up to you!



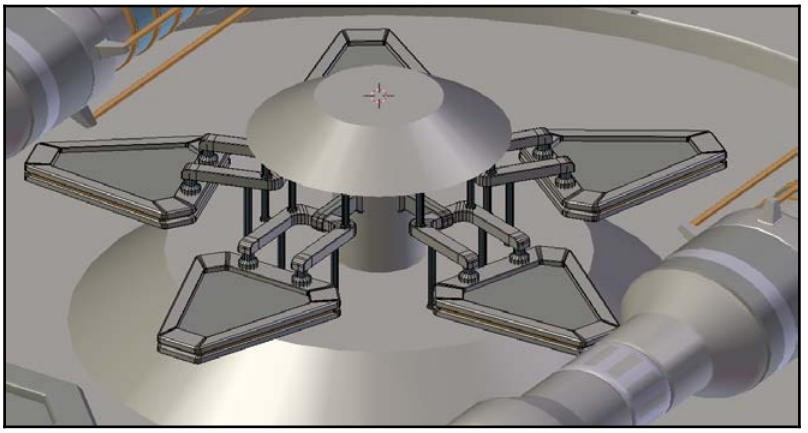
At some point, you'll want to join the top section with the main body. Once you do, you can assign some materials to the different parts:



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:

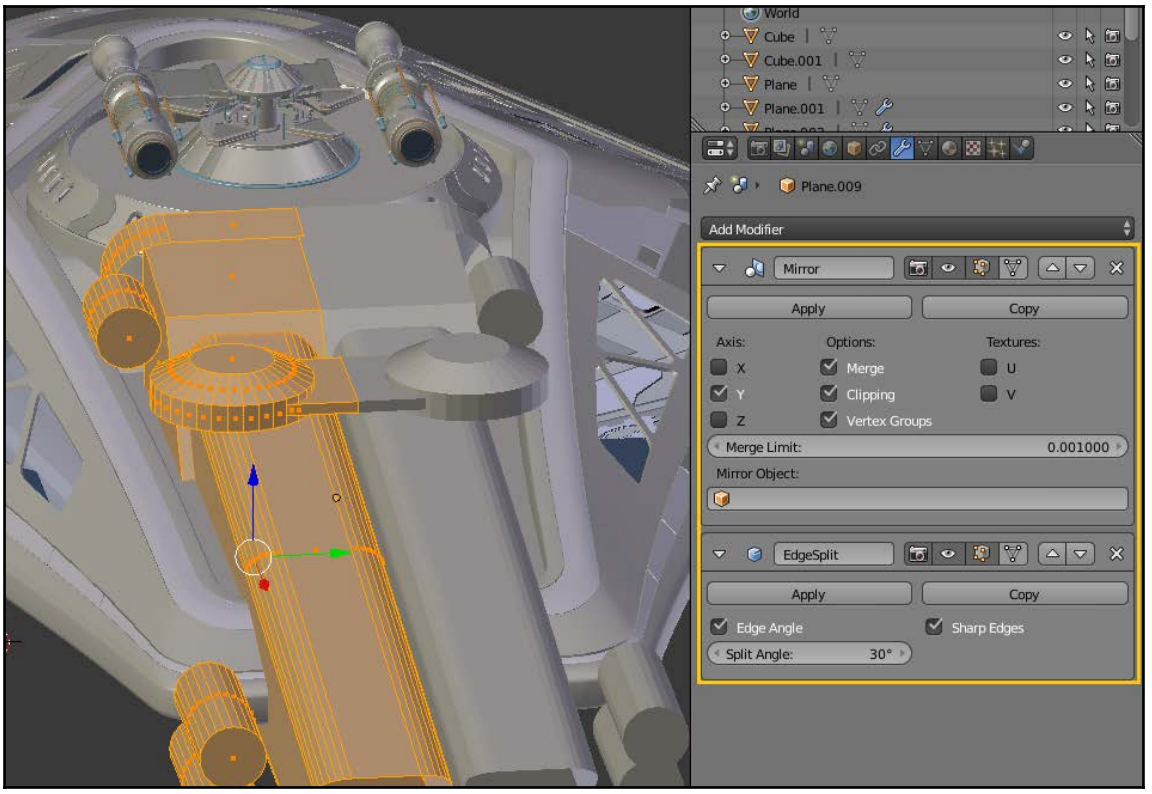


Now, I can add details and materials, and then duplicate and rotate it around the center point:

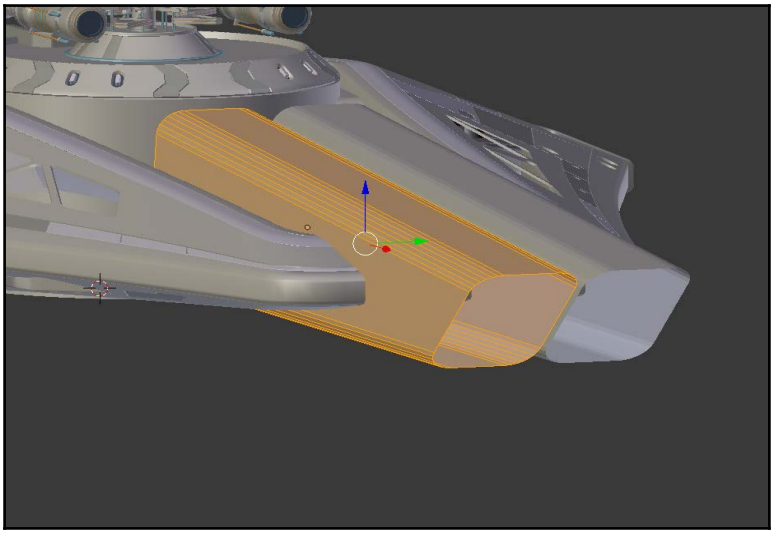


This is a good example of what I was talking about earlier—adding materials first, then duplicating. It tends to save a lot of time.

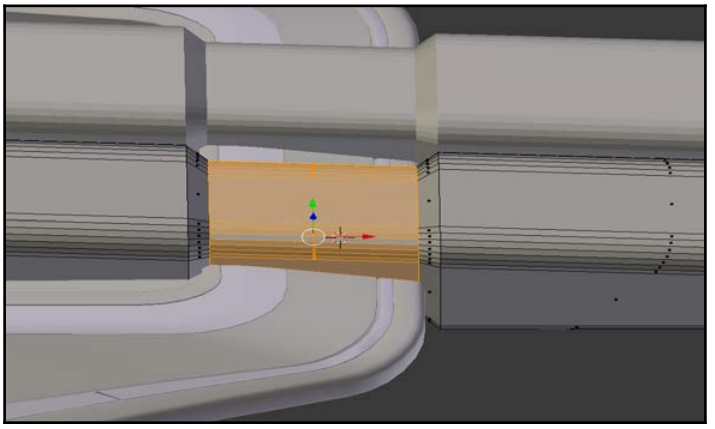
Next, we'll bring in the engine section. I'll delete half of it, then add a **Mirror** and **Edge Split** modifier. I'll also add our materials to the engine so that we can assign them as we go.



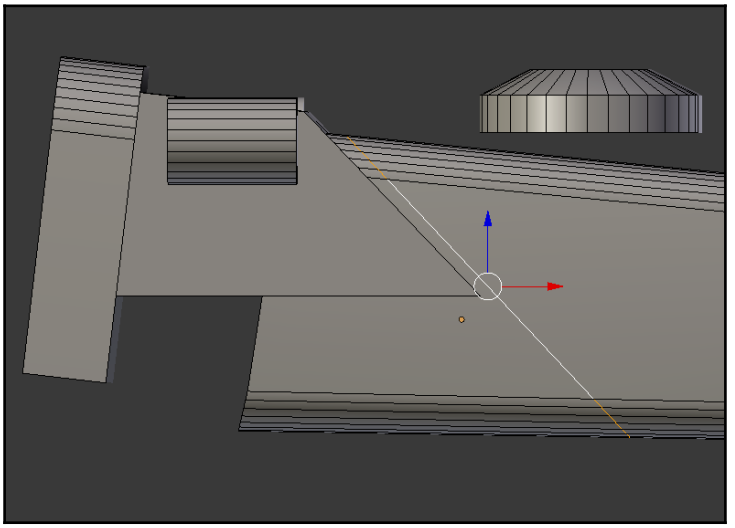
Let's get to work on the engines. I'll start with some basic beveling again:



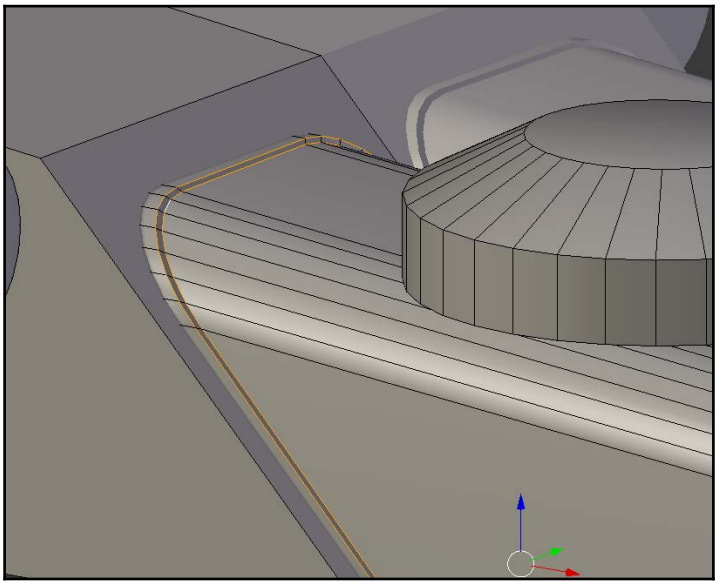
In the middle, we can add just a little detail:



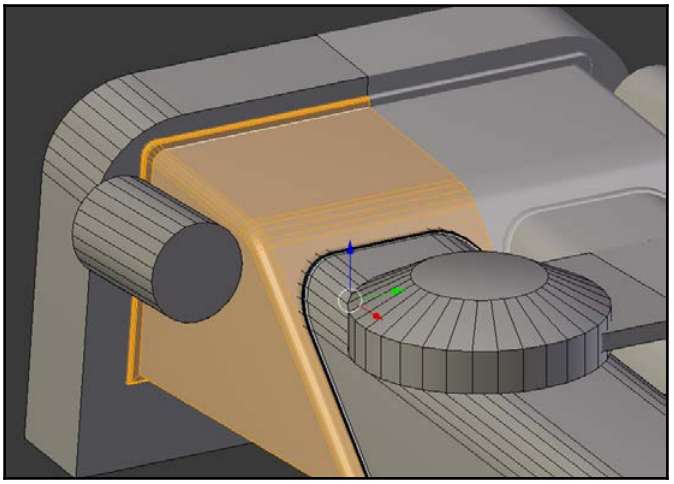
Next, I'll use the **Knife** tool to make a cut across the long part of the engine here:



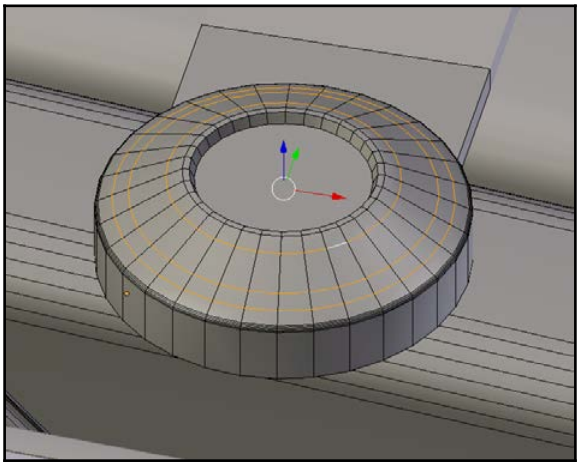
Then, I'll extrude those edges forward so that they make a nice little border:



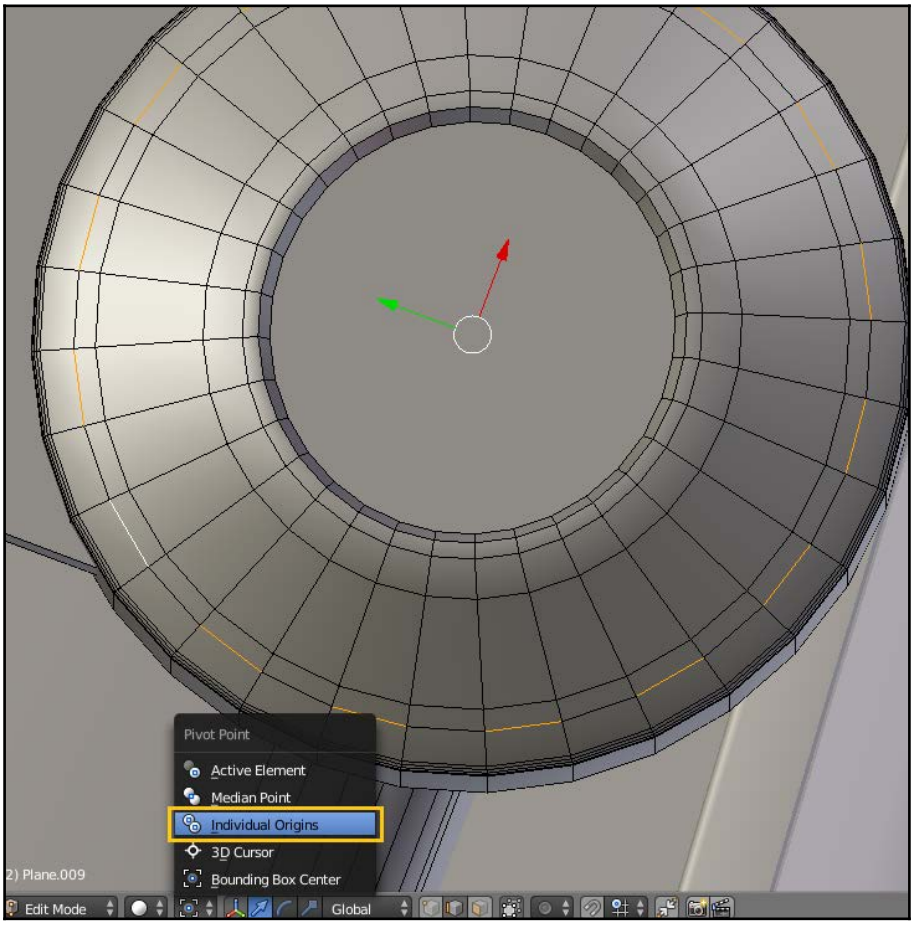
I'll do the same thing on the next section(right in front of it):



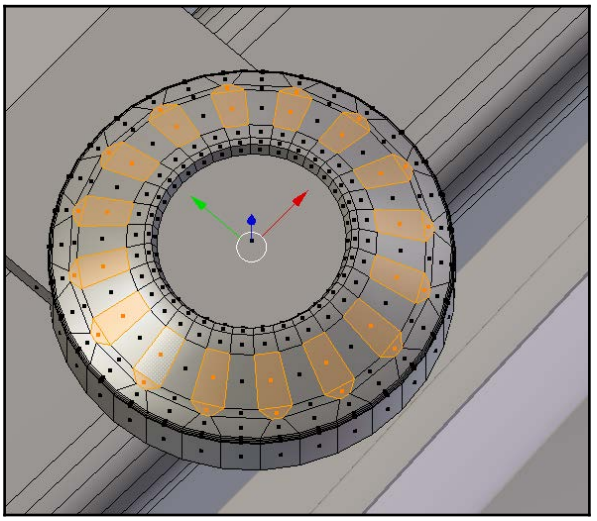
Next, we'll work on the circular parts above the engines. I'll start by adding a few loop cuts:



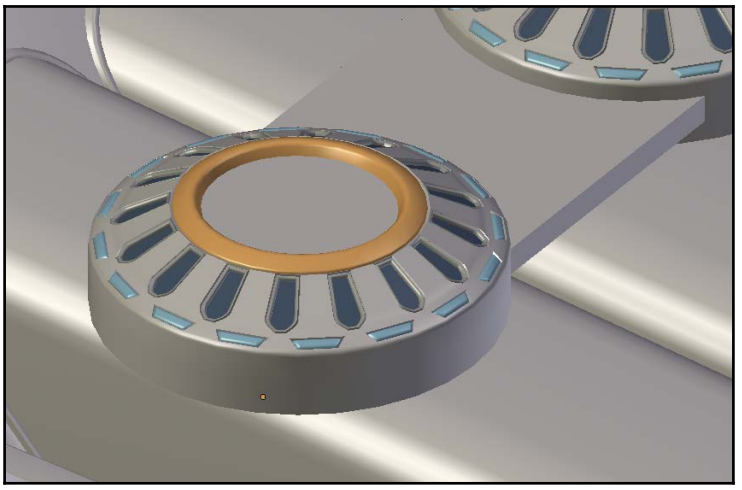
Then, setting my **Pivot Point** to **Individual Origins**, I'll select a few of the edges:



I can then shrink those edges individually and extrude the resulting polygons to create some nice shapes:



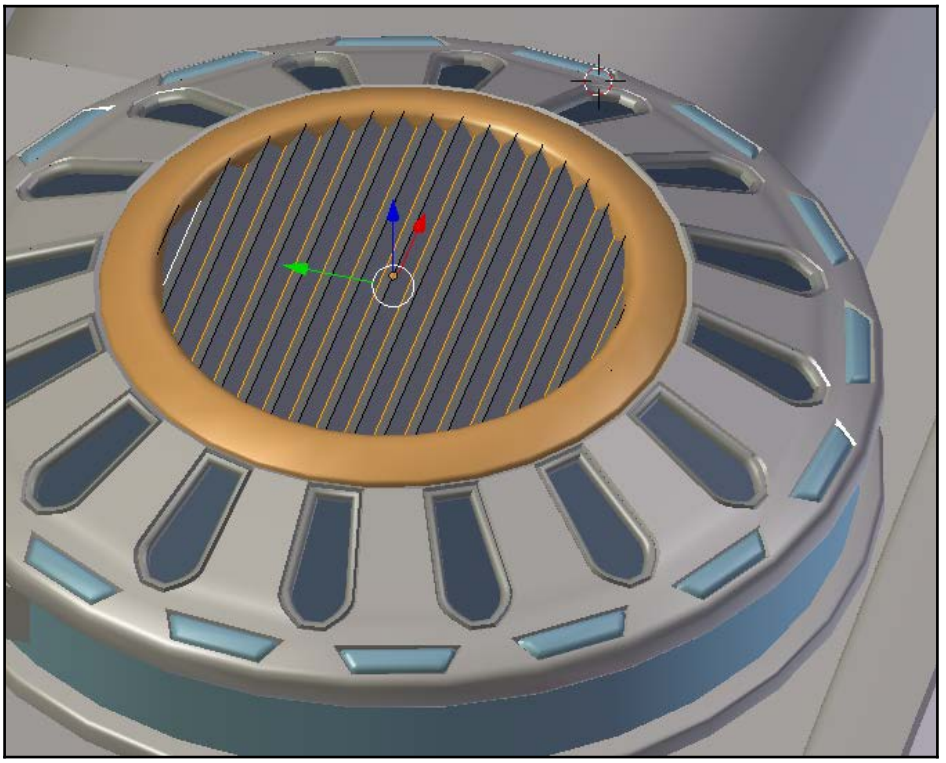
I'll add some materials to it next:



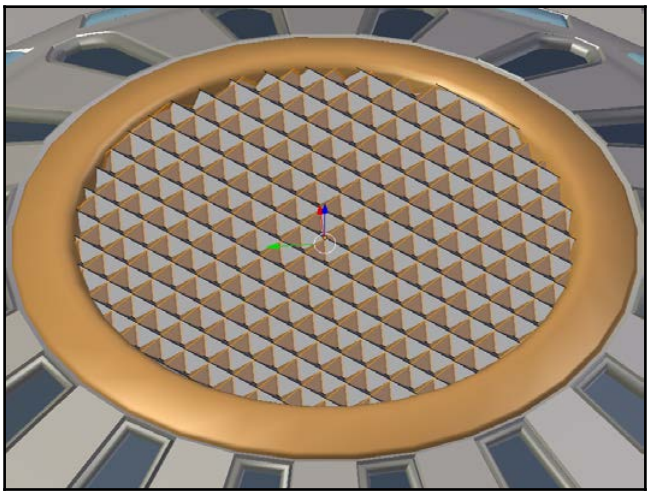
I'd like to put a little detail on the top of it, maybe something that looks like a vent. To do this, I'll just add a subdivided plane. Then, grabbing every other edge, you can pull them down a bit (you can also use the **Checker Deselect** tool from the **Select** menu to do this):



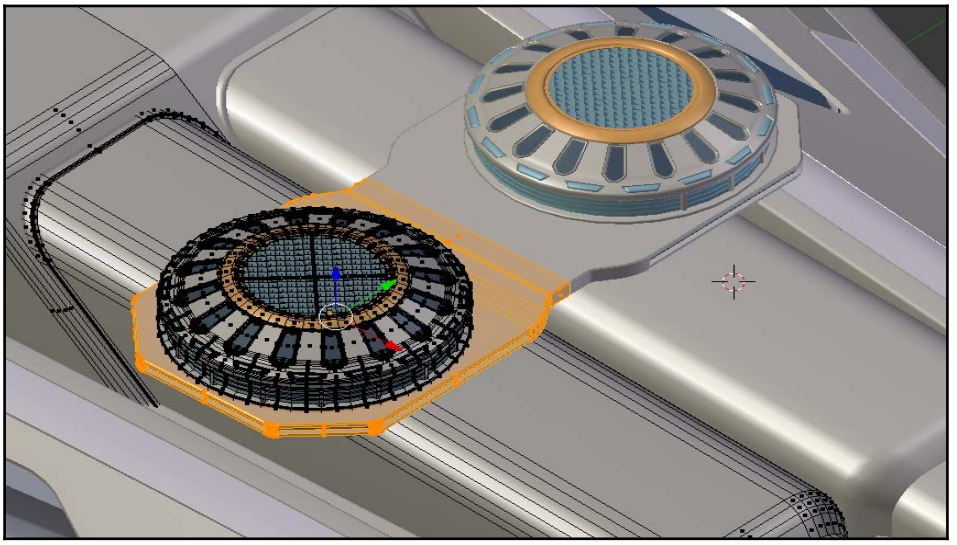
You'll need to pull the corner vertices back so that they don't stick out past the circle.



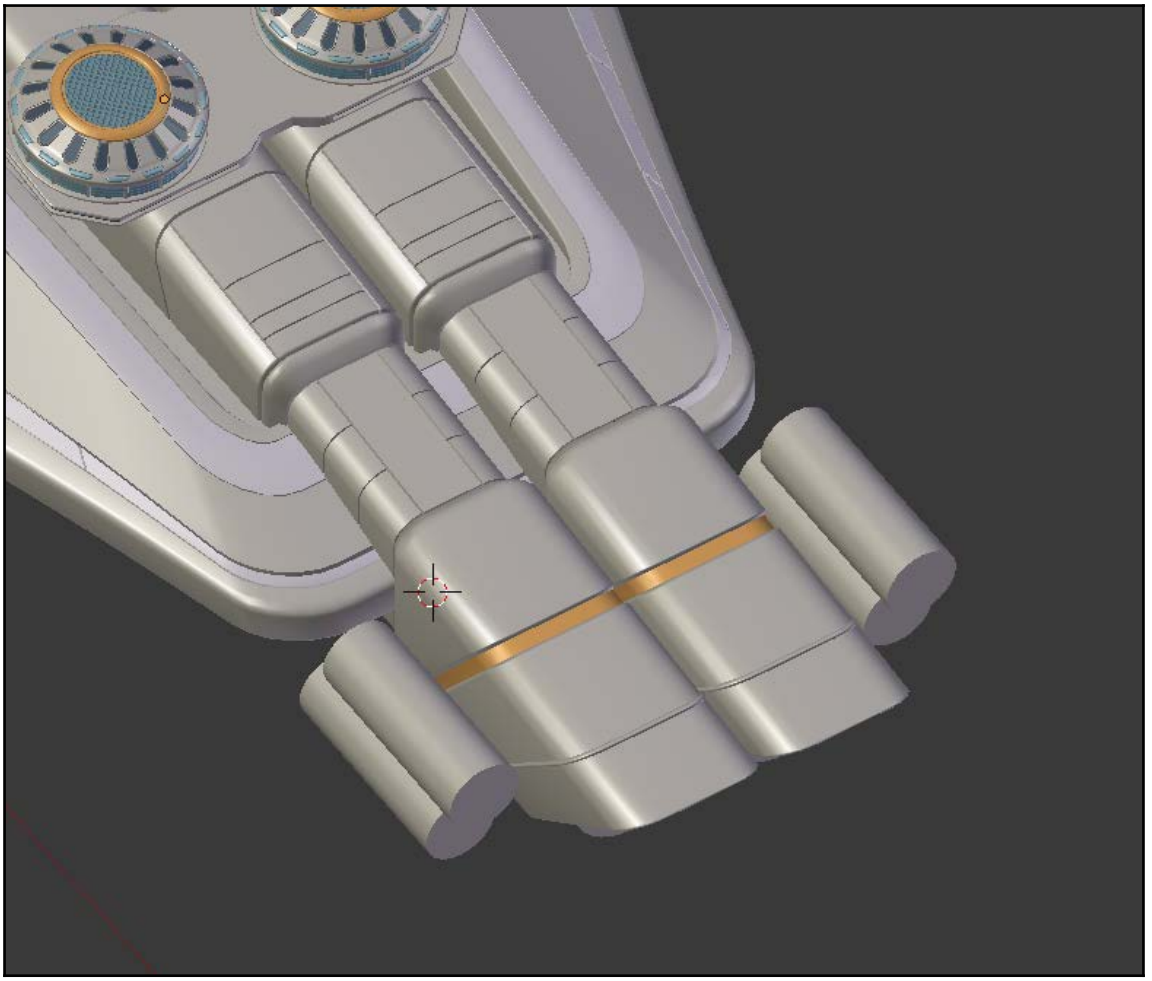
This is a quick way to add something that looks like a vent. If you want, you can even duplicate your vent and rotate it by *90* degrees:



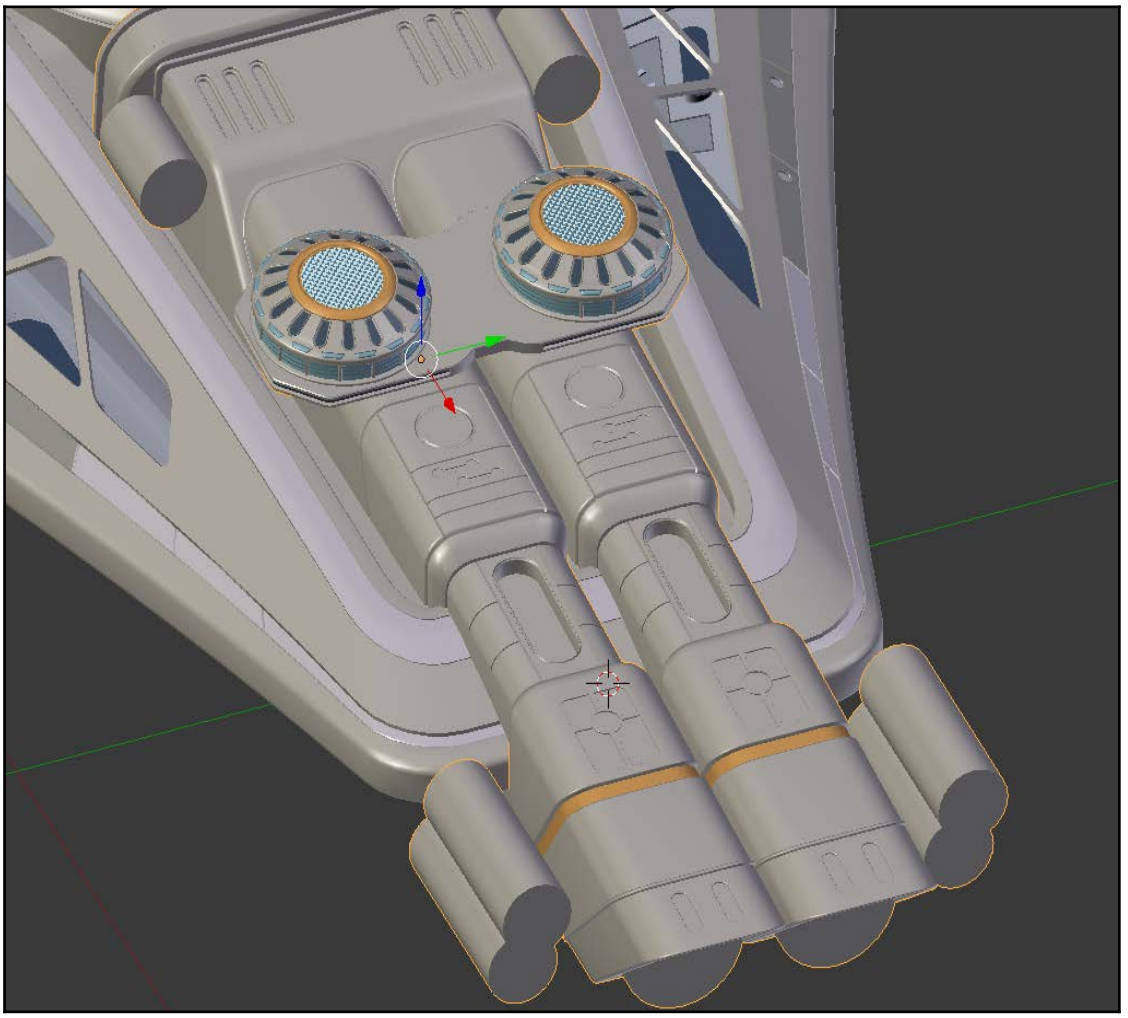
Next, let's carry out some work on the connecting piece:



Now, we can add some panel lines and cuts to the engine section:

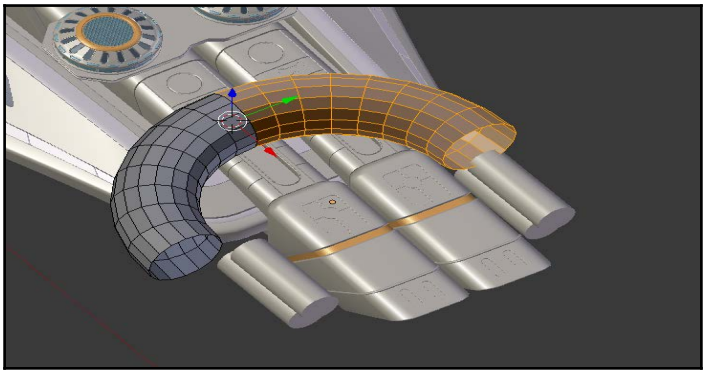


And once again, we'll add cutouts with the **Shrinkwrap** modifier:

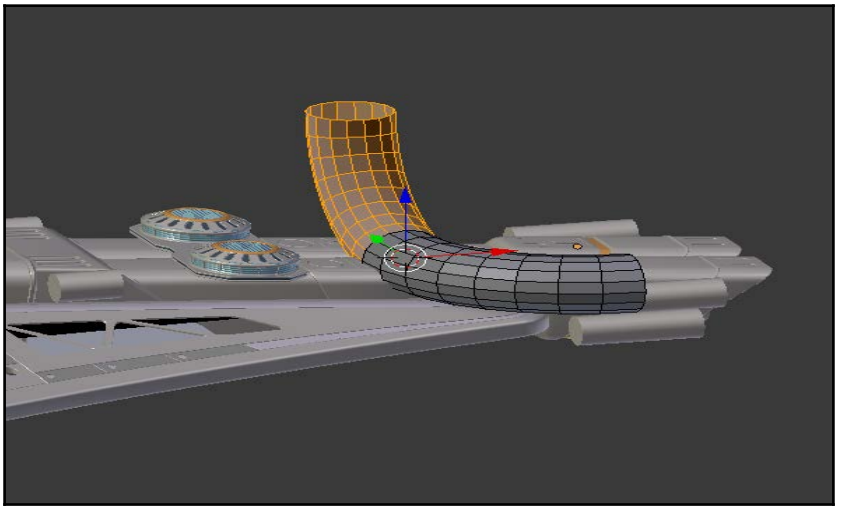


Adding detail with pipes

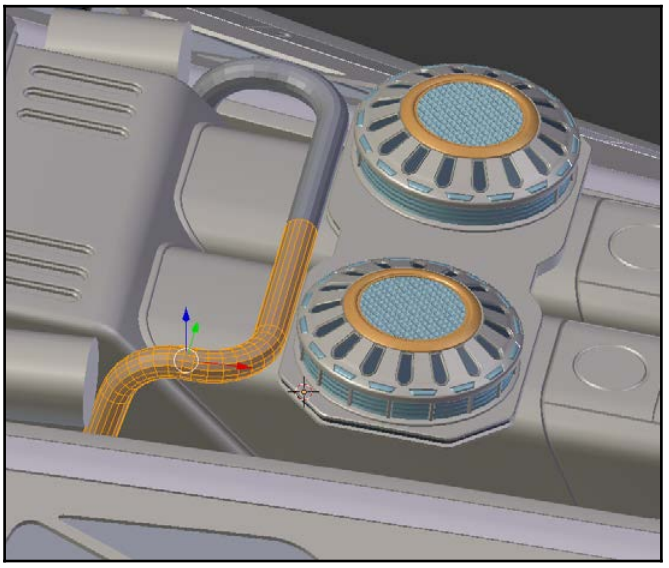
Now, we'll add some pipe sections to the engine. We'll start with **Torus object** for this, the same way we did with the gun model. Just like before, we can cut sections of the *pipe*. In this case, I've cut the front part in half, then placed the **3D Cursor** in the middle:



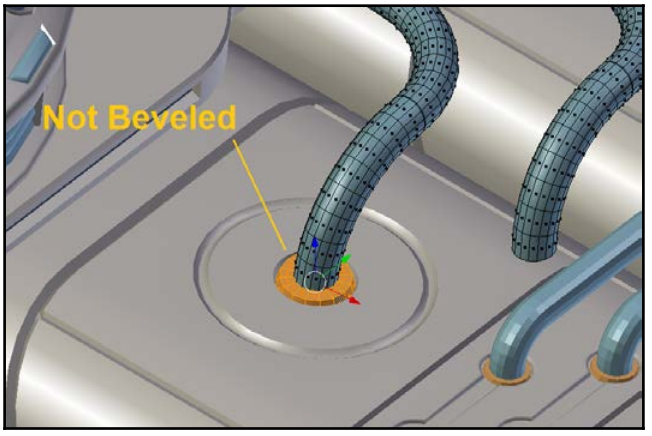
Then, I can rotate one side by 90 degrees around the **3D Cursor**, giving me a nice bend to my pipe:



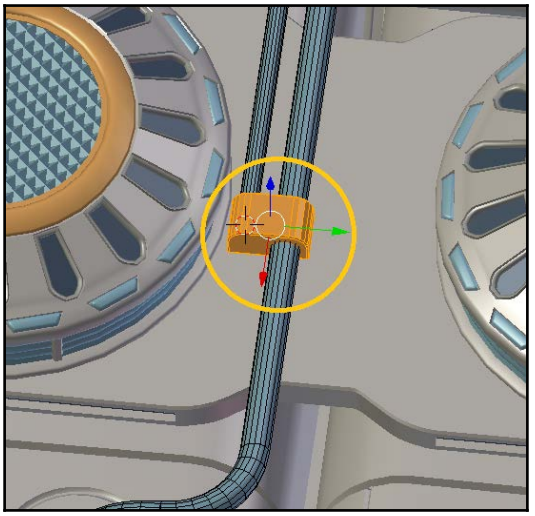
I'll first create one larger pipe section across the back of the engines, sort of like a roll bar (though of course, a spaceship wouldn't really need a roll bar):



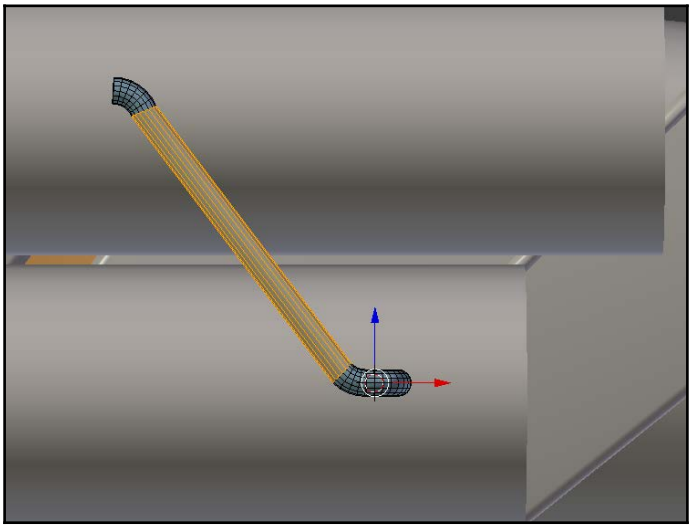
You can then create a variety of pipes going into whatever places you feel are appropriate. I'm going to add little copper connections to the ends of mine. Since they're so small, I'm not going to worry about beveling:



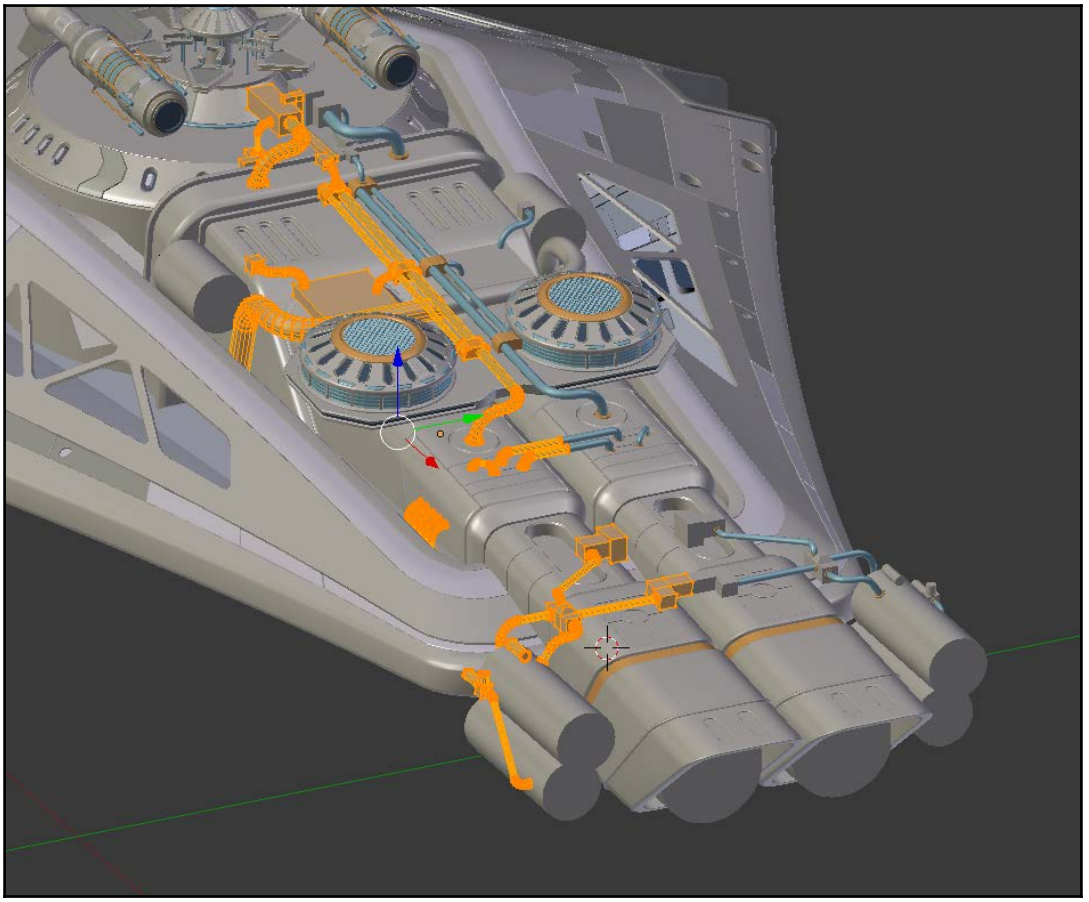
As we do this, you can also add small boxes or circles to connect various parts together:



You don't always need to use right angles, either:



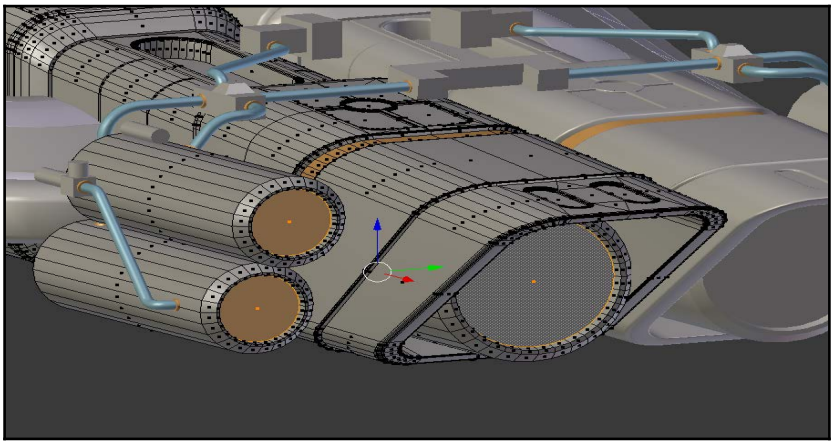
In the end, here's what I ended up doing with the pipes:



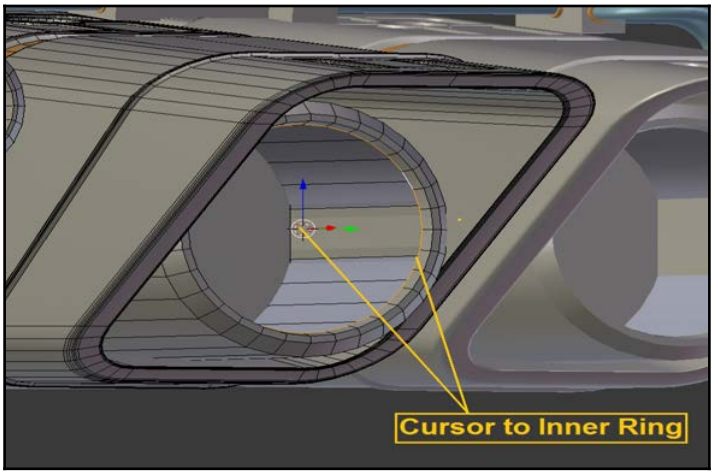
Finishing our main objects

Now that we've got our pipes in place, we'll continue working on the main parts of our model.

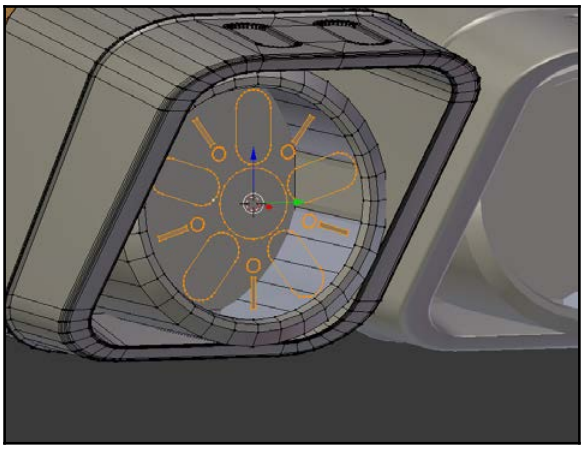
First, we'll want to work on the exhaust ports at the back of the engine section. We'll start by just beveling things down a little:



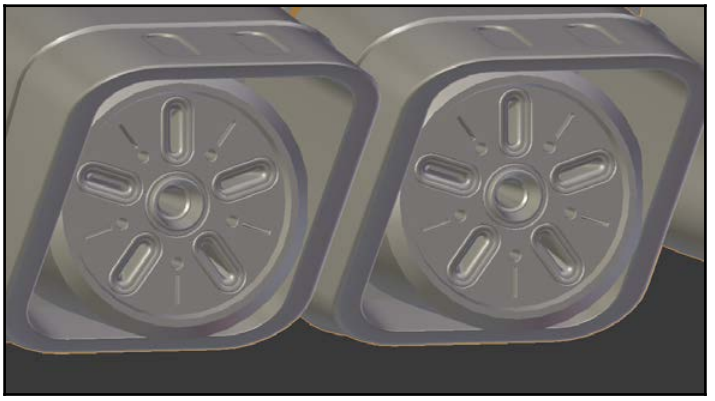
Then, we can delete those end caps and place the cursor at the center of one of the engine openings:



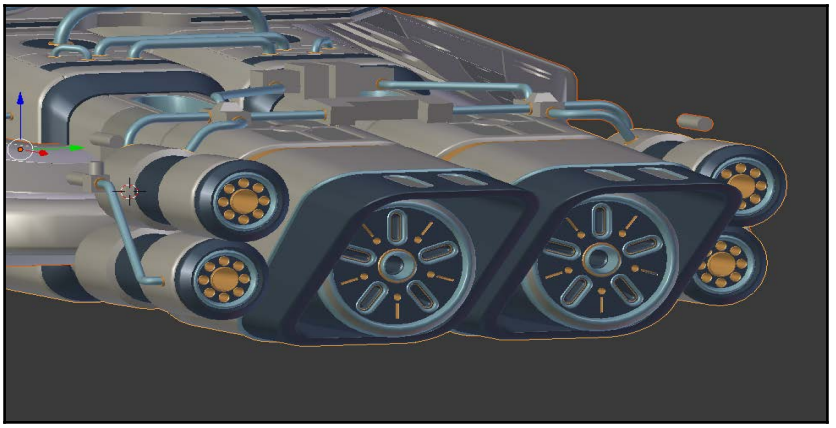
At this point, you can add shapes to cut out without using a **Shrinkwrap** modifier—the shapes are already lined up:



When you're done, you'll need to fill in those faces. Then, we'll extrude and bevel:

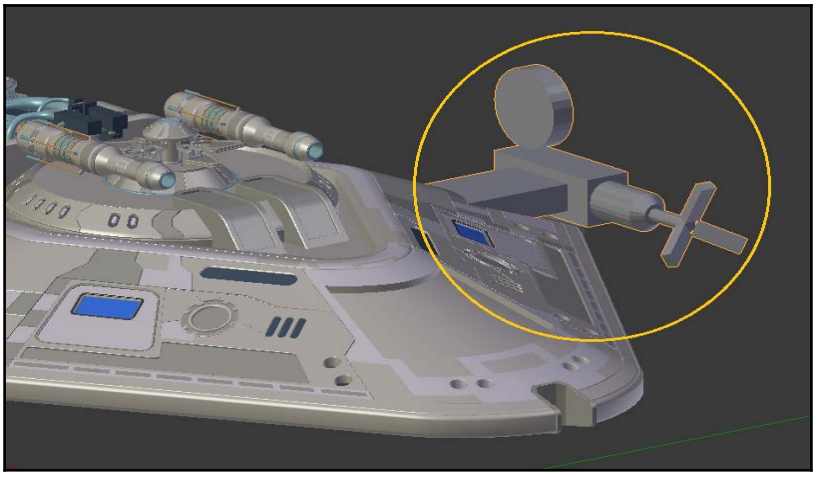


Don't forget to add materials as well:

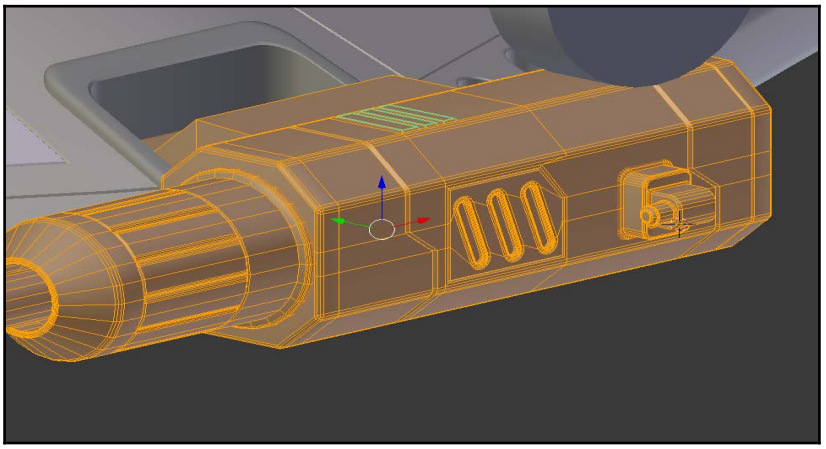


I think that's a good stopping point for now, as far as the engines go.

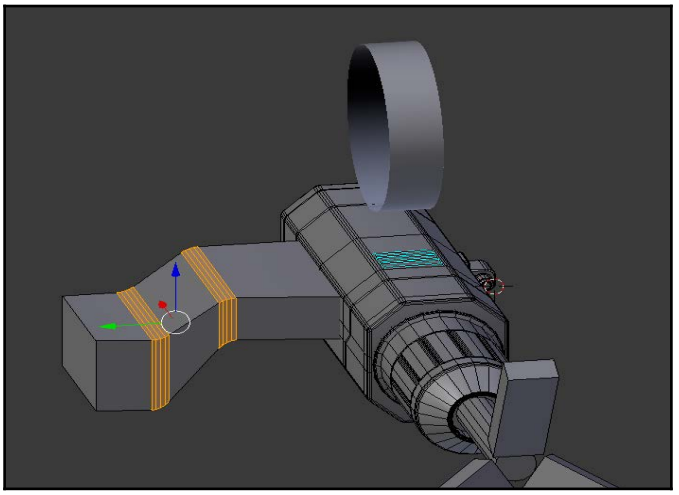
Next, we'll bring in the grappling gun:



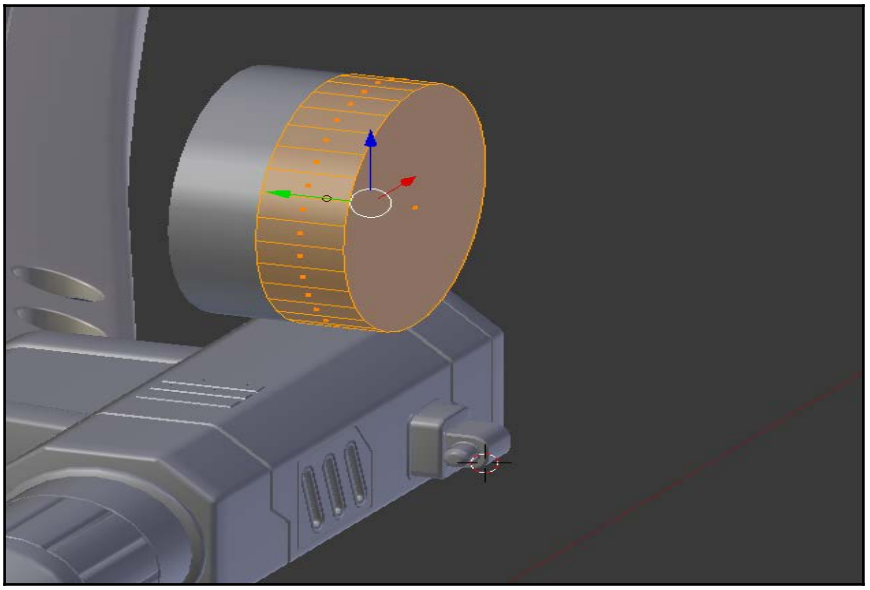
Using the same methods we used on the other sections, I'm just going to add a bit of detail to the main section of the grappling gun:



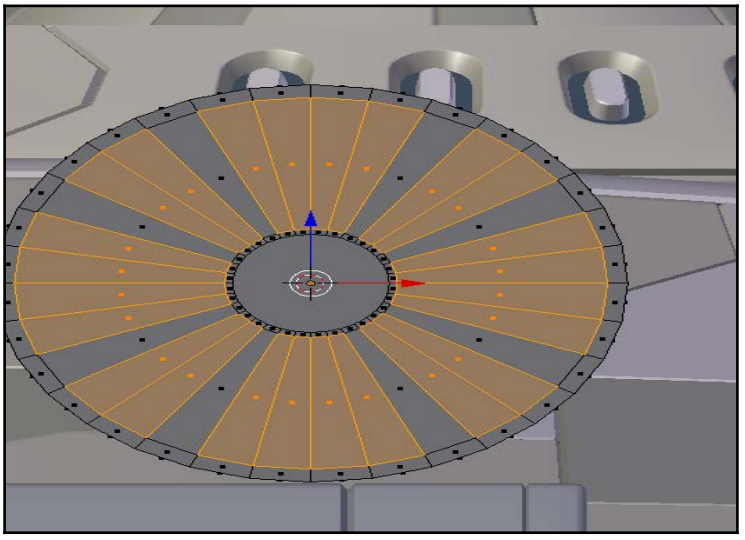
I'll also bevel the arm that connects it to the body and separate the cable reel into another object:



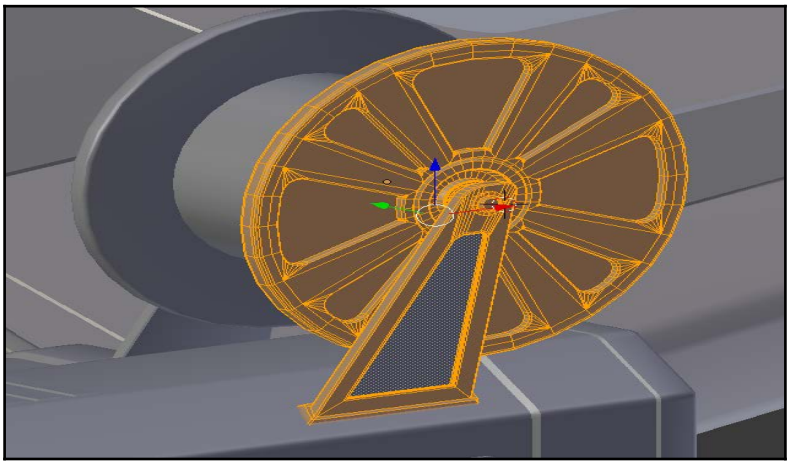
Next, we'll add a mirror modifier to it:



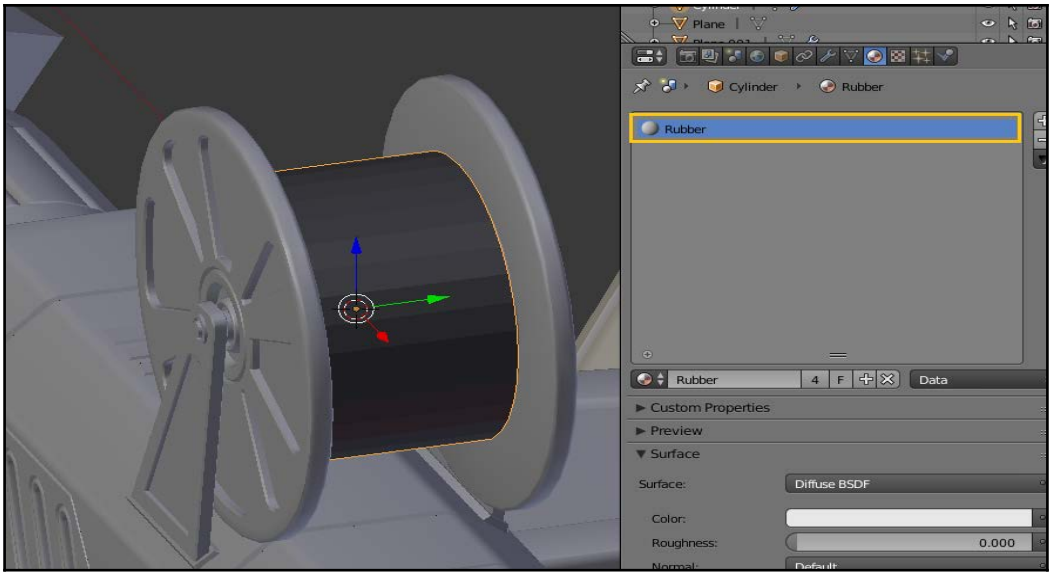
I'll pick a few faces here to create cutouts:



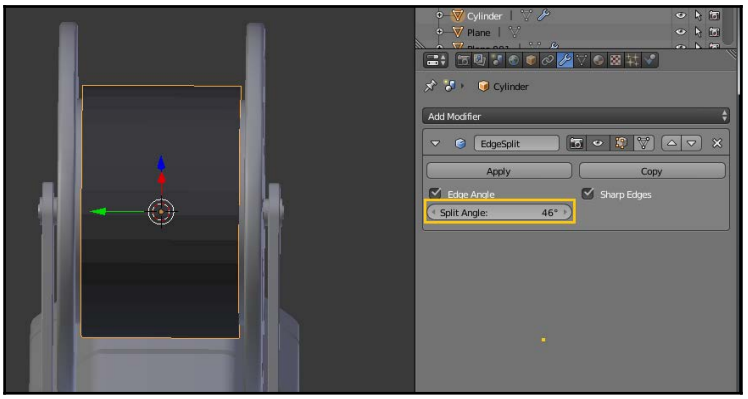
Then, we can round out those corners and add struts to connect it to the main section of the grappling gun:



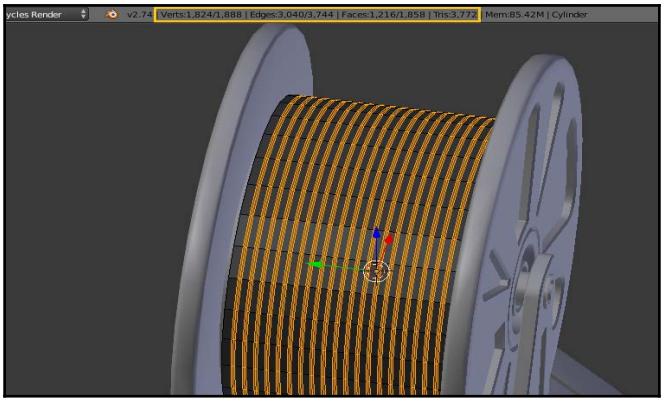
Next, I'm going to add a new cylinder and assign the **Rubber** material to it—this will be our spool of cable:



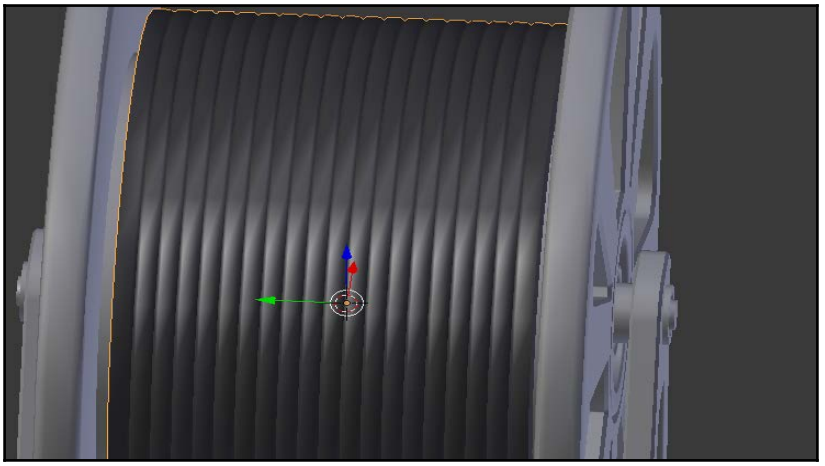
I'm going to add an **Edge Split** modifier to my cable here, but we'll set the **Split Angle** to a higher value:



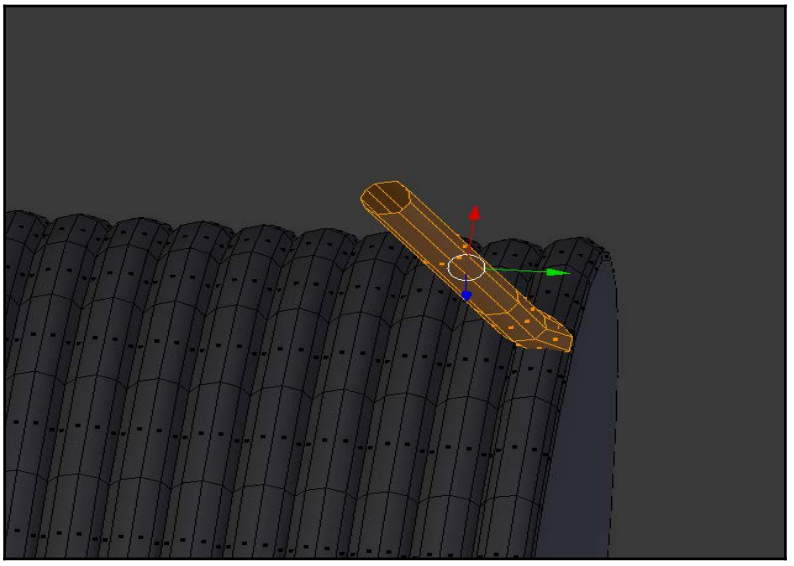
That way, edges that would normally be sharp (more than *30* degrees) will be smooth. The reason for this is just that we're going to add a lot of geometry to the reel anyway (the cable is very thin), and I don't want to have to bevel it a huge number of times:



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:

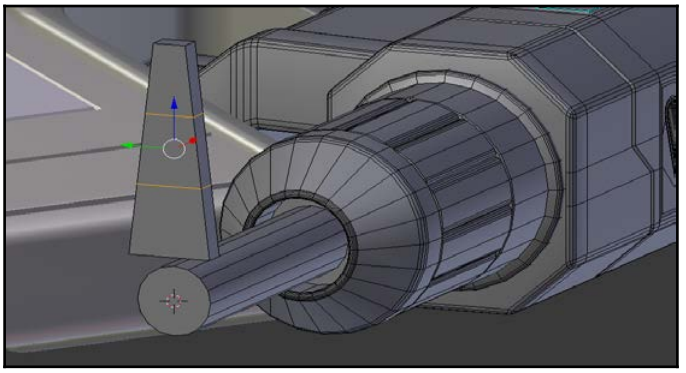


It's not perfect, but it's such a tiny detail that it will look good 99% of the time. Of course,

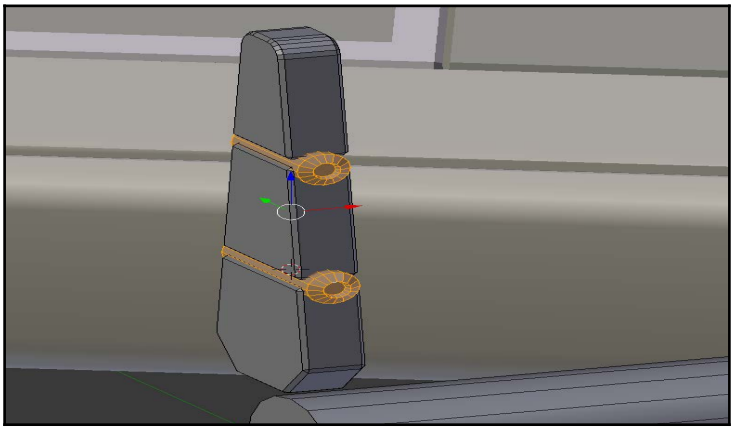
you're always welcome to improve it. One such method might be to use the Screw tool that we'll look at in just a moment.

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.

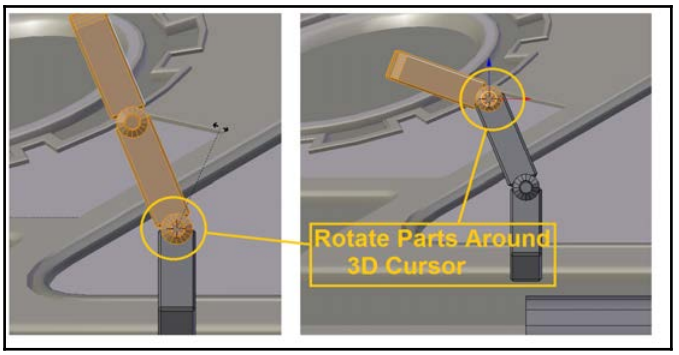
Instead, I'll add a scaled cube and just put two loop cuts in:



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:

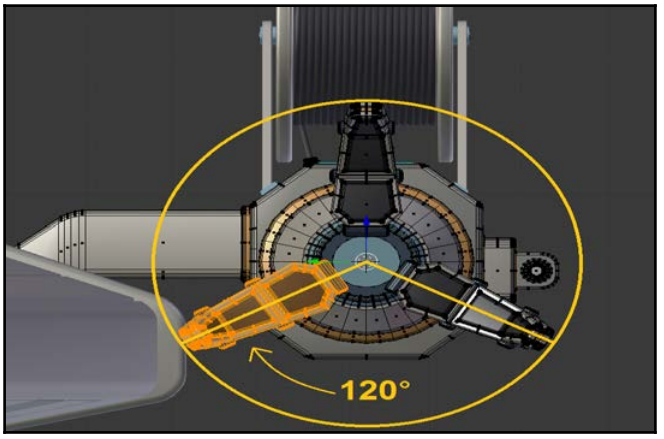


We can then set the **3D Cursor** to those smaller cylinders and rotate the parts how we'd like them:

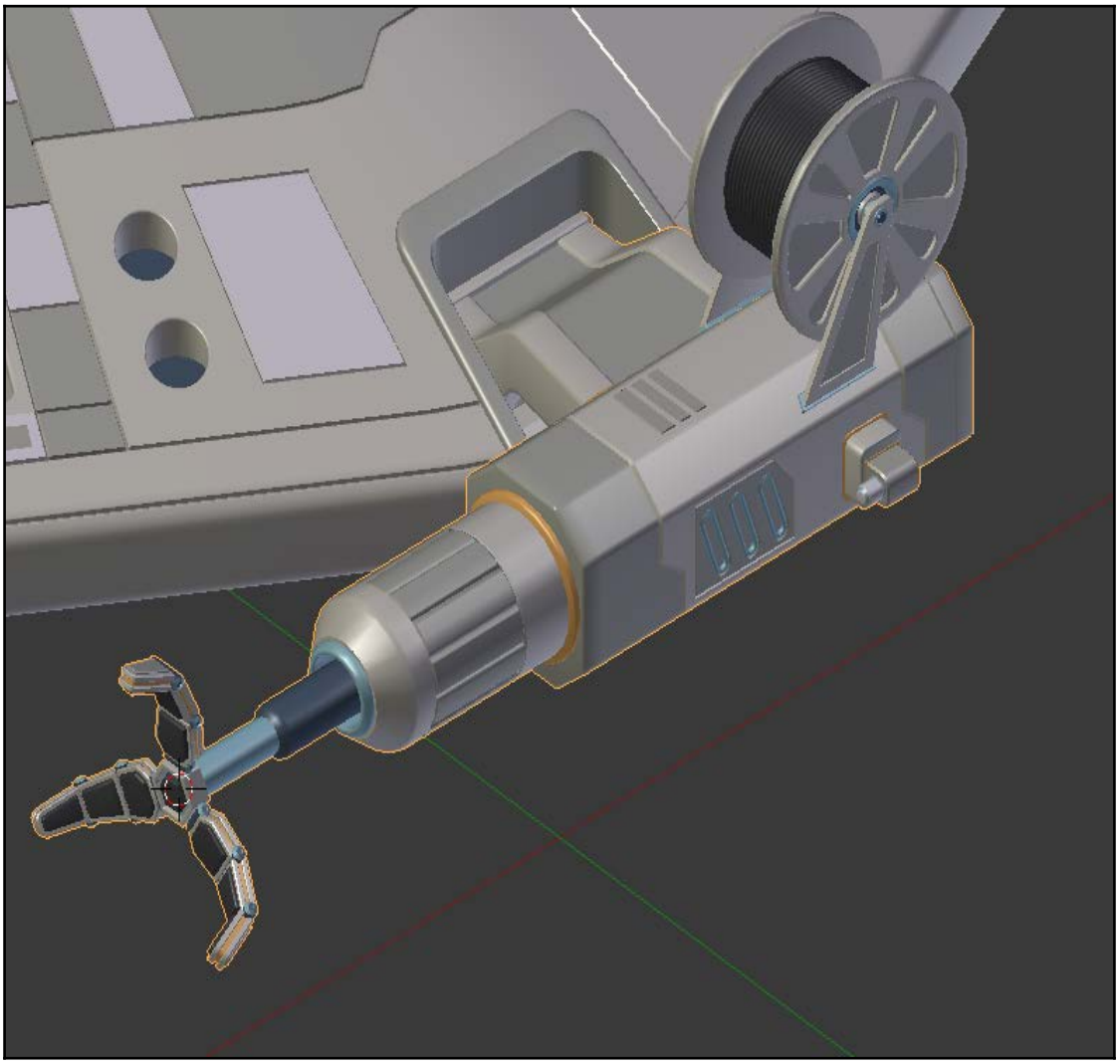


Before going further, let's add materials to our grappling gun. We'll do that now so that when we duplicate the finger, we won't have to do three times the work.

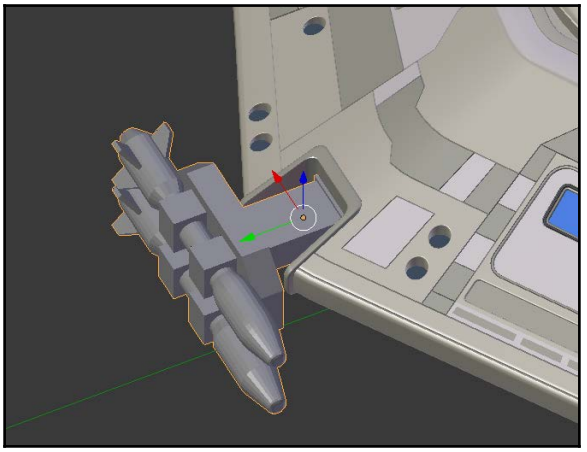
With materials added, we'll duplicate the finger of the grappling claw. Since I want three of them evenly spaced around a circle, I'll be rotating each copy around the center point by *120* degrees:



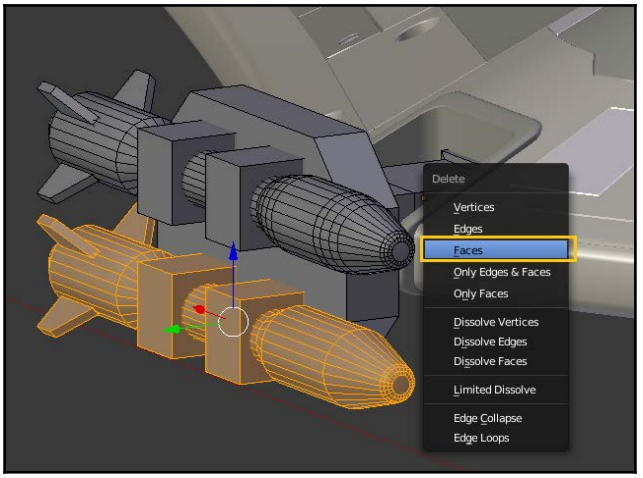
Then, we'll just do a middle part to link the three fingers, and we're done (for now) with our grappling gun:



We'll work on the missile launcher next:

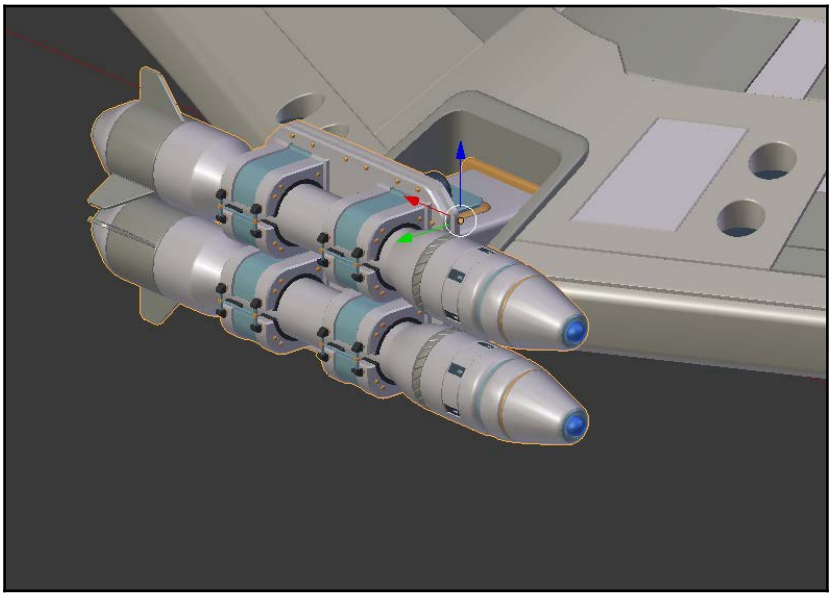


I'm going to start by just deleting one of the missiles. It's easier to just detail one of them and then duplicate it:



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:

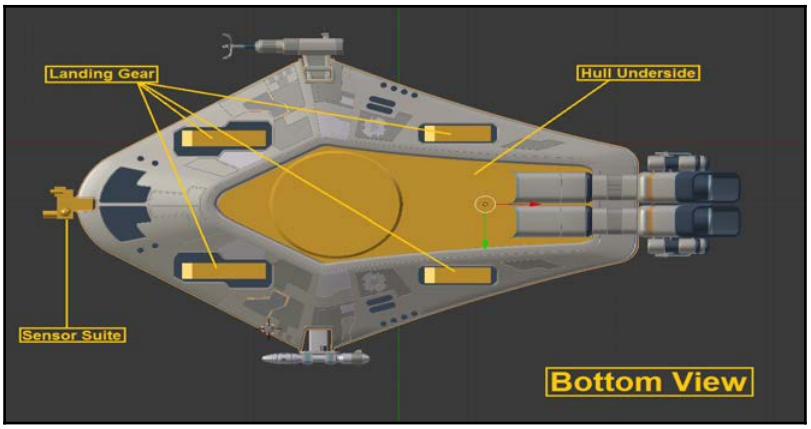




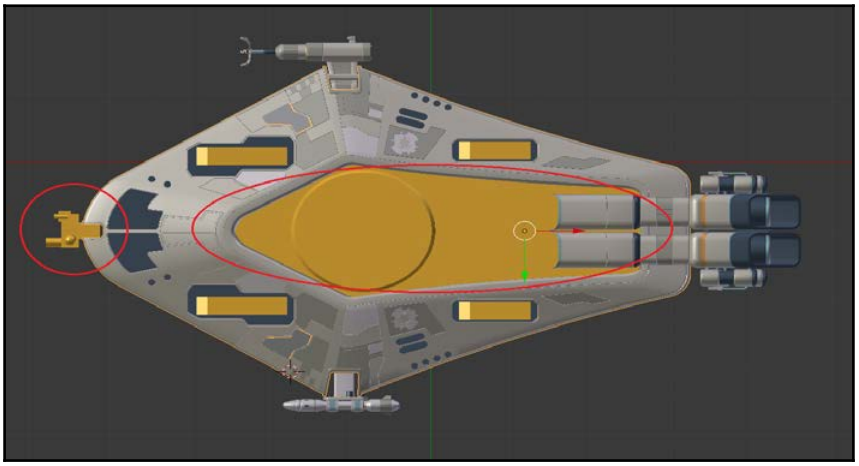
I had to be very careful with the tail fins on the missiles. Because they're so close to each other, I had to use three fins and position them very carefully so that they didn't hit each other. If you want more than three fins, you may need to move them further apart. I don't know if any real military vehicle would be designed quite this way, but I thought it looked cool. Like everything else, that's up to you.

Do it yourself – completing the body

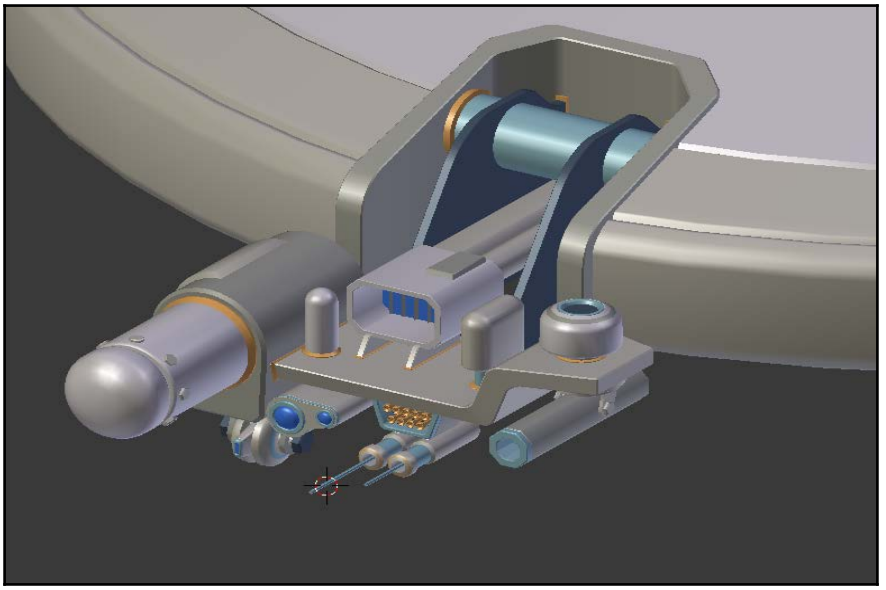
Next, let's take a look at the key areas that we have left to model:



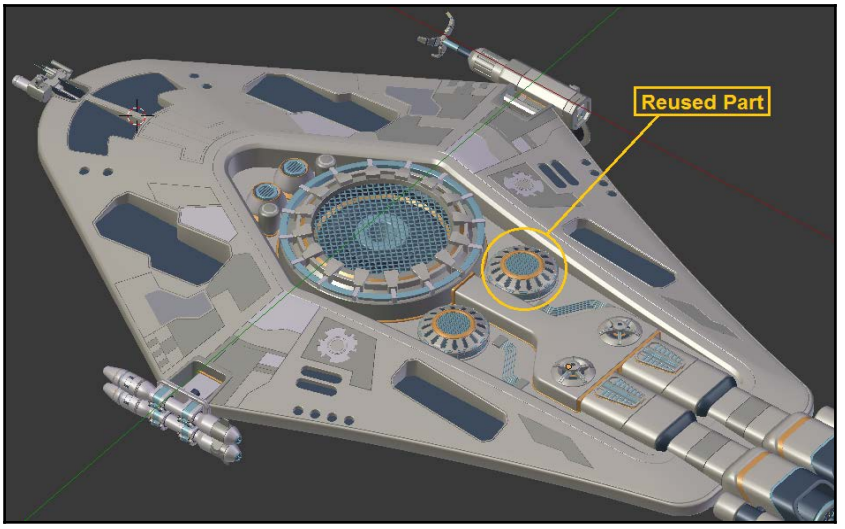
The bottom of the ship and the Sensor Suite (on the nose) are good opportunities to practice on your own. They use identical techniques to areas of the ship we've already done. Go ahead and see what you can do!



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:

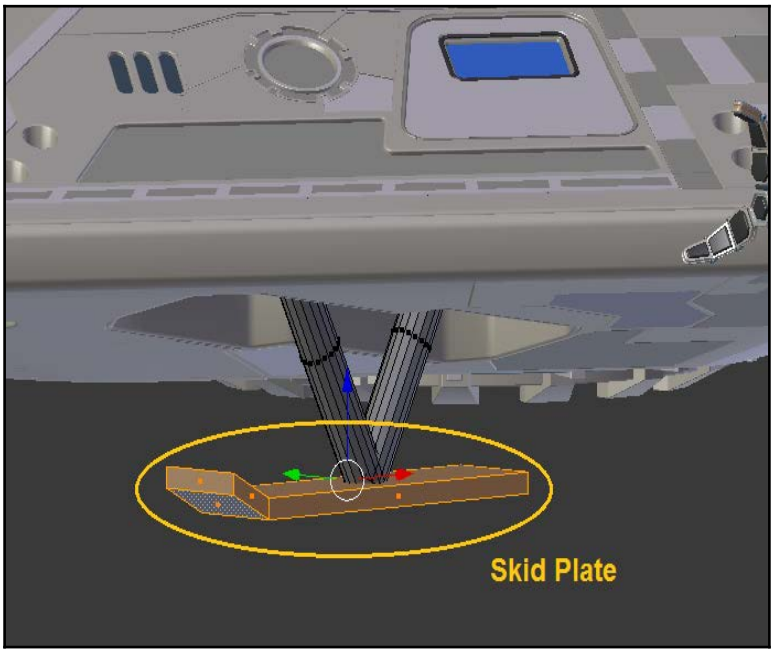


One of the nice things about a project like this is that you can start to copy parts from one area to another. It's unlikely that both the top and bottom of the ship would be shown in the same render (or shot), so you can probably get away with borrowing quite a bit. Even if you did see them simultaneously, it's not unreasonable to think that a ship would have more than one of certain components.

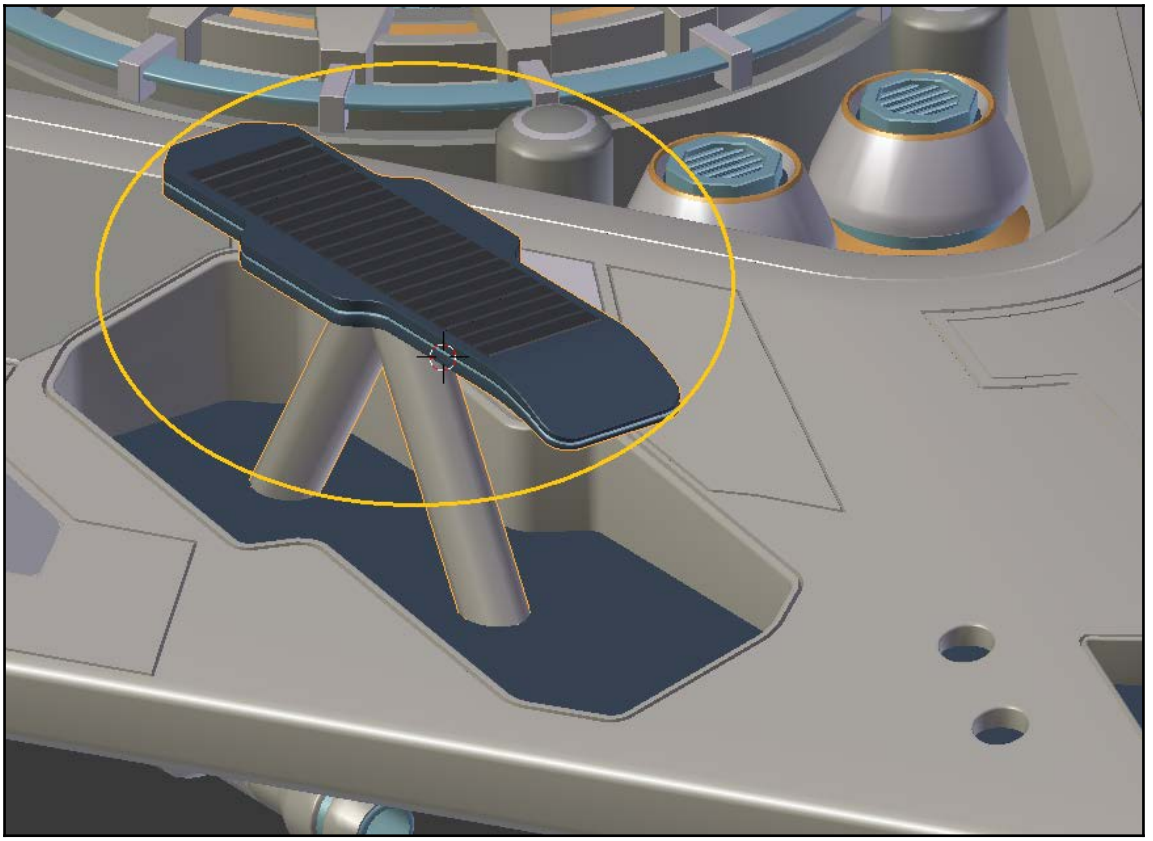
Of course, that's just a way to make things quicker (and easier). If you'd like everything to be 100% original, you're certainly free to do so.

Building the landing gear

We'll do the landing struts together, but you can feel free to finish off the actual skids yourself:

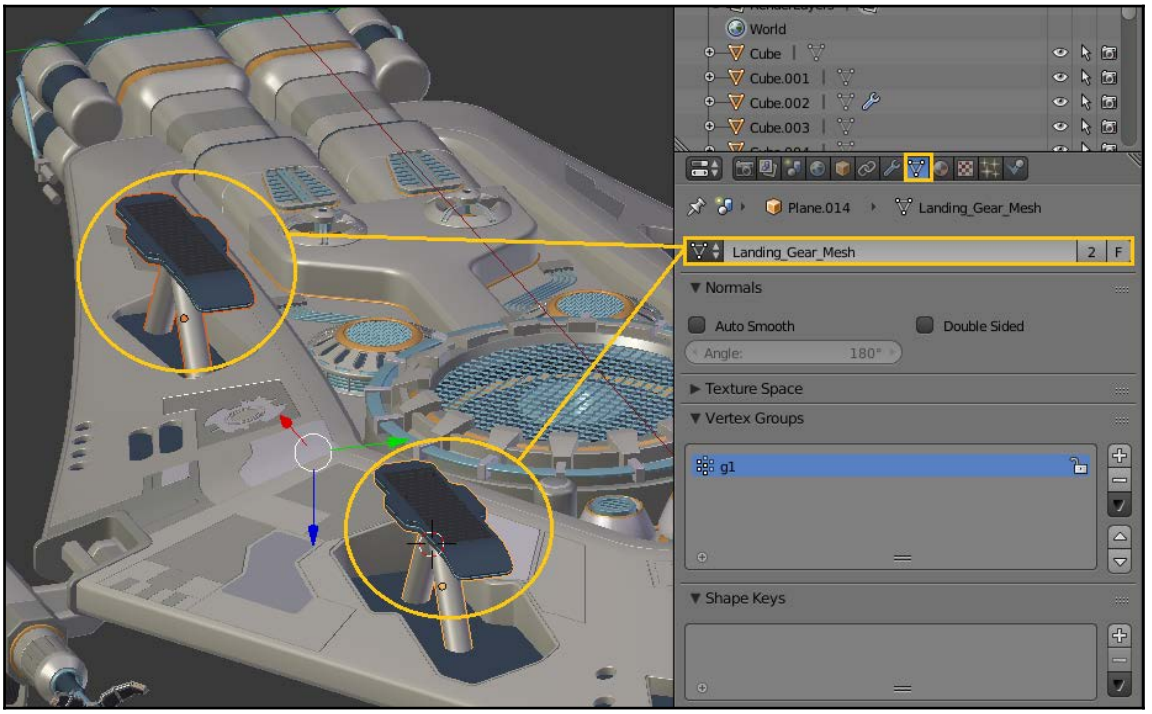


I kept mine pretty simple, compared to other parts of the ship:



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.

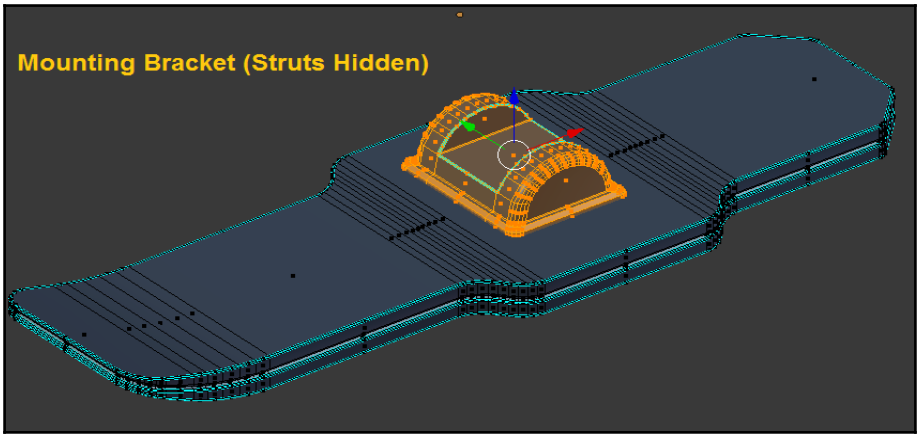
Make a copy of the landing gear part and move it to the rear section (or front, if you modeled the rear). Then, under your **Mesh** tab, you can assign both of these objects the same mesh data:



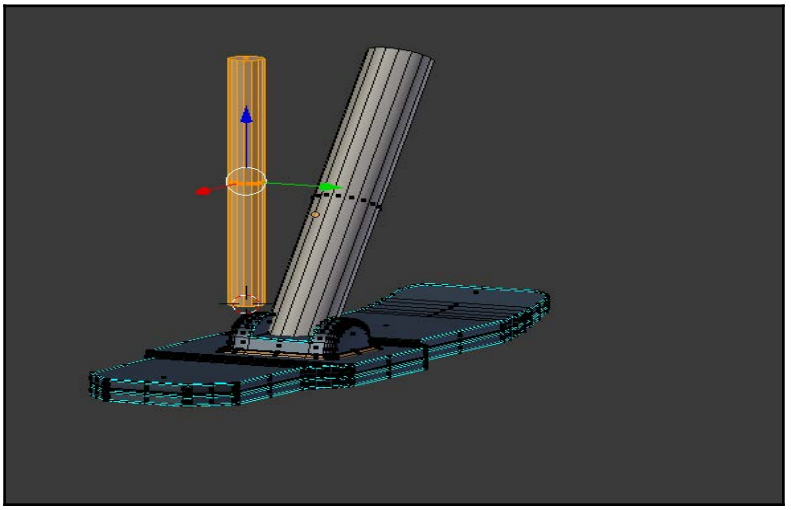
Now, whenever you make a change to one of them, the change will carry over to the other as well.

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.

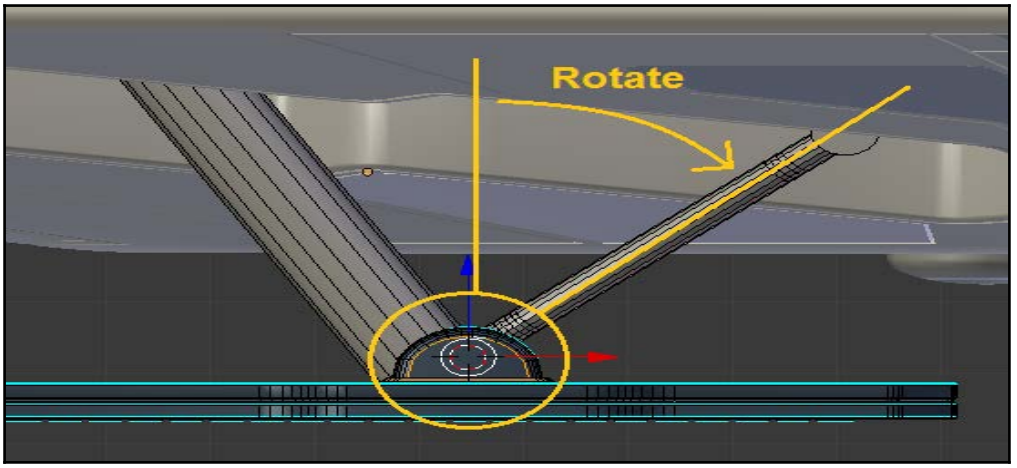
The first detail we'll add is a mounting bracket for our struts to go on:



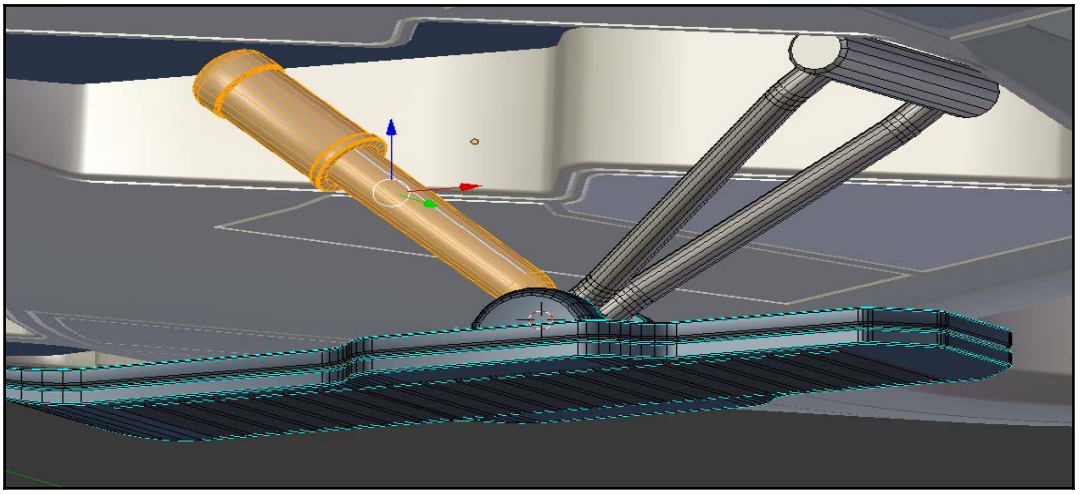
Then, we'll add a small cylinder (the large one, at this point, is just a placeholder):



We'll rotate it just a bit:

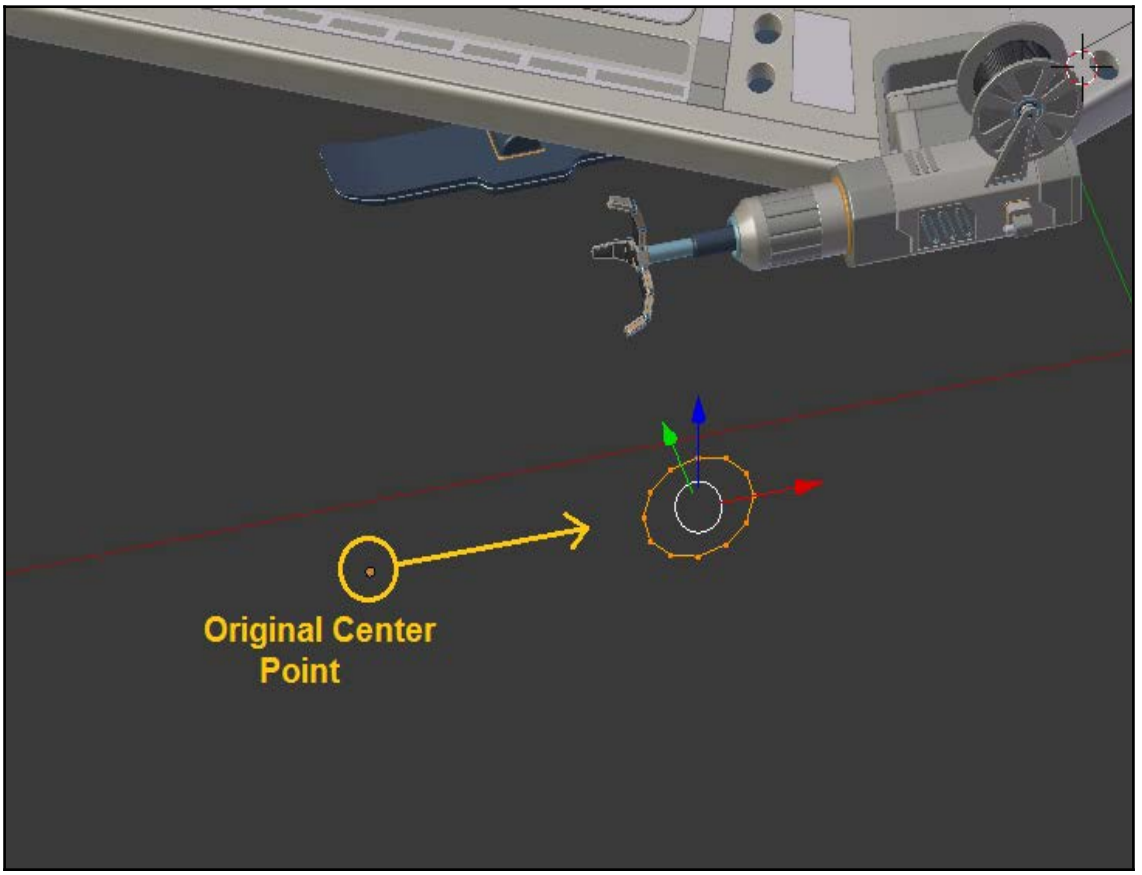


From this, it's pretty easy to create a rear mounting piece. Once you've done that, go ahead and add a shock absorber for the front (leave room for the springs, which we'll add next):

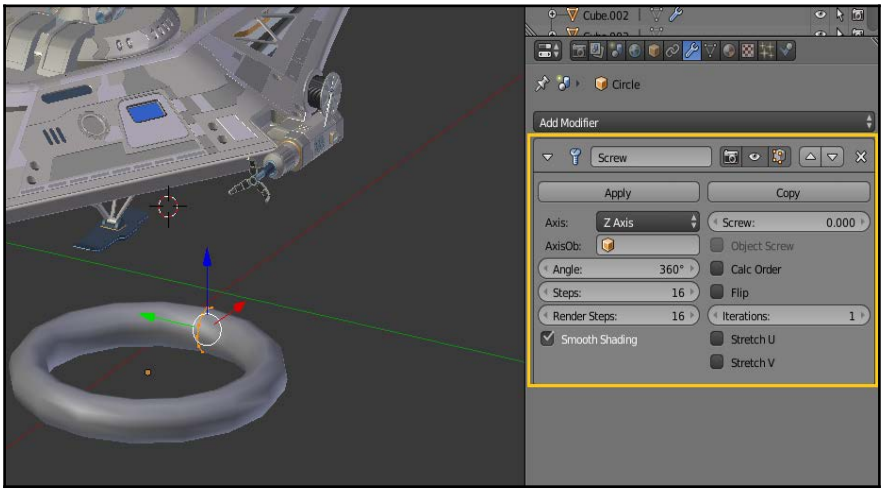


To create the spring, we'll start with a small (12-sided) circle. We'll make it that small because, just like the cable reel on the grabbling gun, there will naturally be a lot of geometry and we want to keep the polygon count as low as we can.

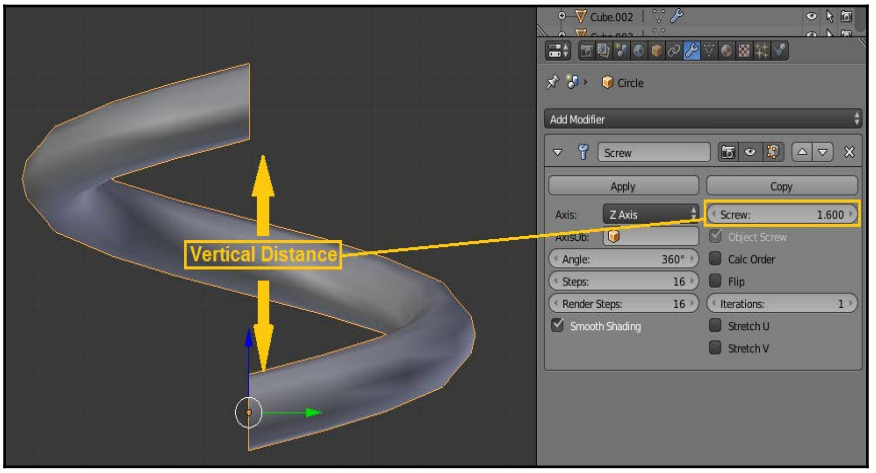
Then, in **Edit Mode**, move the whole circle away from its original center point:



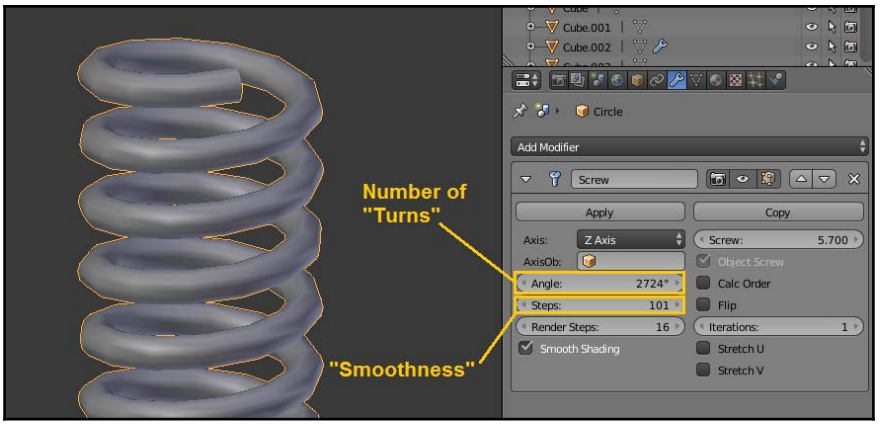
Having done that, you can now add a **Screw** modifier. Right away, you'll see the effect:



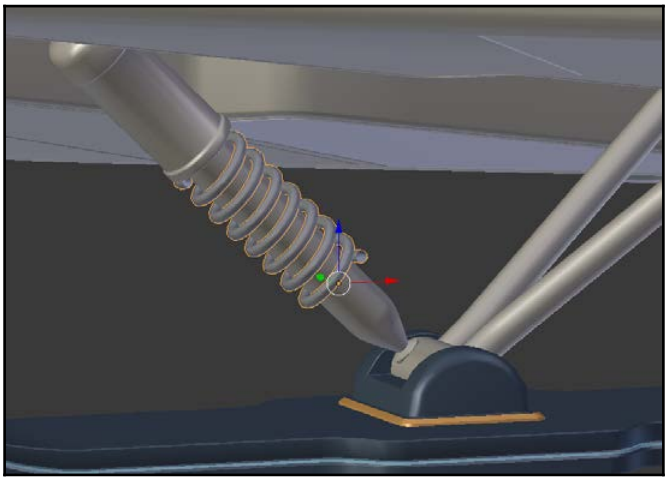
There are a couple of settings you'll want to make note of here. The **Screw** value controls the vertical gap or distance of your spring:



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):

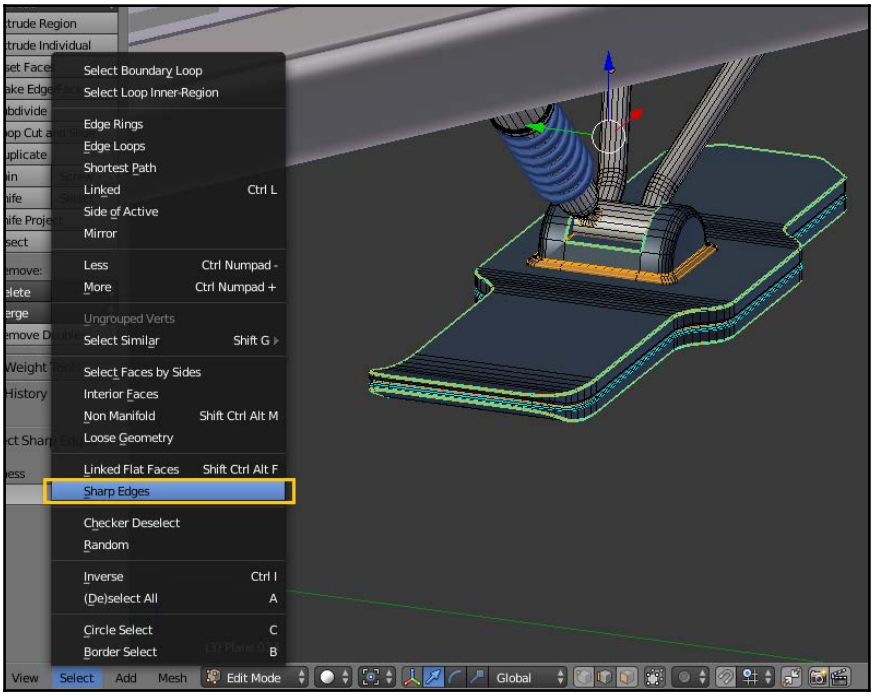




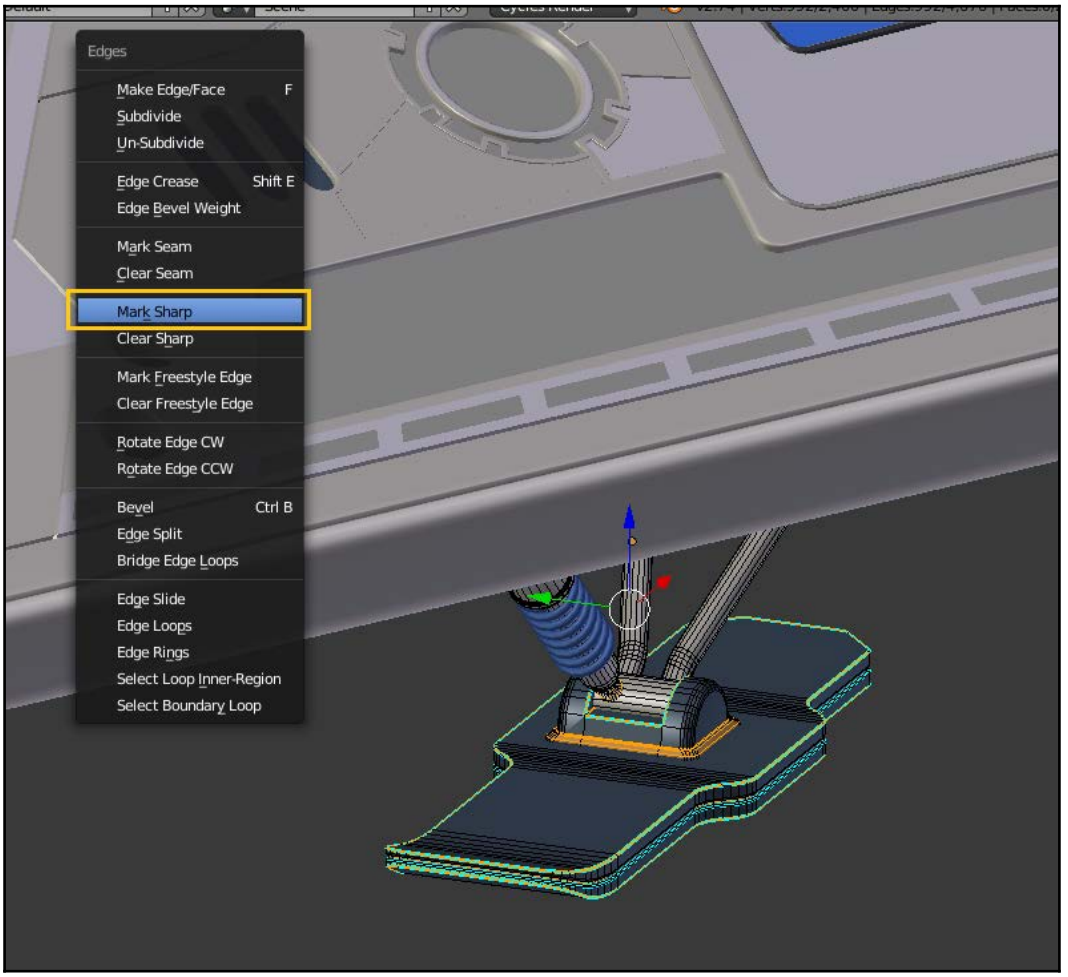
None of my existing materials seemed right for the spring. So, I went ahead and added one that I called Blue Plastic.

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.

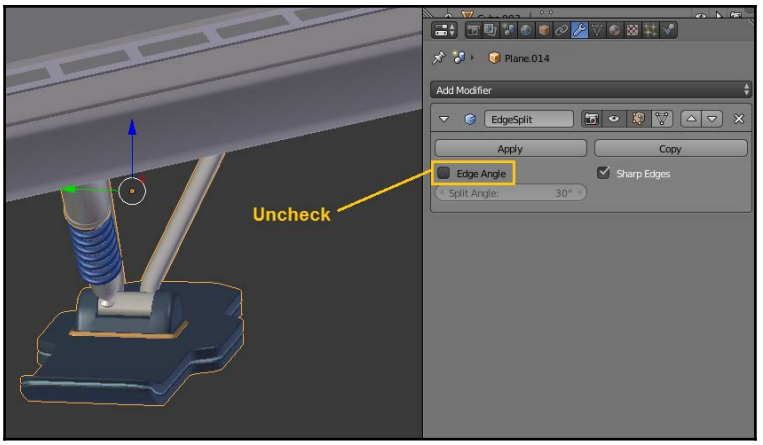
Instead, we'll go to our shock absorber. Using the **Select** menu, we'll pick the **Sharp Edges** option:



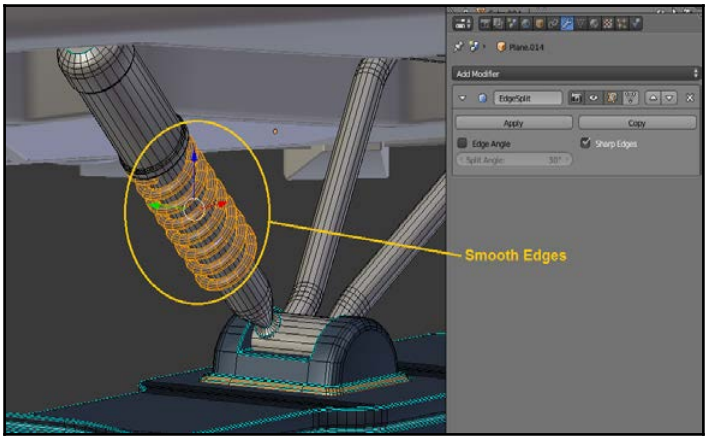
By default, it will select all edges with an angle of 30 degrees or higher. Once you do that, go ahead and mark those edges as sharp:



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 (uncheck it) and the landing gear remains exactly the same:

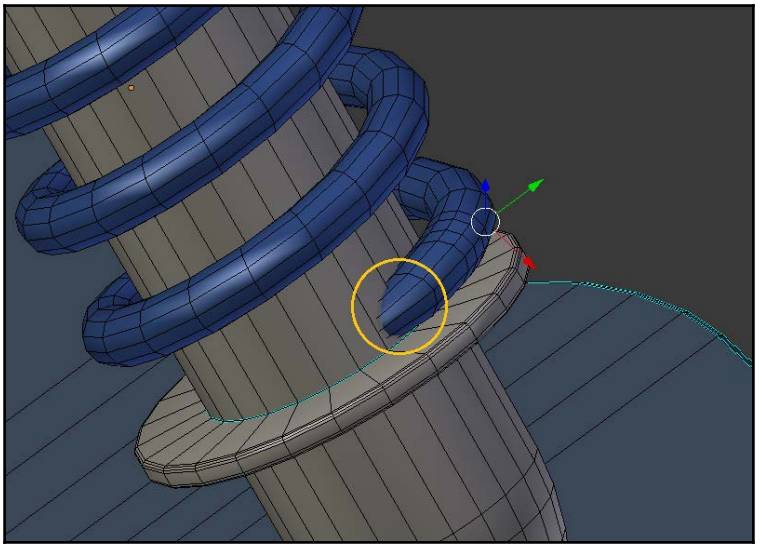


Now, you can join the spring to it without a problem:

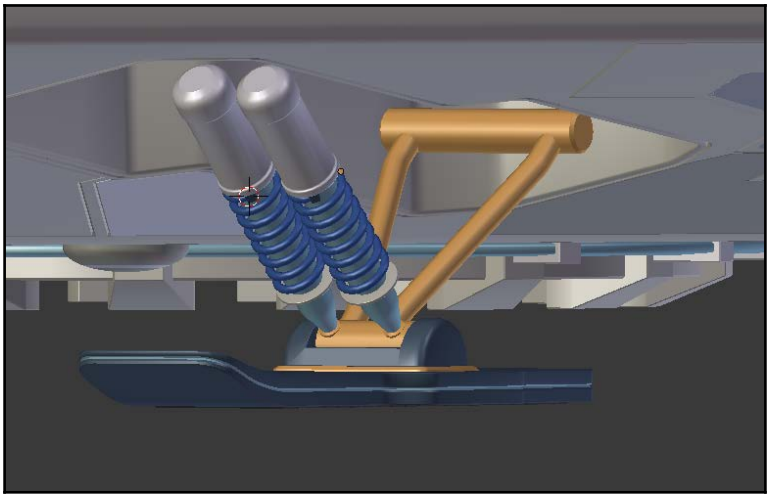


Of course, this does mean that when you create new edges in your landing gear, you'll now have to mark them as sharp. Alternatively, you can keep the **Edge Angle** option selected and just turn it up to 46 degrees—your choice.

Next, we'll just pull the ends of our spring in a little so that they don't stick out:



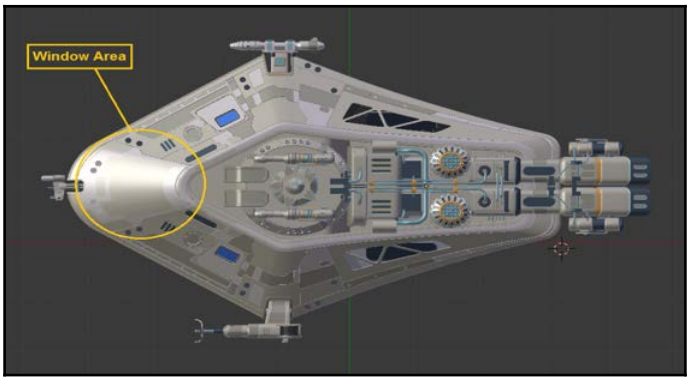
And maybe we'll duplicate it. This is a big, heavy vehicle, after all, so maybe it needs multiple shock absorbers:



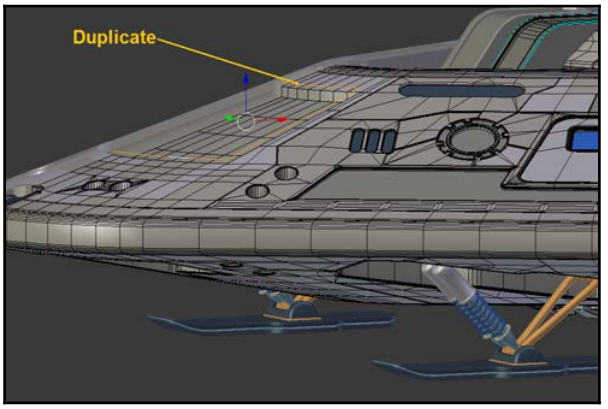
That's a good place to leave our landing gear for now, I think.

Adding the cockpit

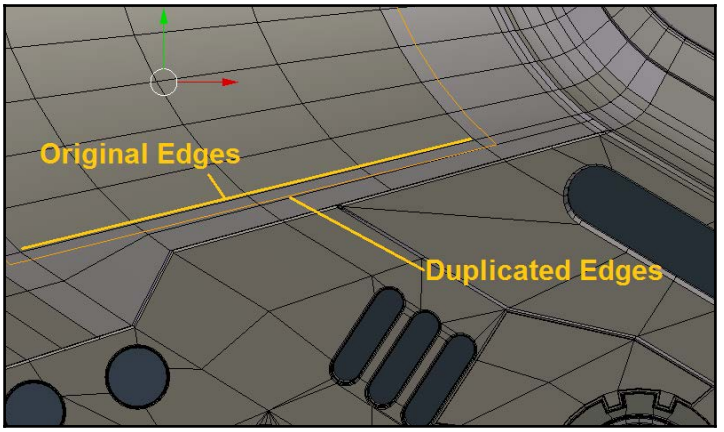
One of the last major areas we need to do is our front window:



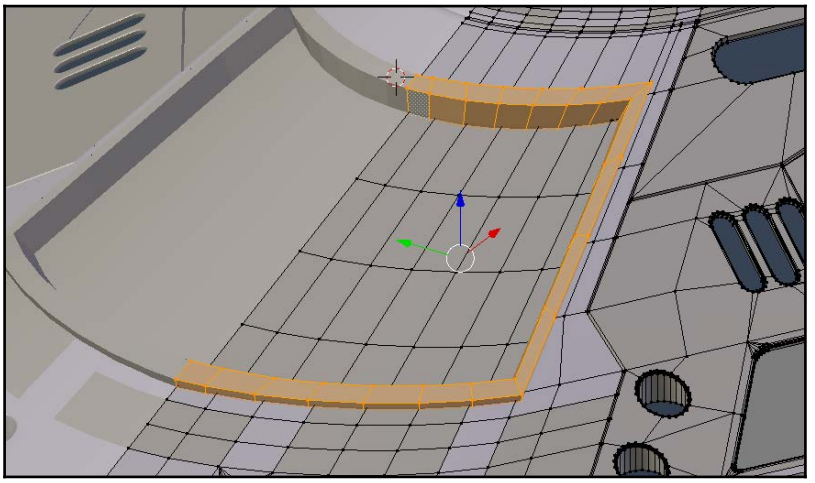
There are a number of ways we could do this, but let's try something a bit different. We can just duplicate the ring of edges that makes up the window area:



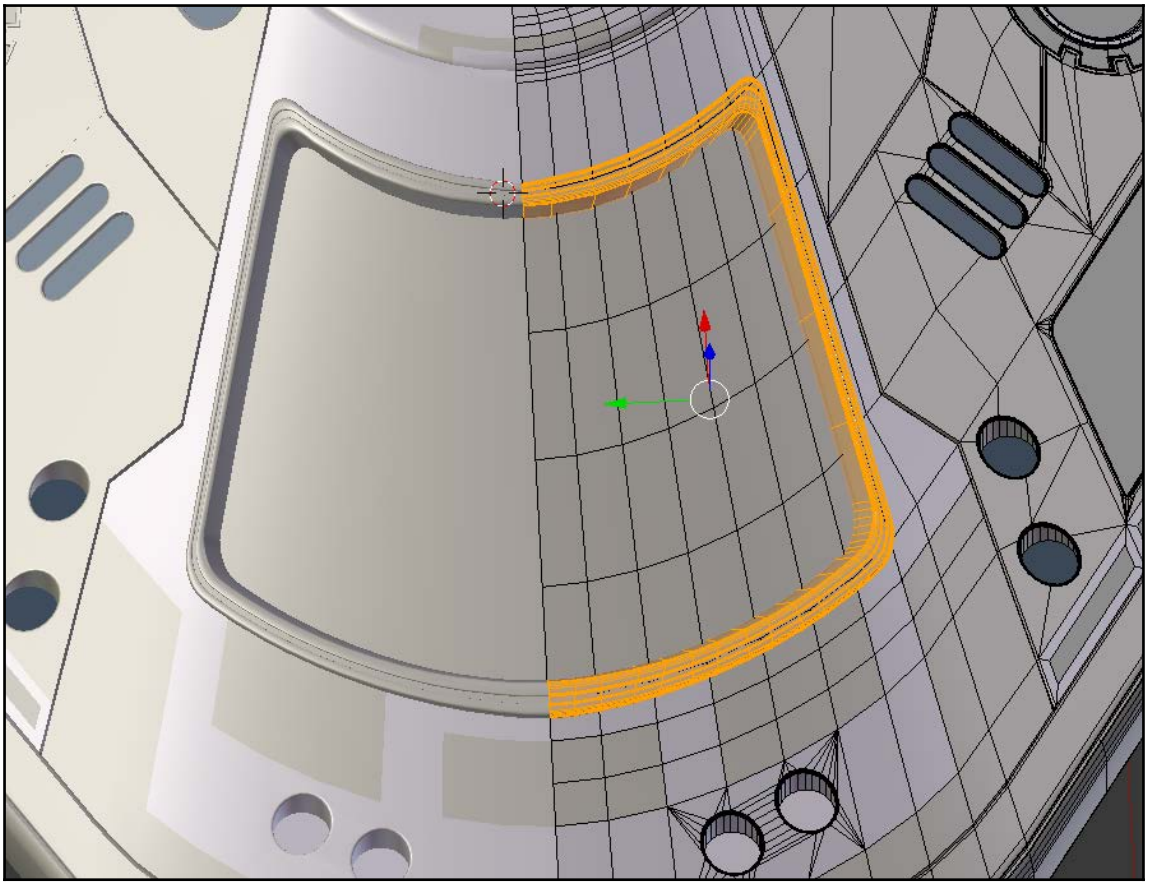
Then, we'll scale those duplicate edges up just a little:



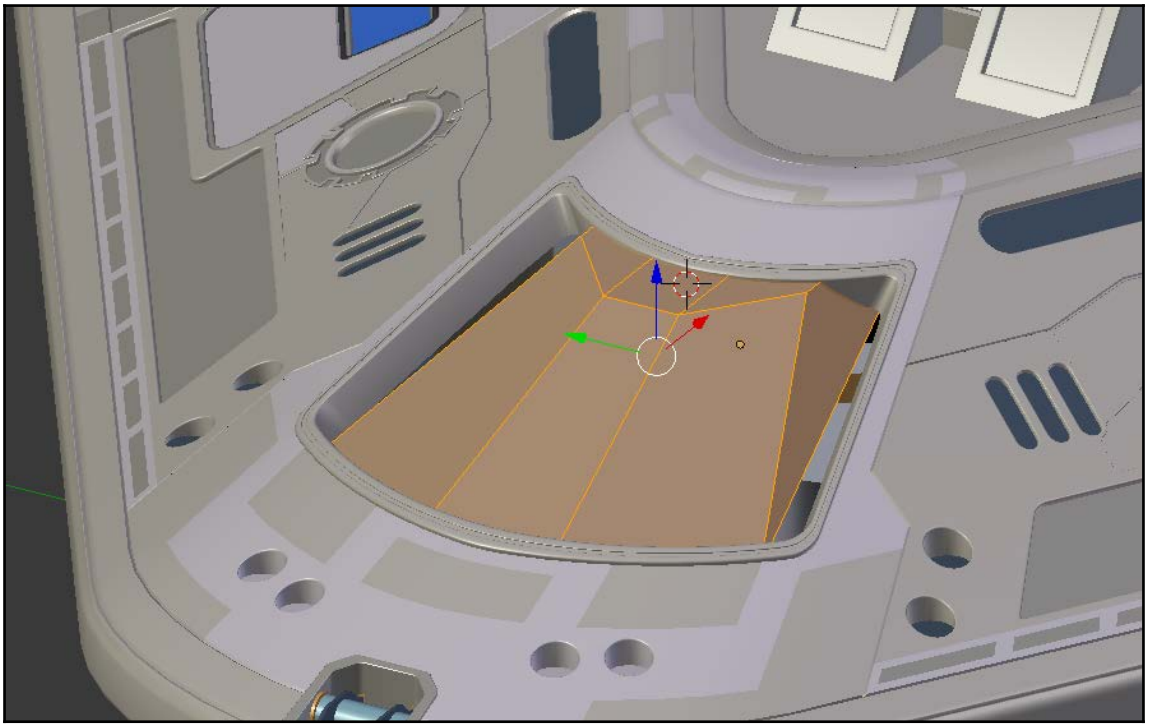
With the **Inset** and **Extrude** tools, you can now create a nice border for your window area:



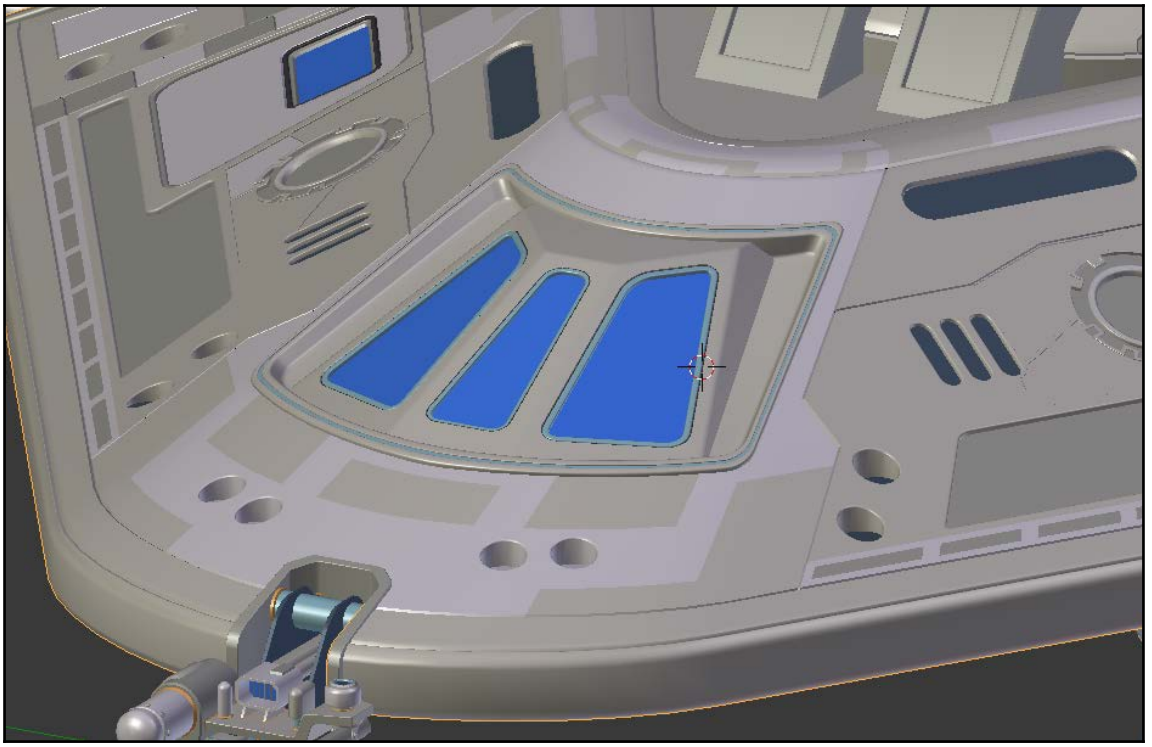
And we'll just bevel this so that it looks a little nicer:



Then, I'll delete the back of the window faces and add a new object to serve as my cockpit:

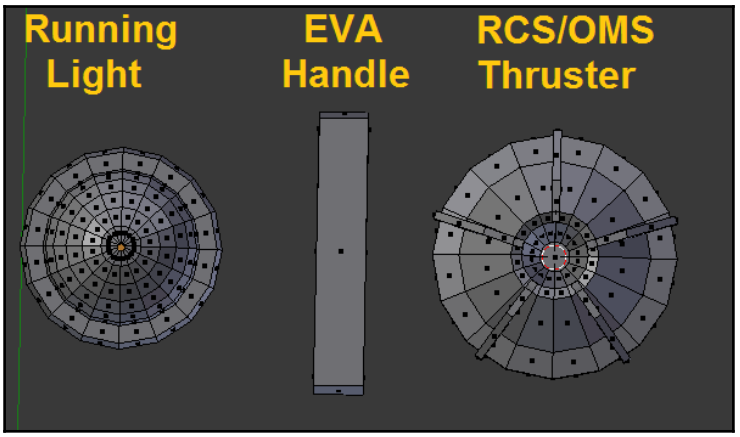


With a little beveling and extrusion, we can make it look unique:

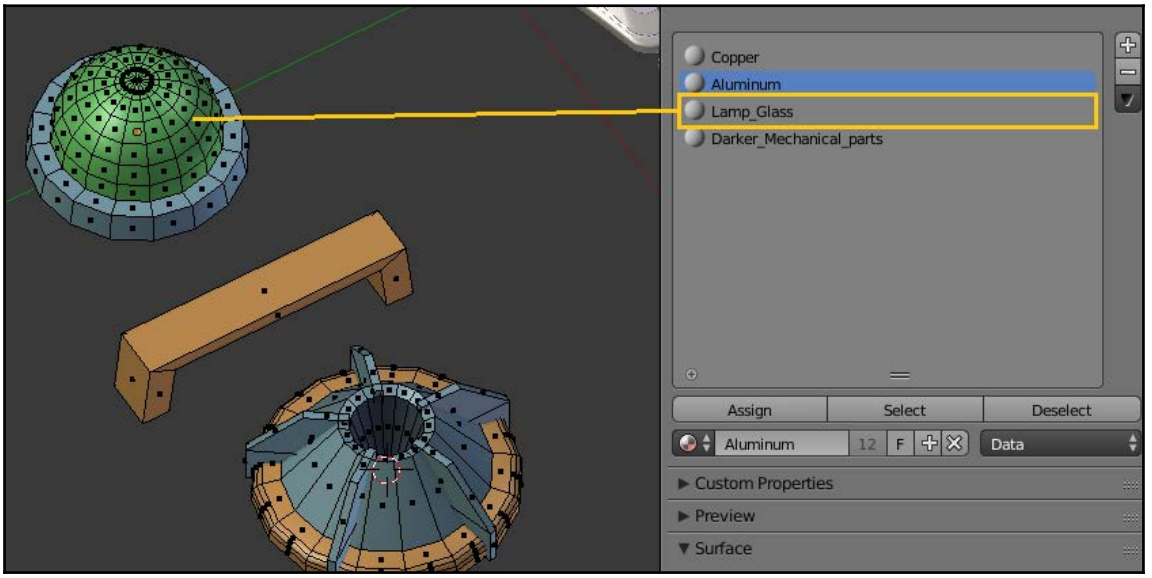


Creating small details

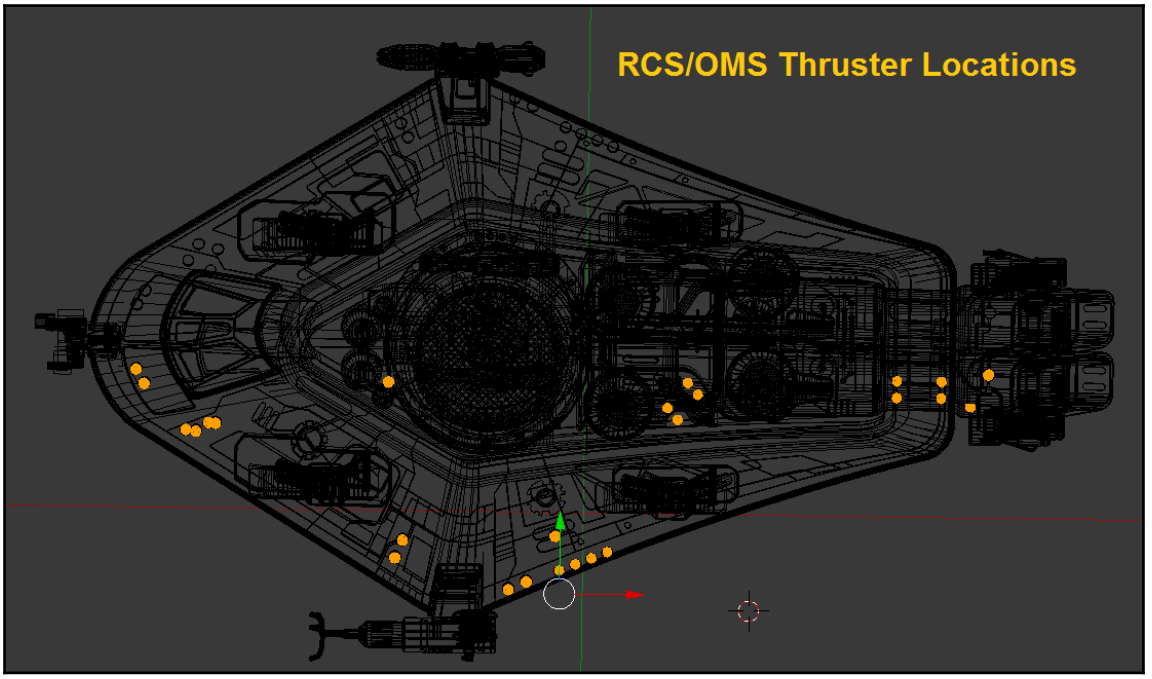
Next, let's make some small parts. These things will be duplicated and placed all over the ship, so it's easier to just model one of each and then move them into position. The more copies of each one that you plan to make, the lower you should keep the polygon count:



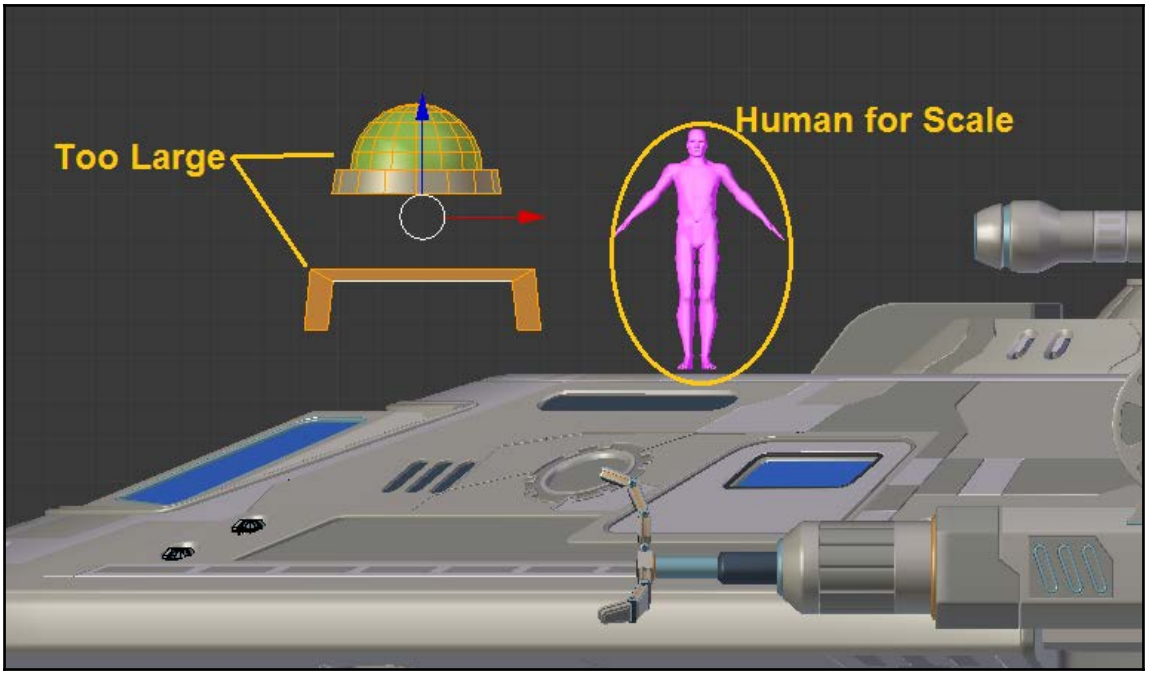
Before duplicating, of course, we'll want to add our materials:



Here's where I ended up putting the RCS/OMS thrusters (on 1/2 the ship):

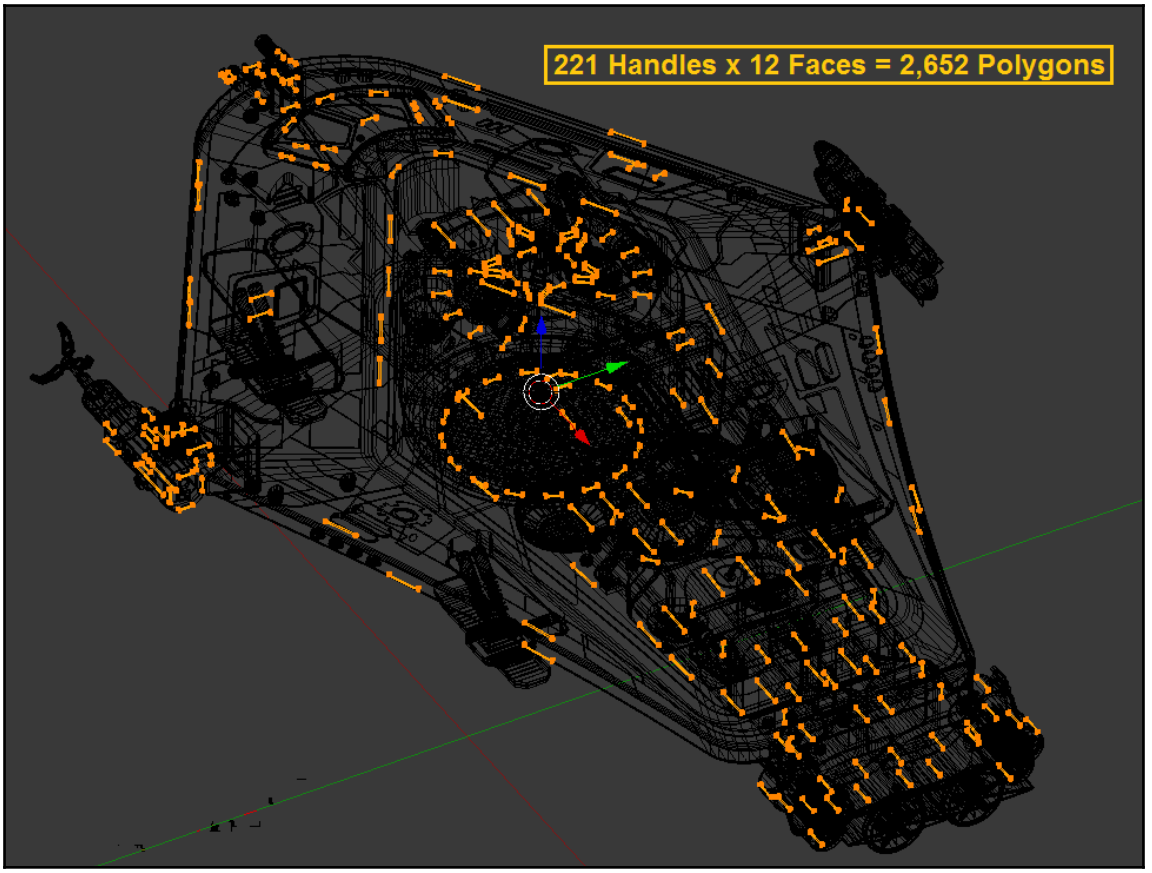


The thrusters can be any size (scale) that they need to be, but the handles and lights are a little different. These should be related to the size of a human. To make sure we get this right before making a lot of copies, we can just add a quick human model to our scene (or at the very least, just a cube that's as tall as a person would be):



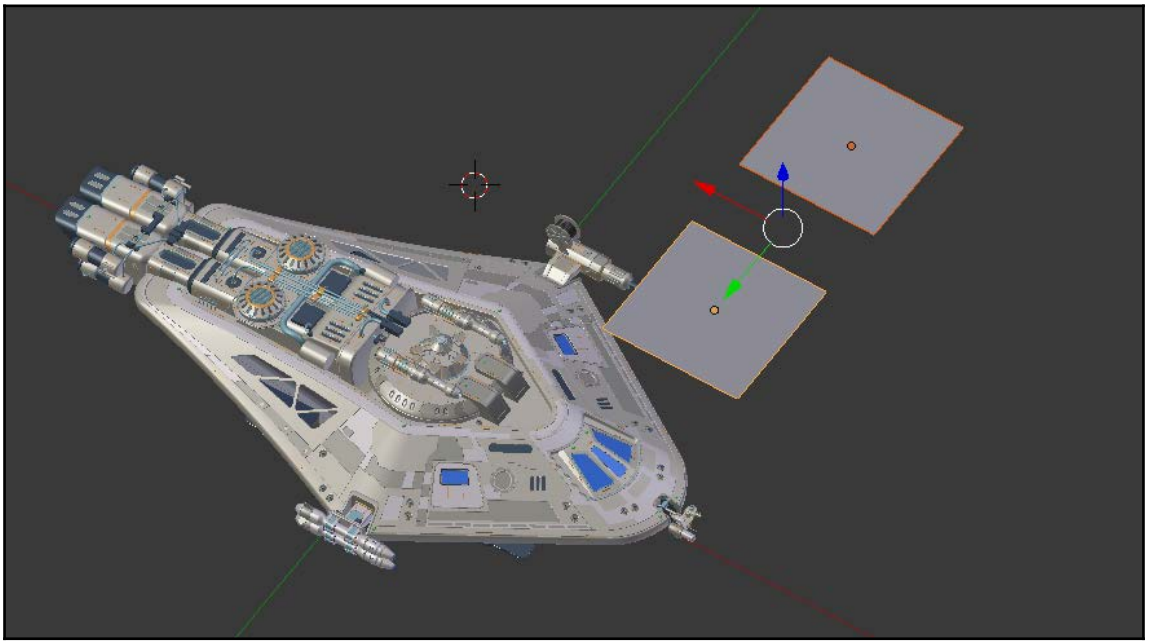
This will give you a good sense of scale and allow you to get your little details right. For instance, it wouldn't make sense if the EVA handles were larger than a human. Nor would it make sense if the running lights were larger than a bathtub. So, we can scale these down to the correct size.

I knew I was going to make a lot of EVA handles, which is why I kept them so small:

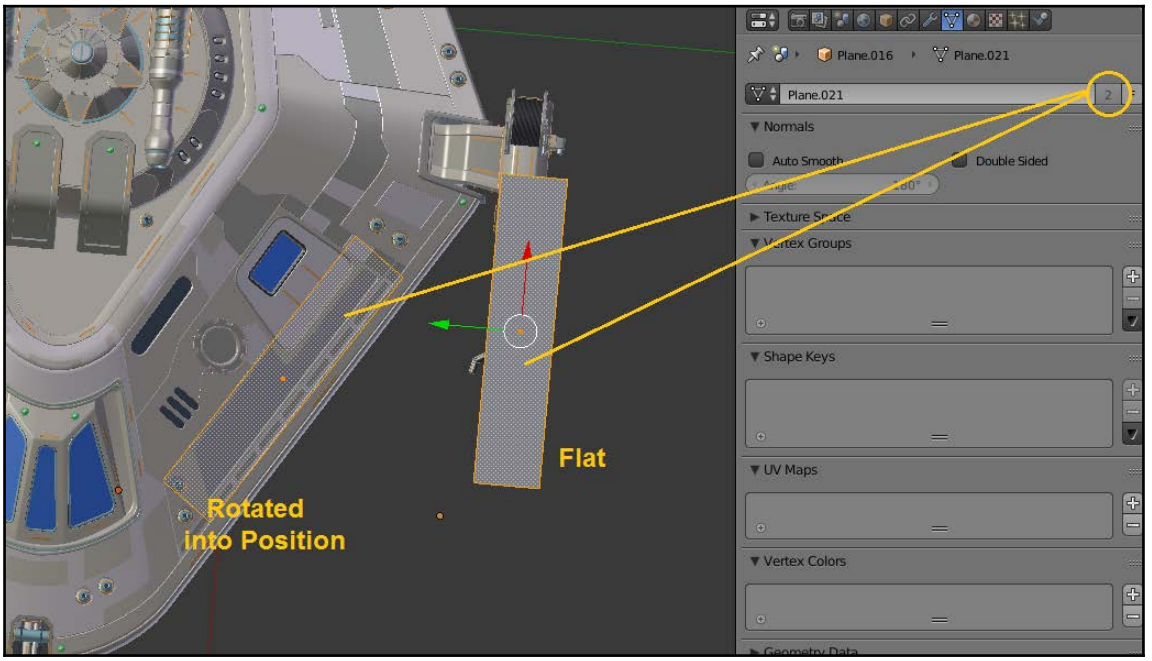


Another thing I wanted to do was add some very small mechanical parts to the front of the ship. It's easier to model these things when they're *flat* (aligned with the *X*/*Y*/*Z* axes), so we'll use duplicate meshes again.

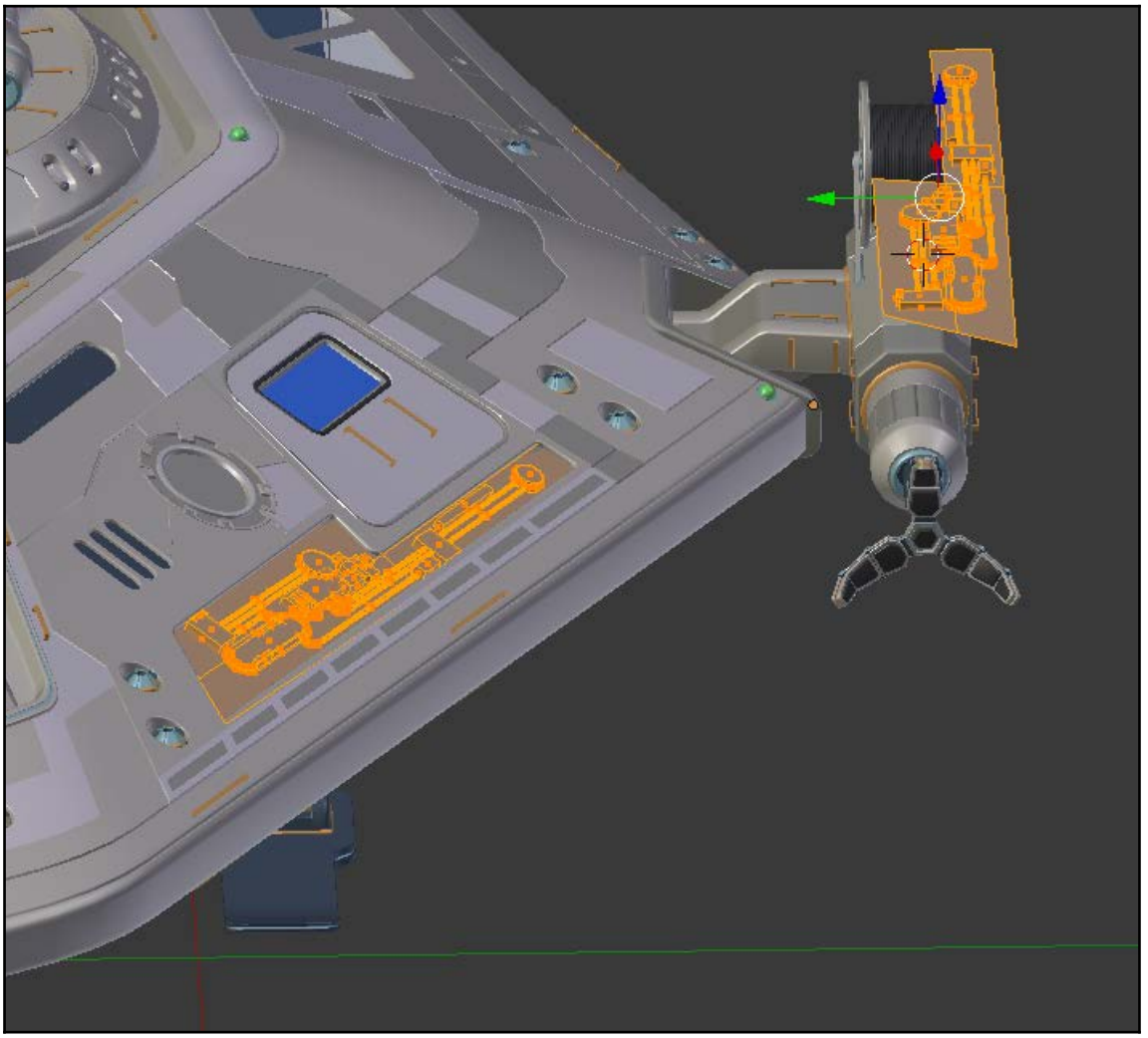
First, add two **Planes**:



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:

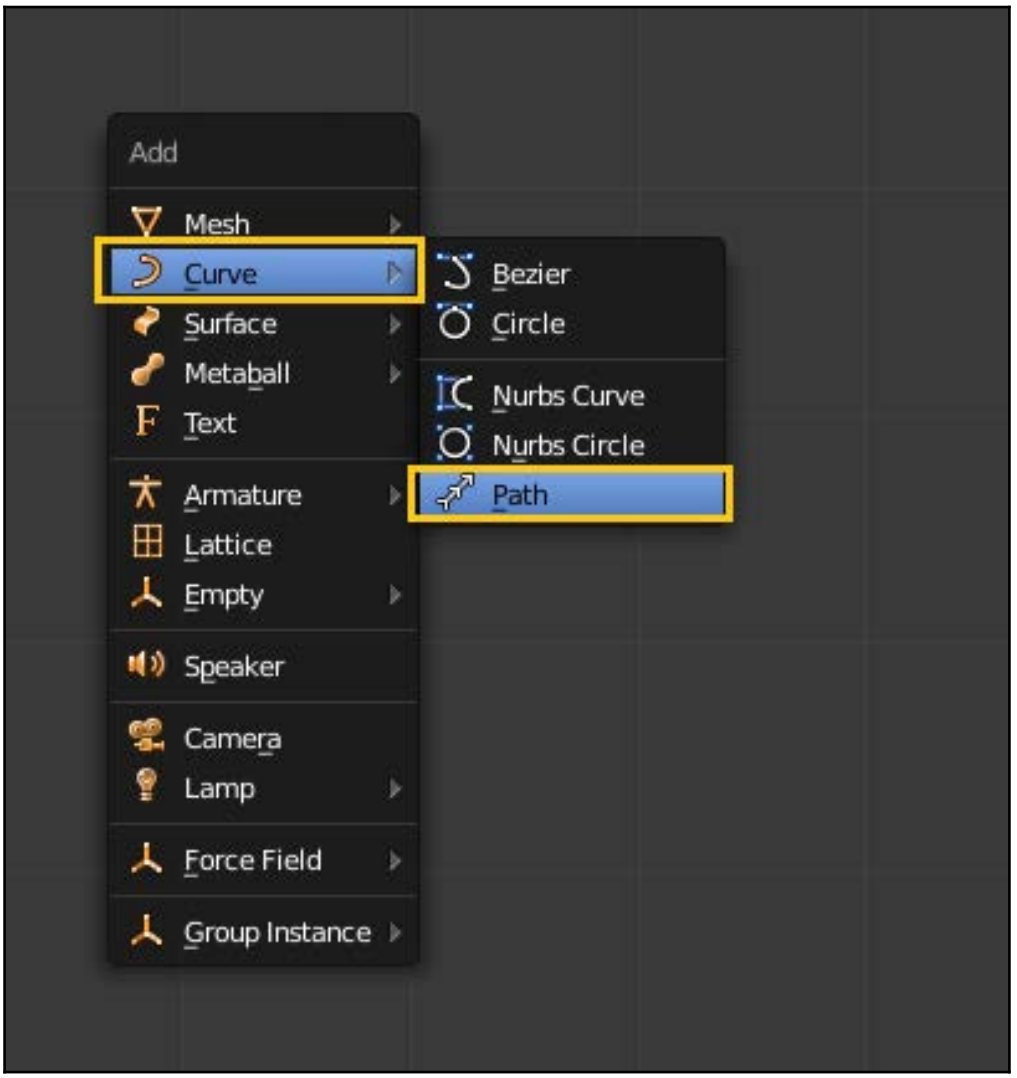


Now, you can model the other one, which is quite a bit easier, and the detail will show up on both:

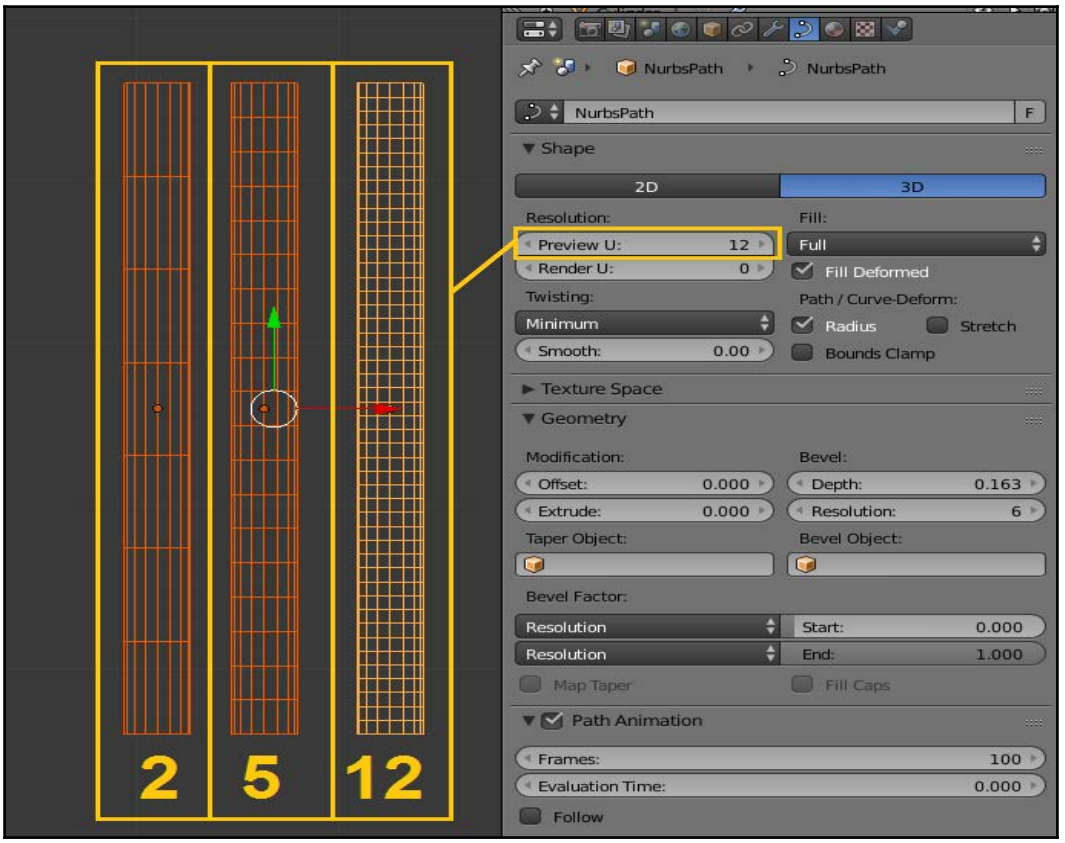


Working with Path objects

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:



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:

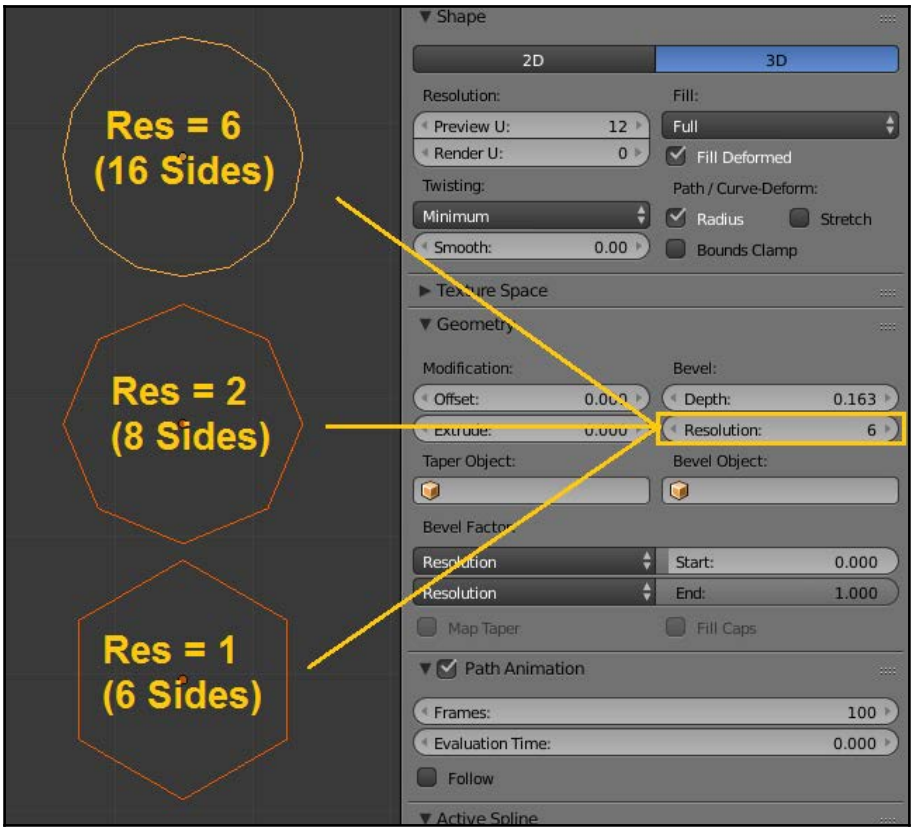


This is very important. If your **Path** is going to have a lot of curves to it, you'll need a decent resolution to make it look smooth. On the other hand, it should be obvious that the higher the resolution, the more polygons you'll eventually be using.

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.

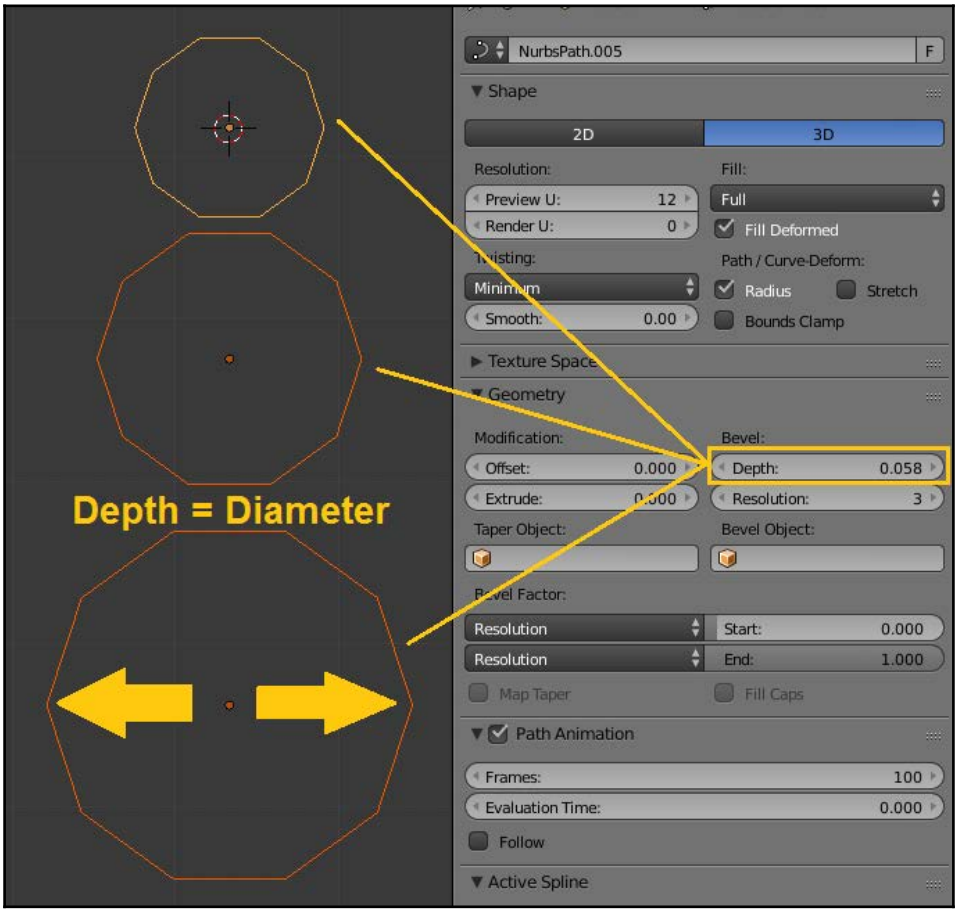
Eventually, we'll want to turn our **Paths** into real mesh objects. It means that the **Preview Resolution** will be reflective of our final model.

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:

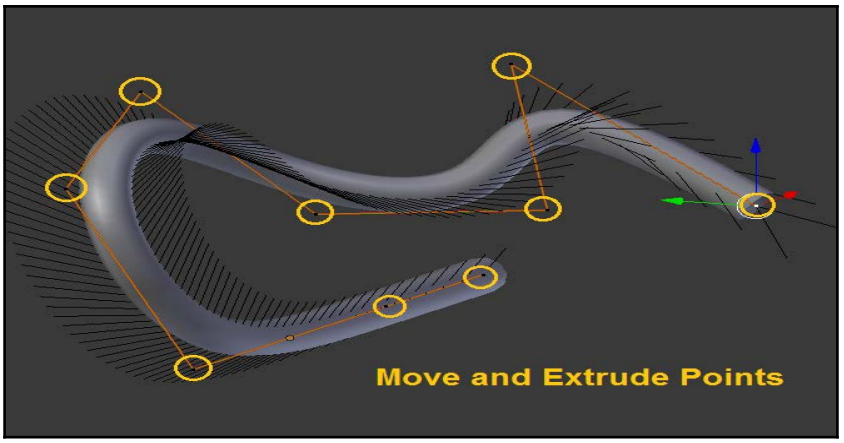


How many sides should your **Path** have? Well, it's really a question of how many you can afford, in terms of polygon count. If you're going to put in one or two **Path** objects, you can probably turn the resolution way up. But if you're going to be making a whole lot of them (like I am), you probably 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.

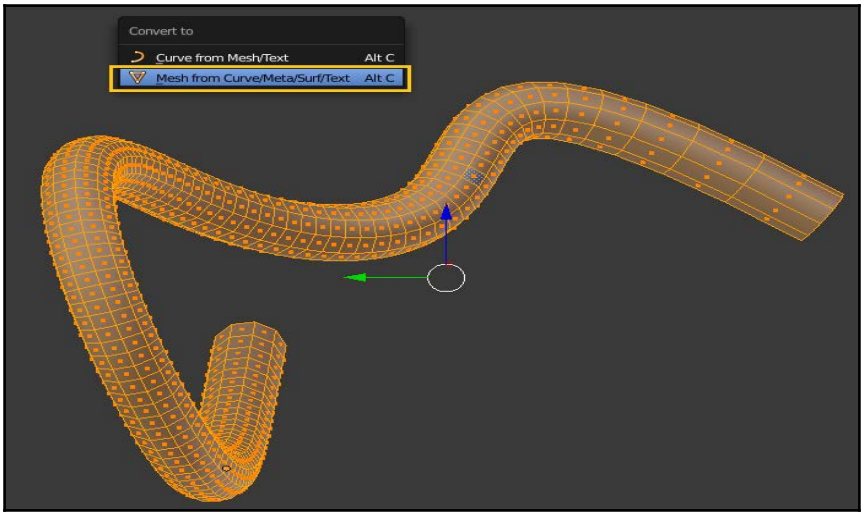
The next setting we'll want to take a look at is **Depth**. This is the diameter of your **Path**:



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.

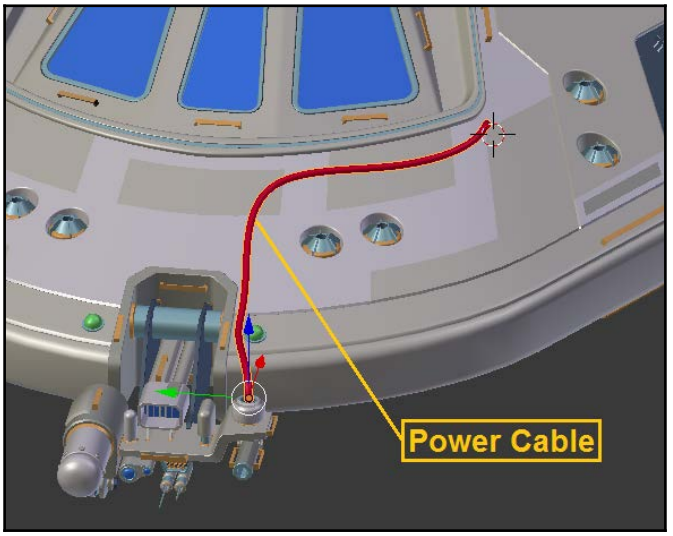


Maybe you're wondering why we need to convert at all. Can't we just leave these as **Paths**?

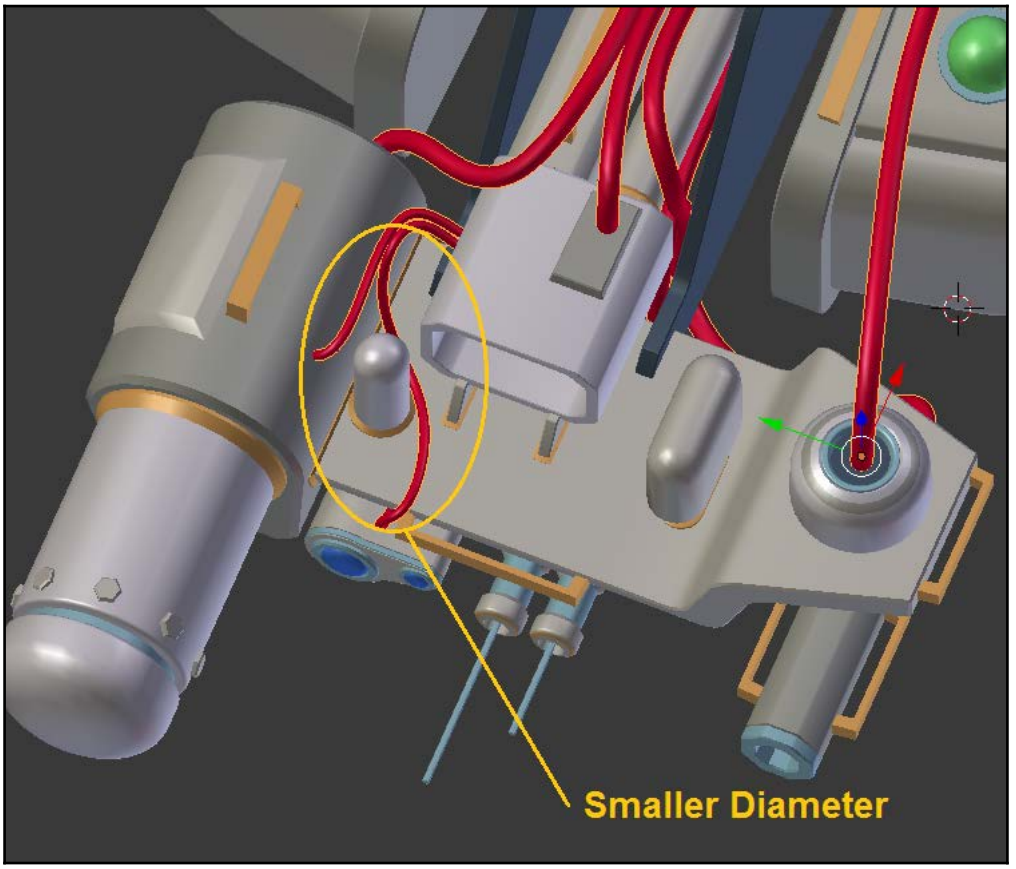
Sure, you can. There are a couple of downsides, though. For example, you can't do any UV unwrapping on a **Path** (we haven't covered this yet, but it's how you assign an image texture).

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.

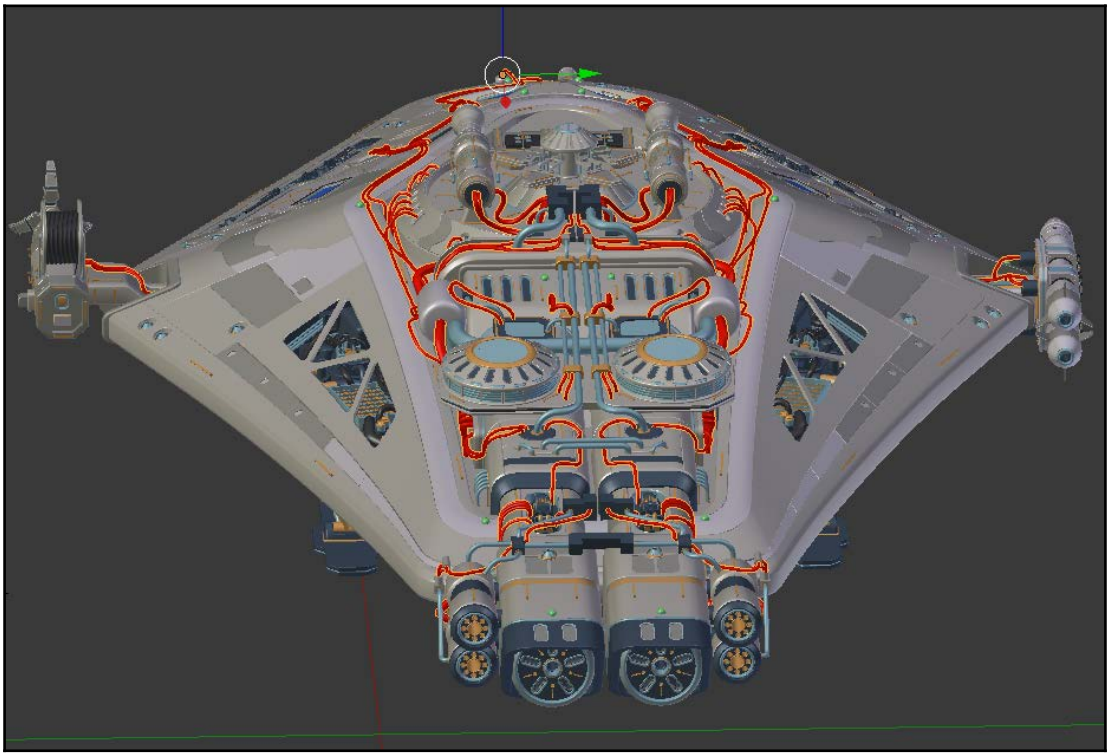
Here's an example of a **Path** object that I've added to the front of the ship to make a power cable:



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**:



Here's what I ended up doing with my **Path** objects:

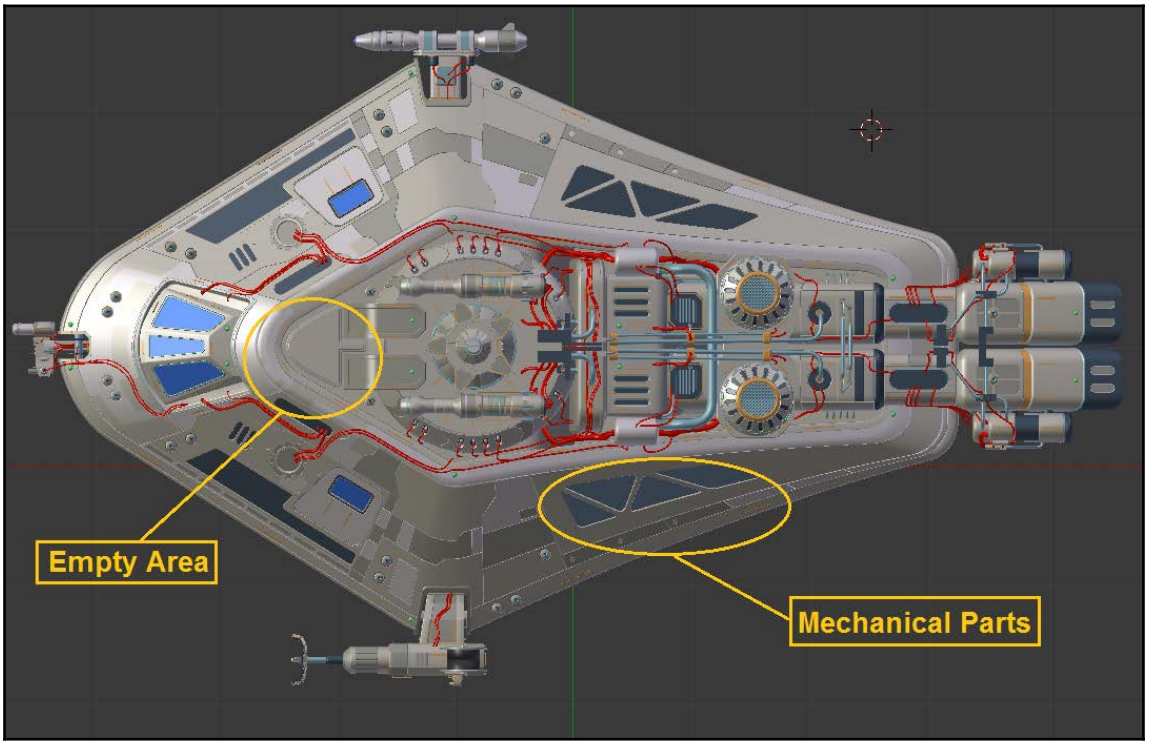


You may think that this is too many **Paths**. After all, it does appear a bit cluttered and messy. For obvious reasons, engineers tend not to design things with random cables and wires all over the place. If they're exposed, they tend to be tied down in fixed cable guides or rails.

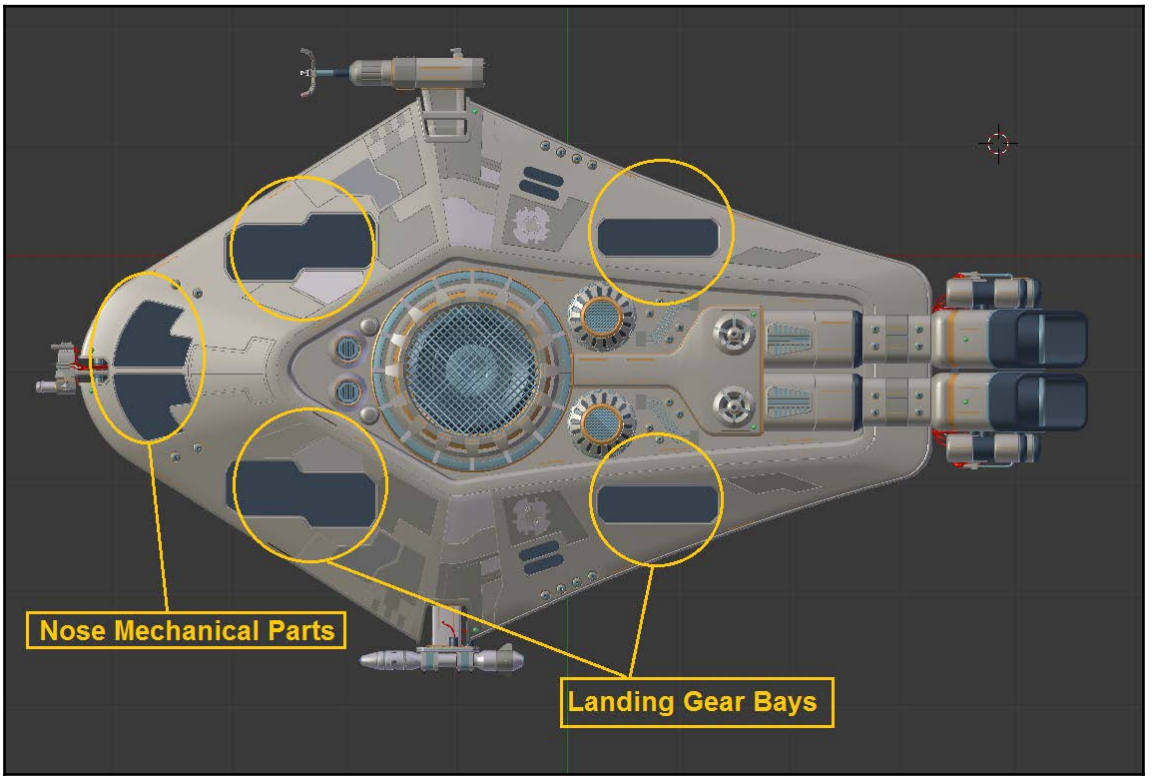
On the other hand, maybe you want your ship to be a little messy. Maybe it's not a brand new vessel straight from the factory, but something that's been modified and haphazardly repaired over the years. Maybe a lot of that equipment wasn't in the original design, so it's been wired up over time in an unsafe manner. This is really an artistic choice, so I leave it to you.

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:



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.

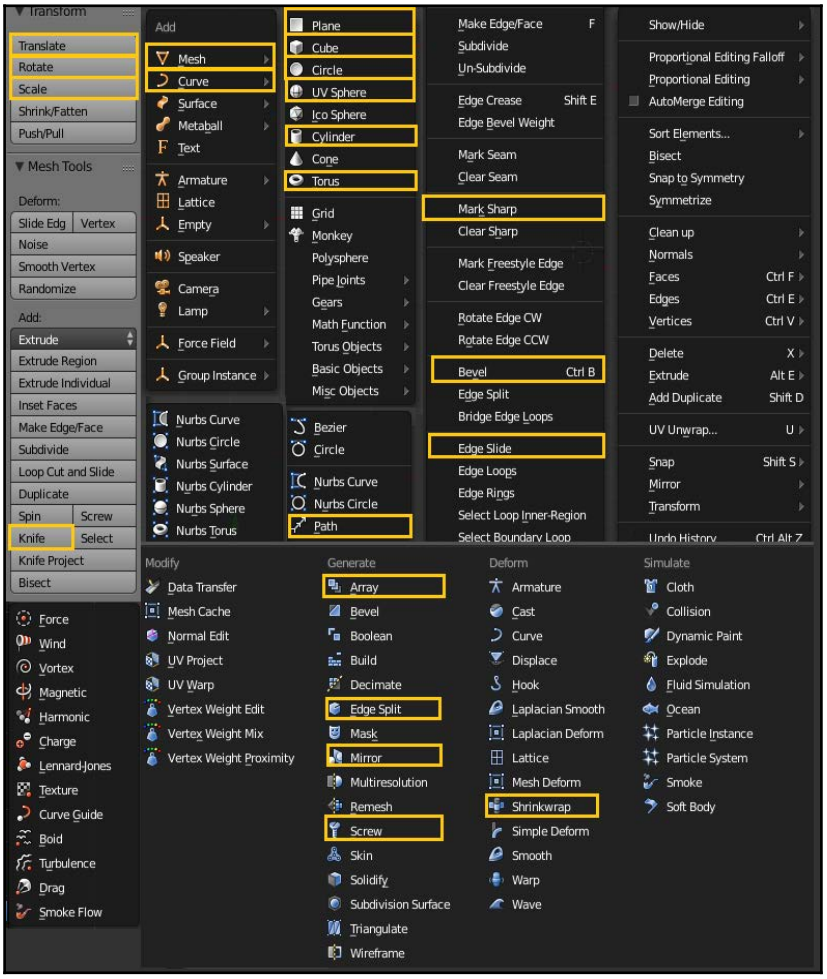
Here's what my final ship looks like (with no materials):



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.

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.

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!

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:

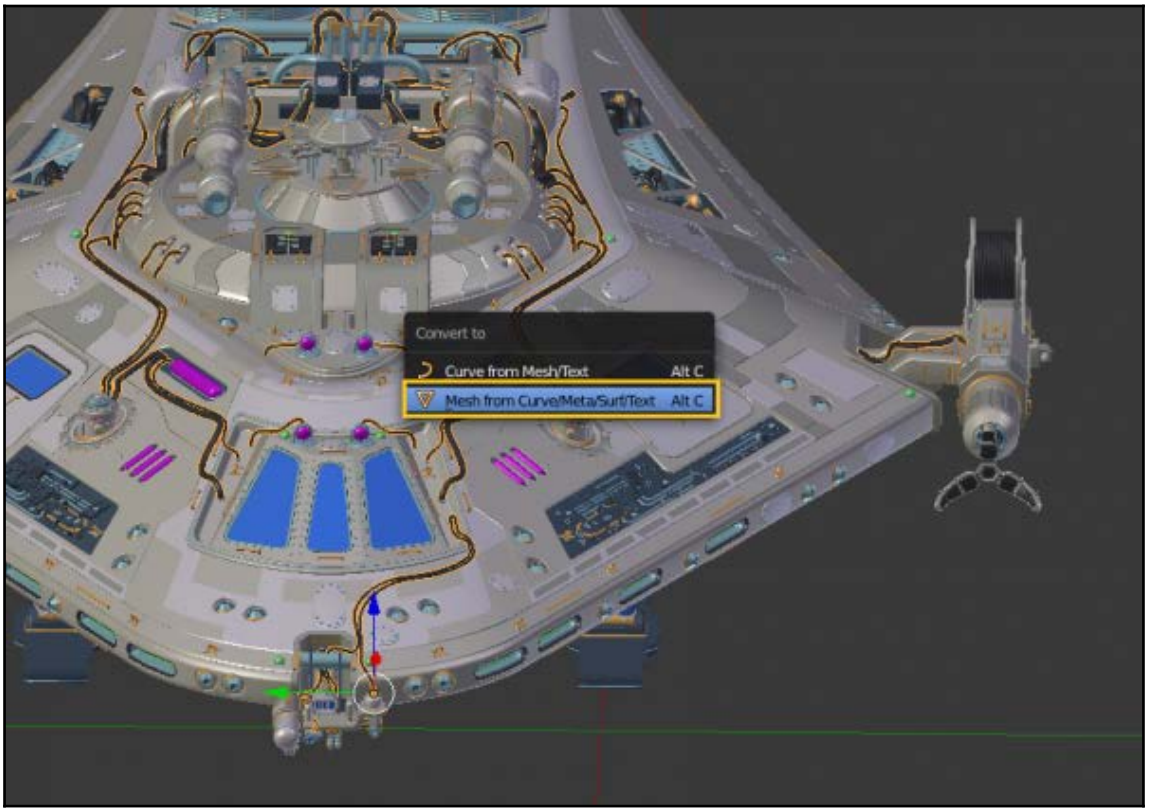
- Preparing the model and scene
- Creating materials
- Adding decals with UV maps
- Other possibilities

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!

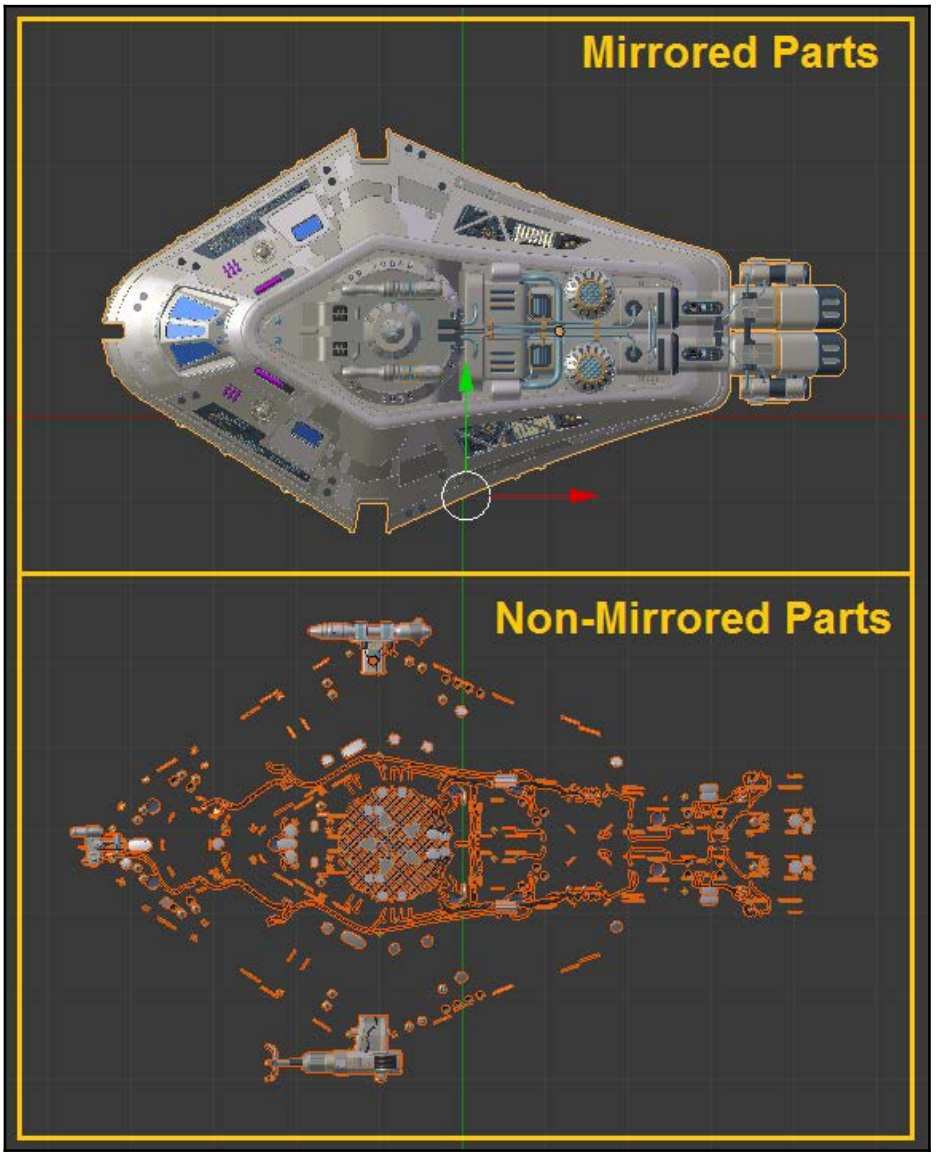
Preparing the model and scene

Before adding materials, let's prepare our model. First, make sure that path objects have been converted to meshes with *Alt* + *C*.

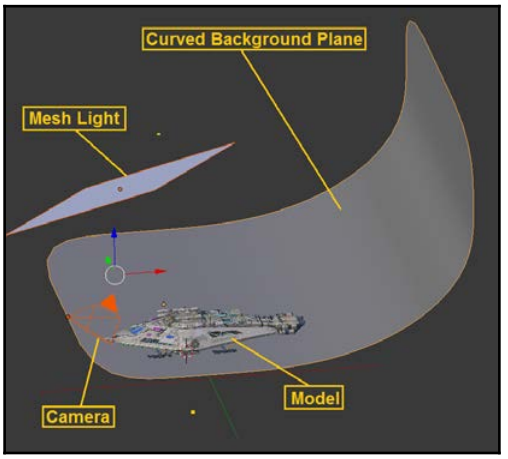


Then, we'll combine our various parts together. In the end, I'm just going to be left with two objects: mirrored parts and non-mirrored parts. This will make the texturing a bit easier.

In general, it's good practice to have the minimum number of separate objects as necessary.

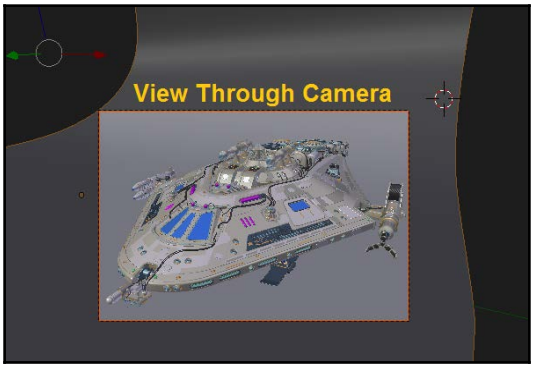


Now, we'll add a basic scene to render our spacecraft. I've added a curved background here, along with a mesh lamp and a camera:



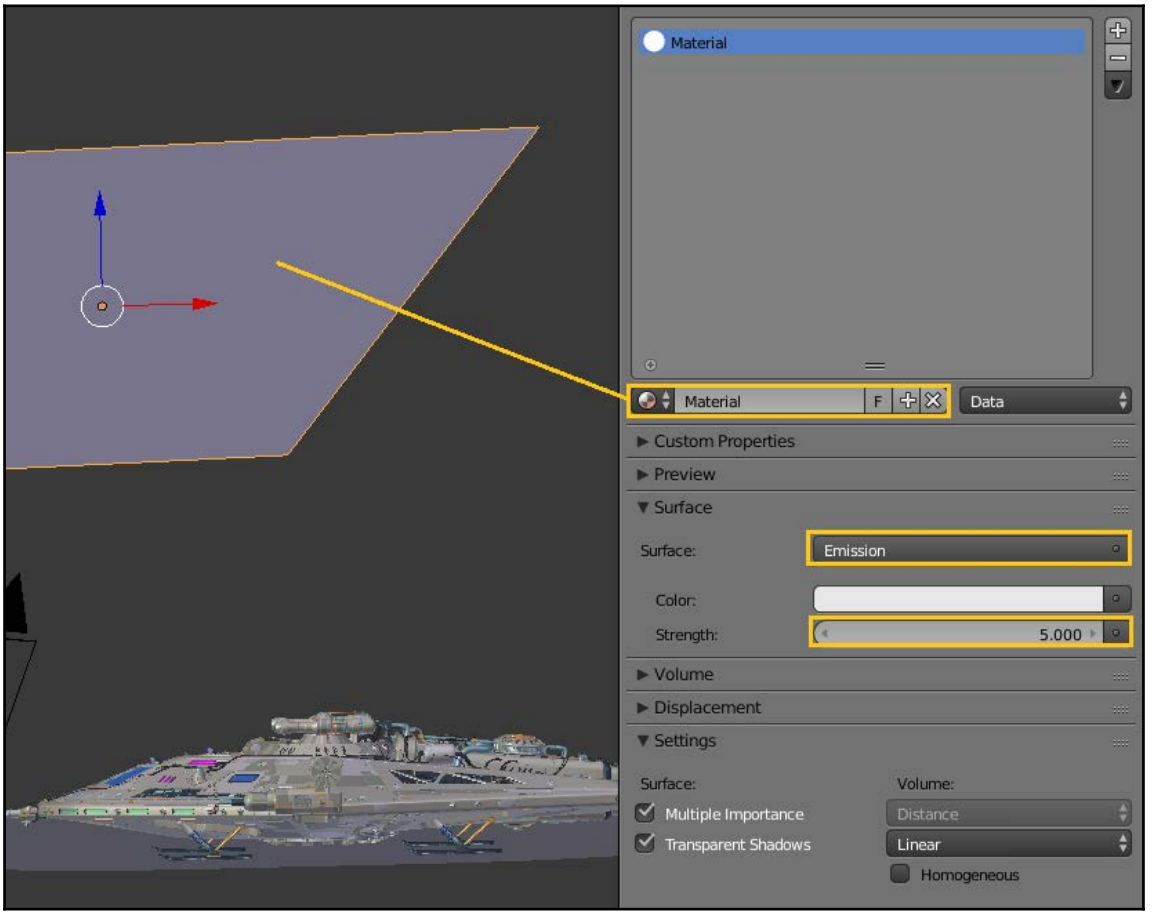
Obviously, this is a very simple scene. You could get far more complex with it, but this will work to show off the model.

By pressing 0 on the number pad, you can now look through your camera.

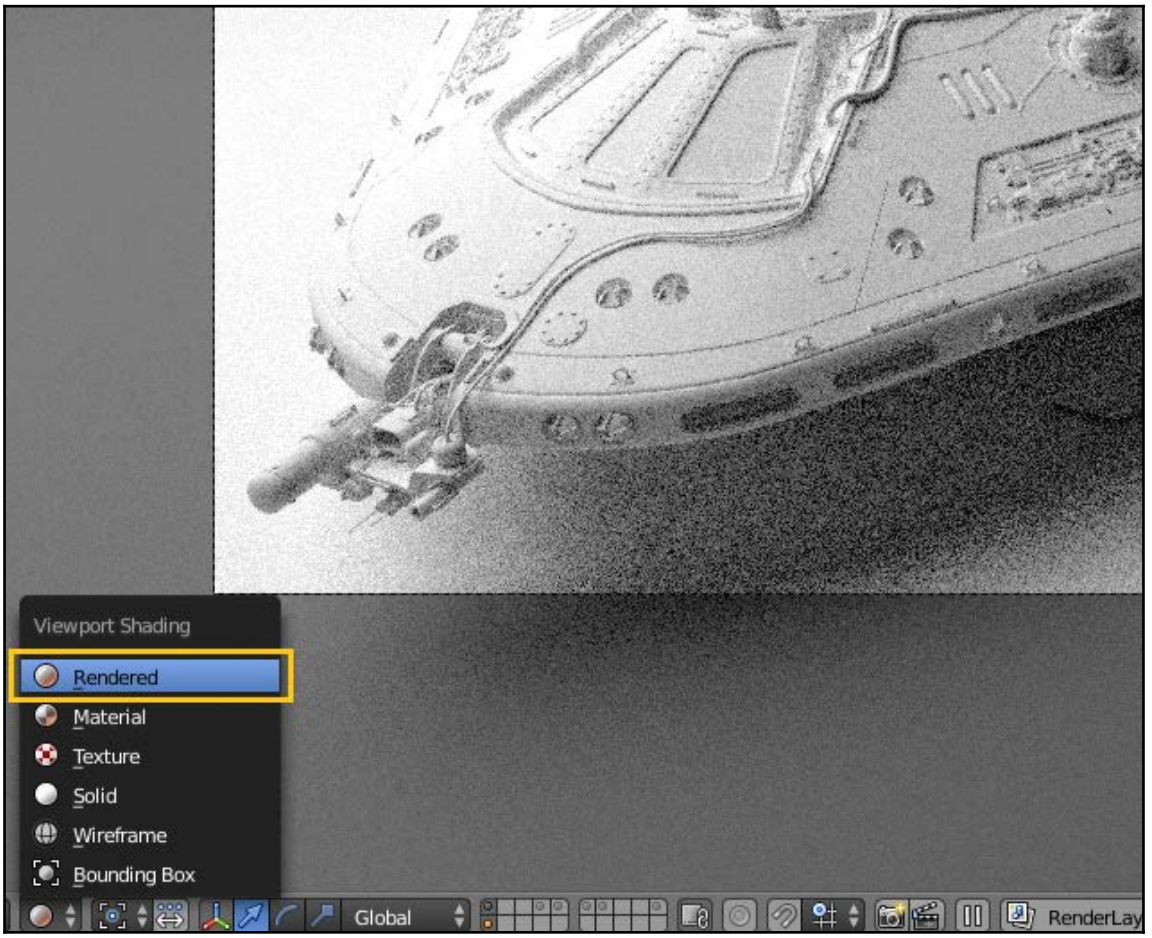


Make sure that your background is large enough and that you see exactly what you want. You may need to adjust the position of various objects.

Now, we'll add an **Emission** material for our light:

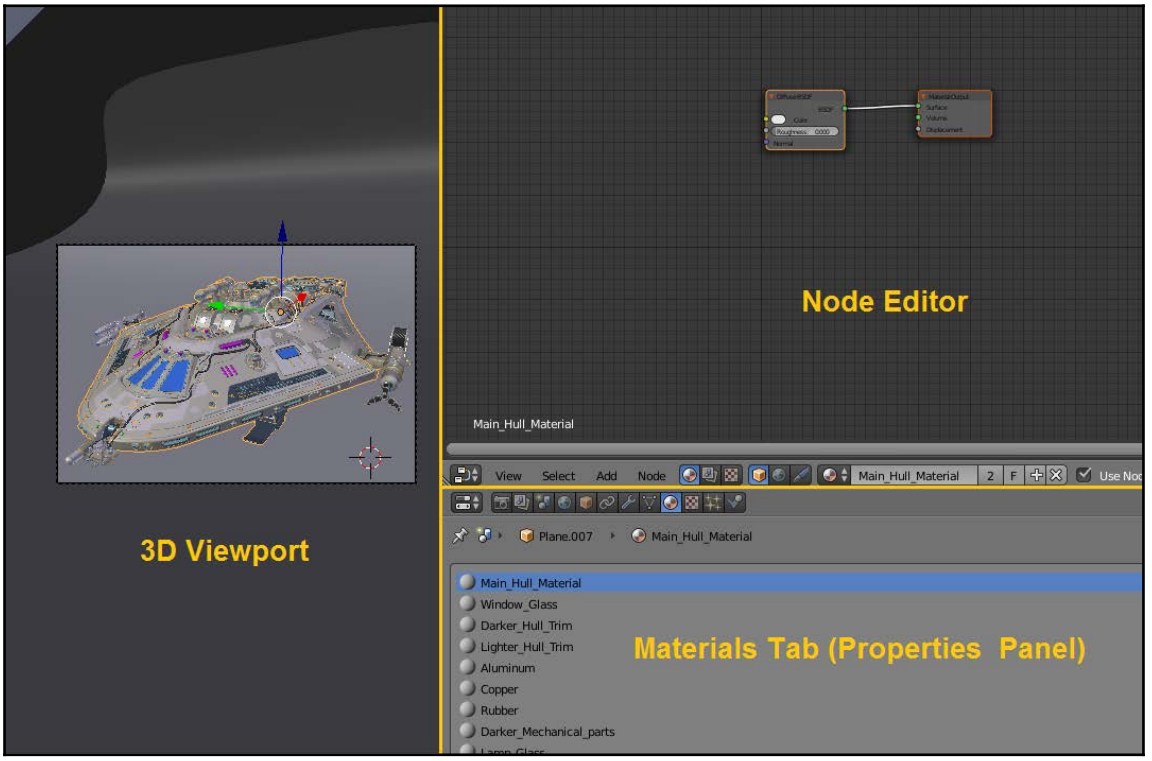


When we switch to the rendered view, we can see that there's now illumination in our scene:



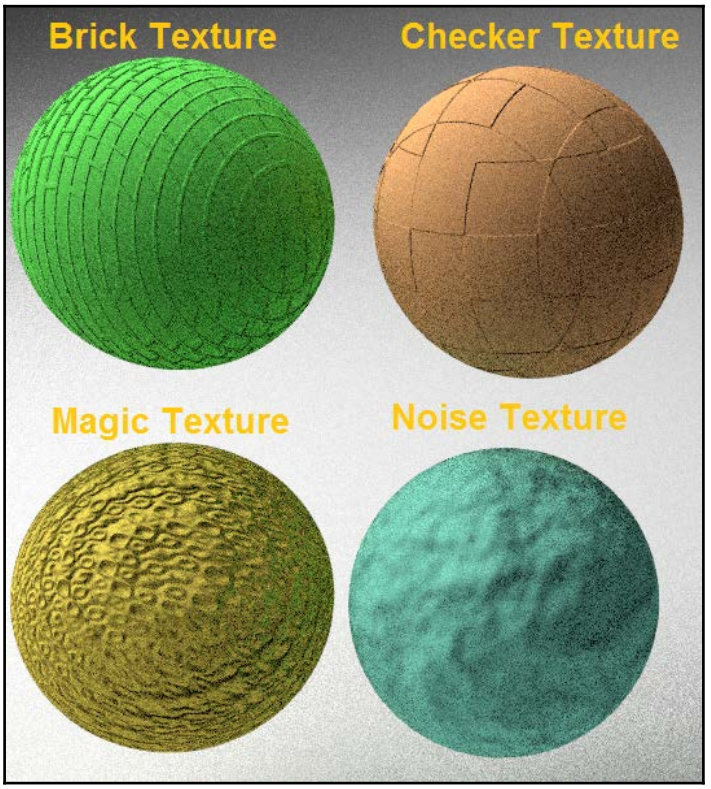
Creating materials

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.

Before going further, let's take a look at some of the basic procedural textures that come with Blender:

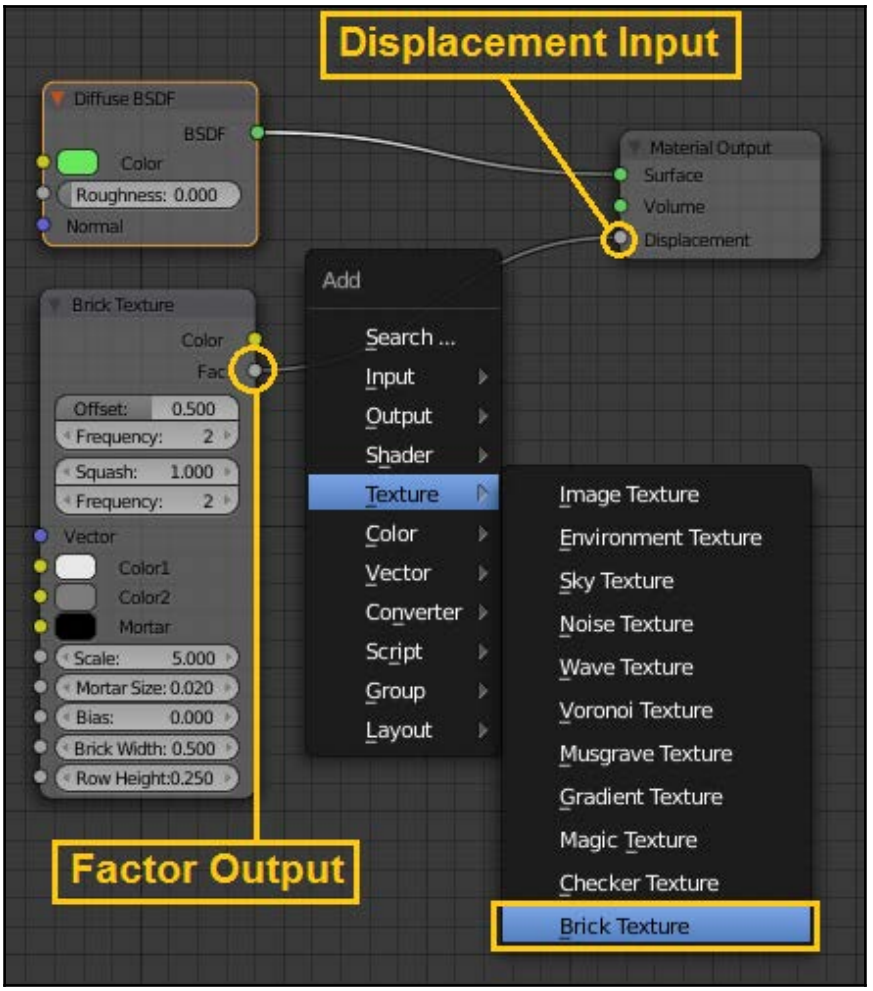


As you can see, I've used these textures to affect the displacement or normal of the material. The textures are basically acting as **normal maps**.



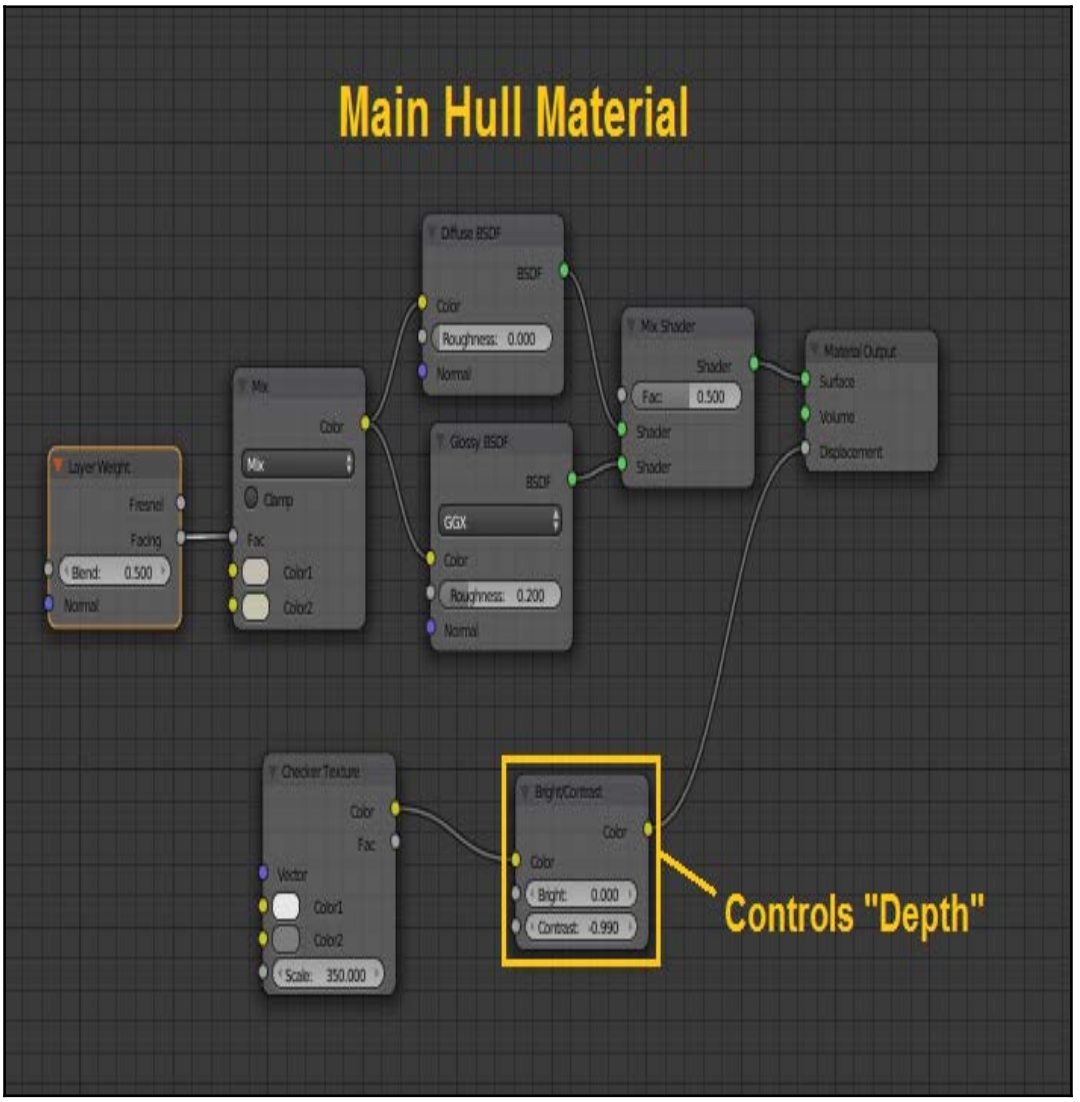
A normal map is a way to add additional detail to the model without physically building it. For example, if you wanted a cloth pattern, it would not be practical to model a cloth with all of the tiny folds and stitching. Normal mapping allows us to simulate the light and shadows of a complex surface without adding all of the geometry to model it.

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.

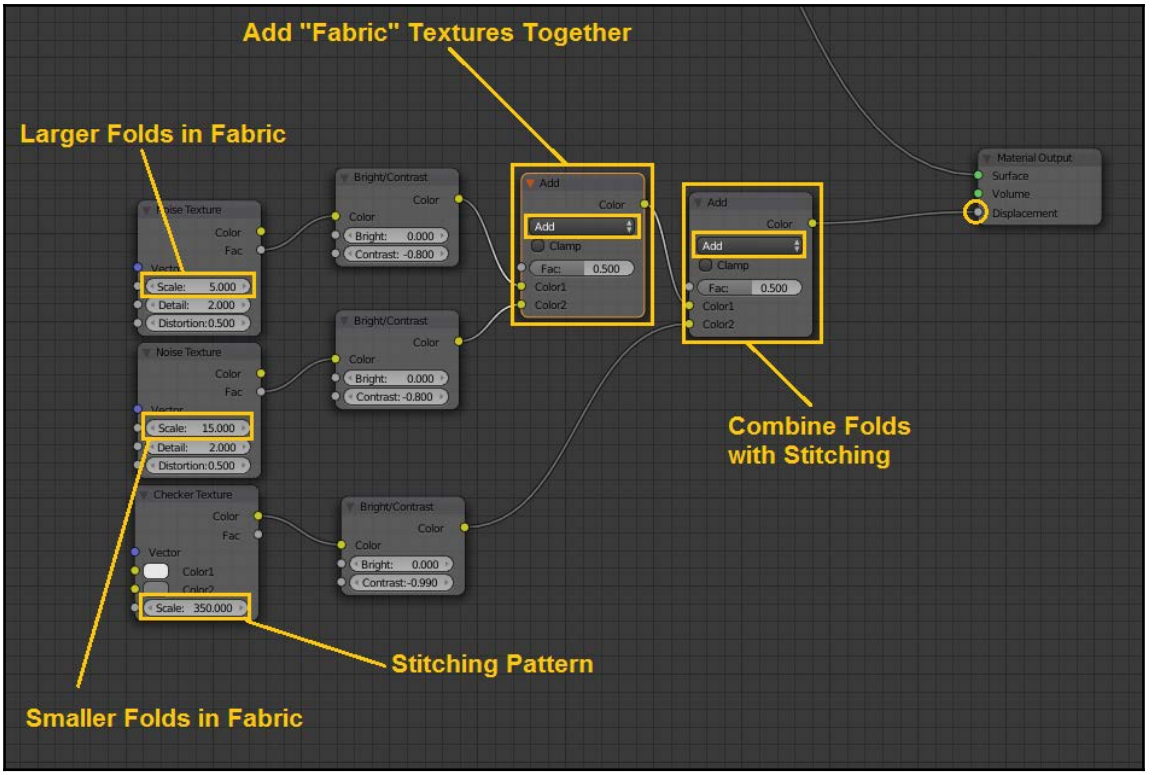
I'm also going to use a **Checker texture** to create the appearance of small stitching in the fabric. The **Bright/Contrast** node allows us to control the apparent depth of the fabric.



You can also combine a number of different textures for a more complex normal map. Here, I've combined a couple of **Noise texture**s with the **Checker texture** to get the overall normal or displacement that I want.



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!



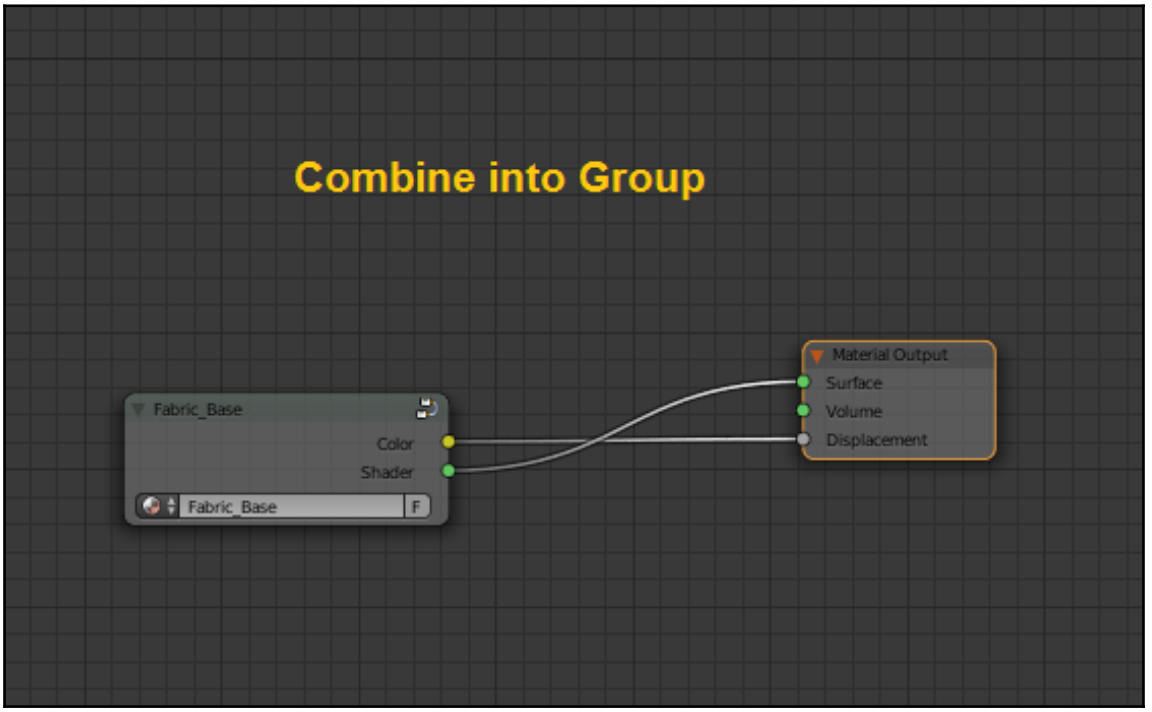
You can see what that looks like by either rendering your model or switching to the rendered view in the viewport:



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.

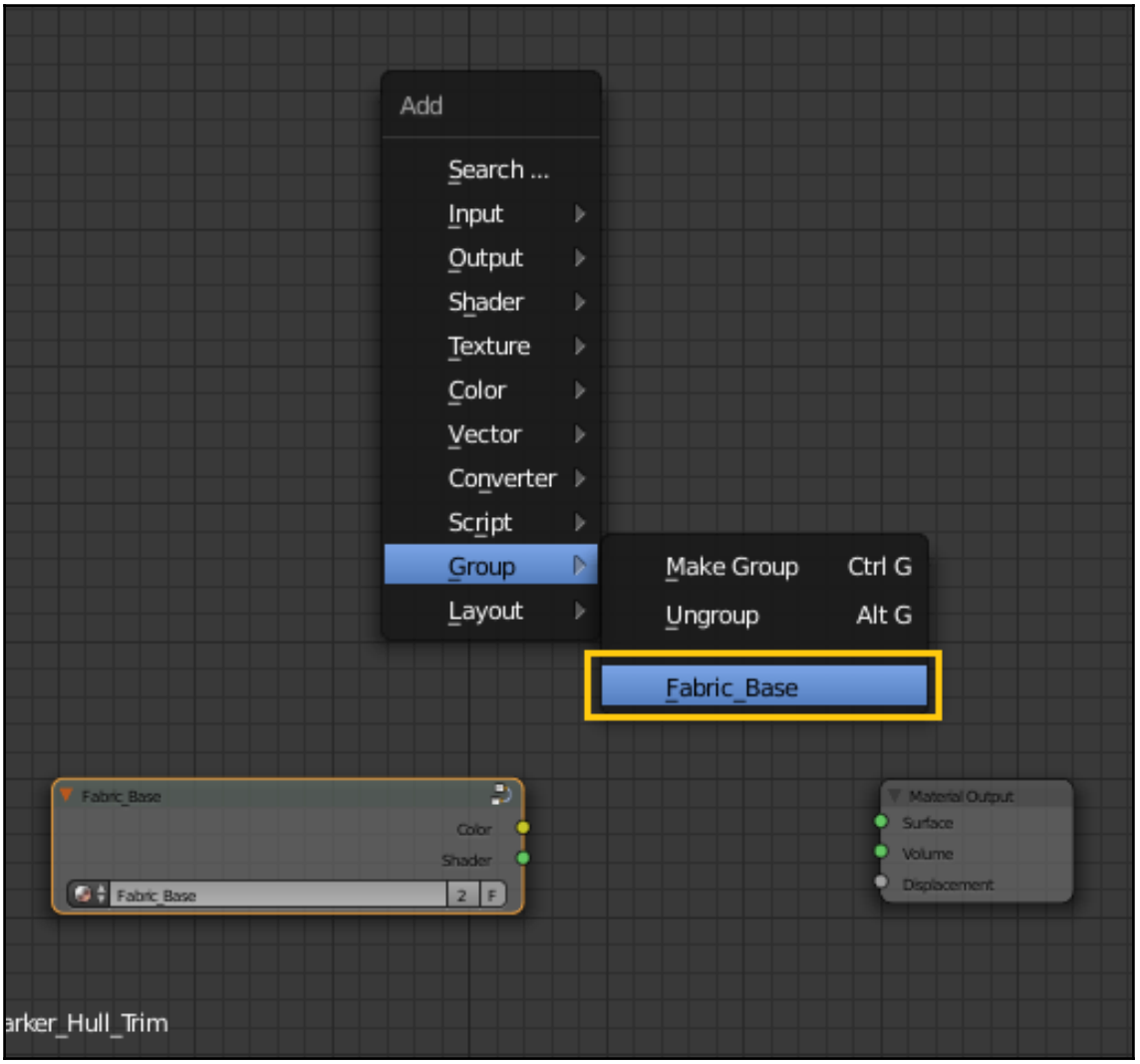
A quick way to do that is to just combine all of your nodes into a **node group**. This group can be plugged into multiple materials. It's also a good way to "copy and paste" between materials.

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.



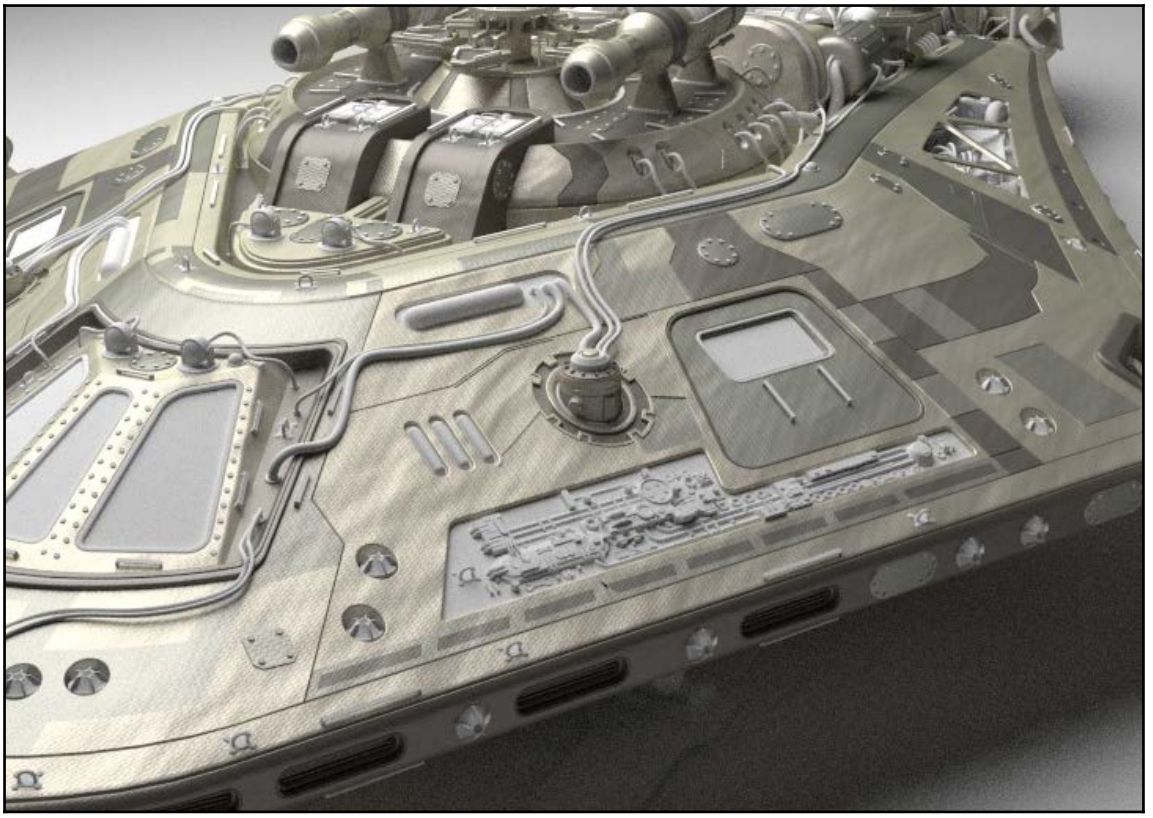
I've named this group **Fabric_Base**, but it doesn't matter what we call it. We're only using the group temporarily.

Now, I can go into another fabric material and add that node group:



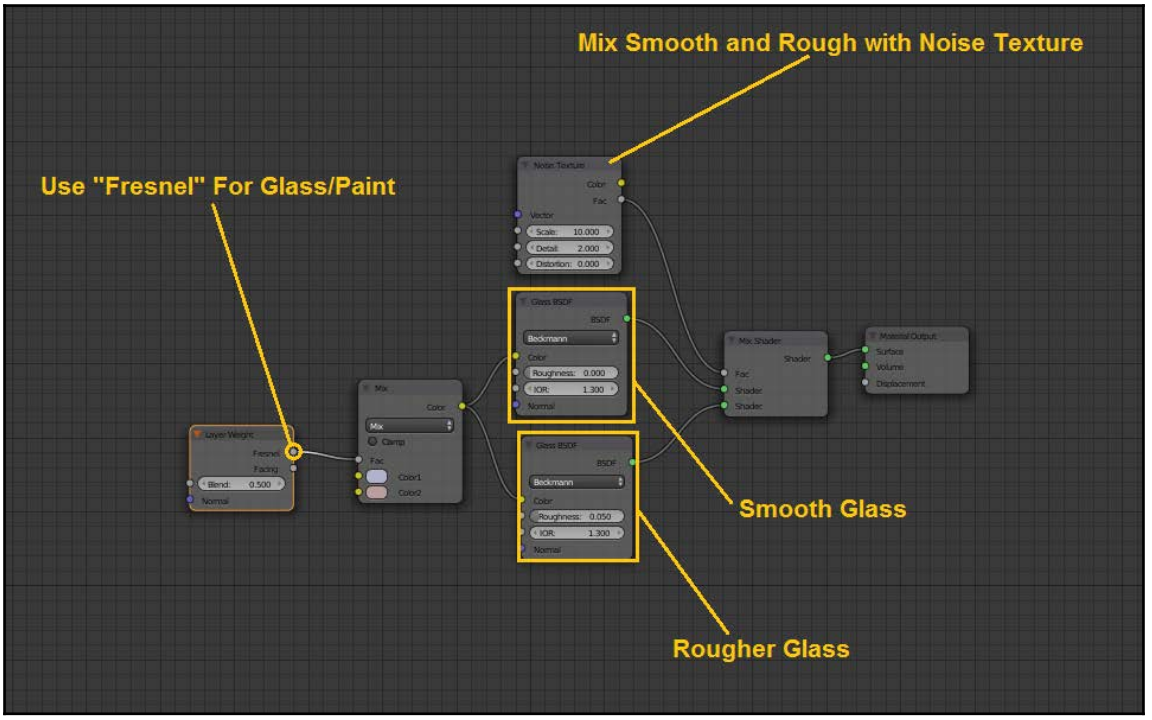
Using *Alt* + *G*, we can ungroup the nodes. We've basically copied everything from our first material. Now, you can go in and make a few minor changes, so the material looks different from the first one.

Repeat this process for our various fabric materials:

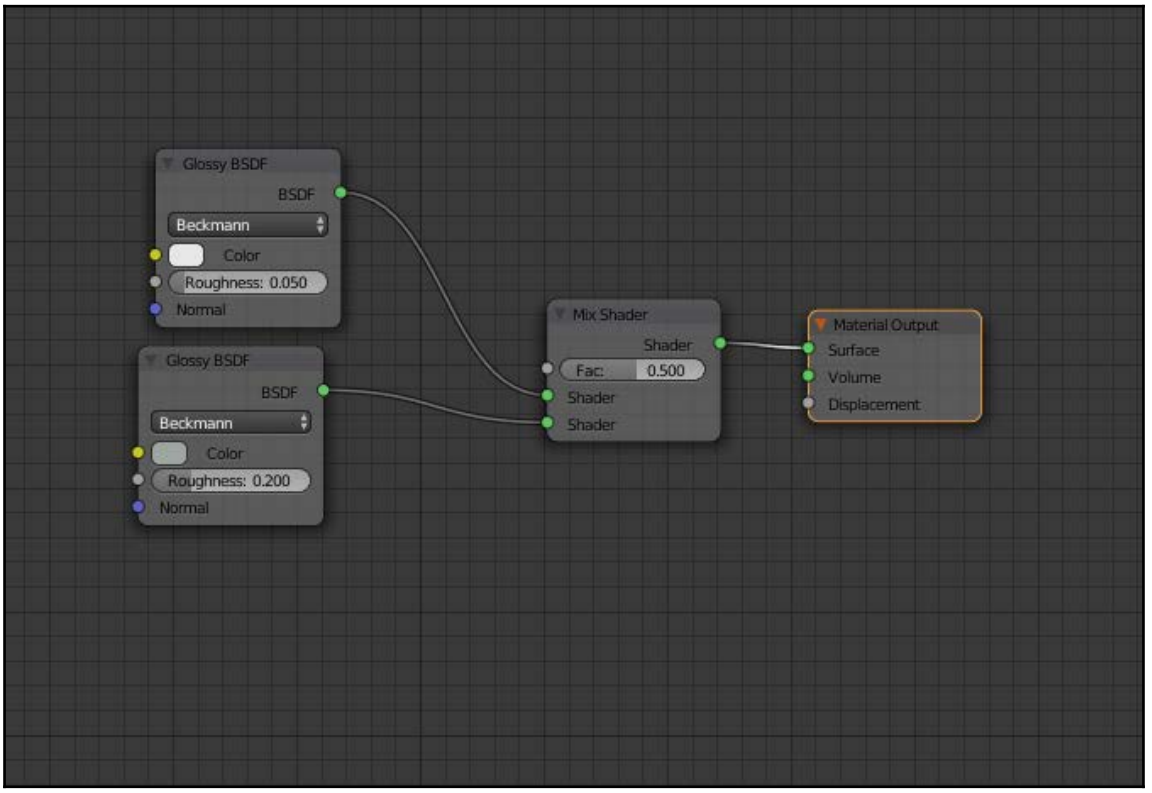


The next thing we'll tackle is the window glass. For this, I'm using a combination of two glass shaders. One is a bit rougher than the other, and I'm combining them with a texture, to give the appearance of glass that's not perfectly consistent.

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.



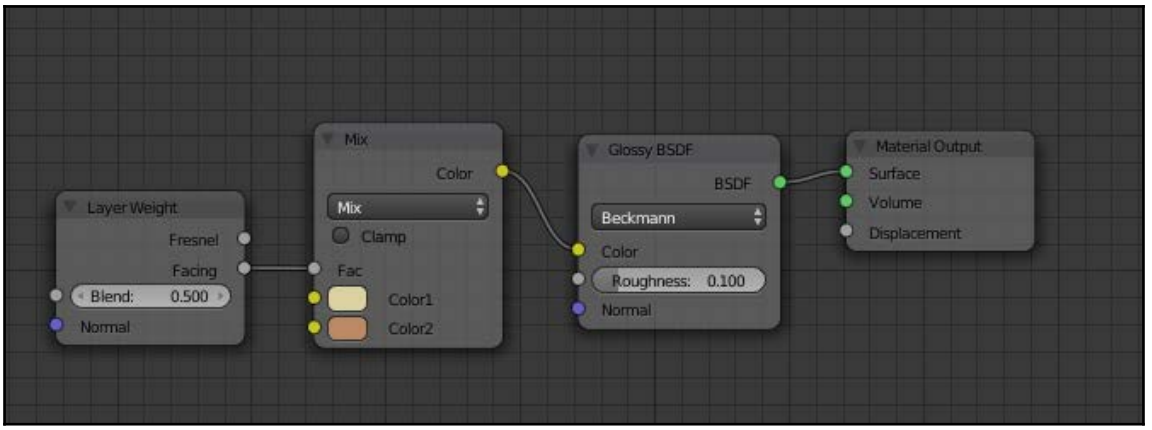
For my aluminium material, I'm going to use a simple combination of two different glossy shaders:



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.

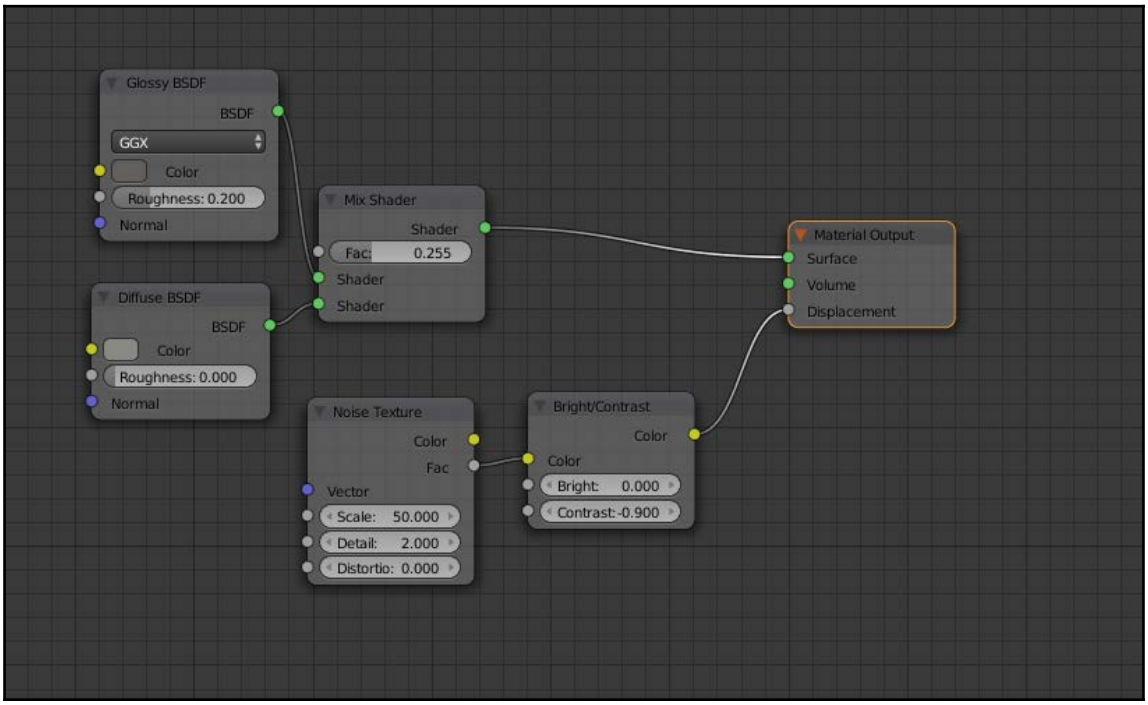
However, this is a point of diminishing returns. More complexity isn't always better. The simple setup we're using will work a large portion of the time. As always, of course, you're free to take this further.

For our copper material, we'll just combine a couple of colors into a glossy shader:

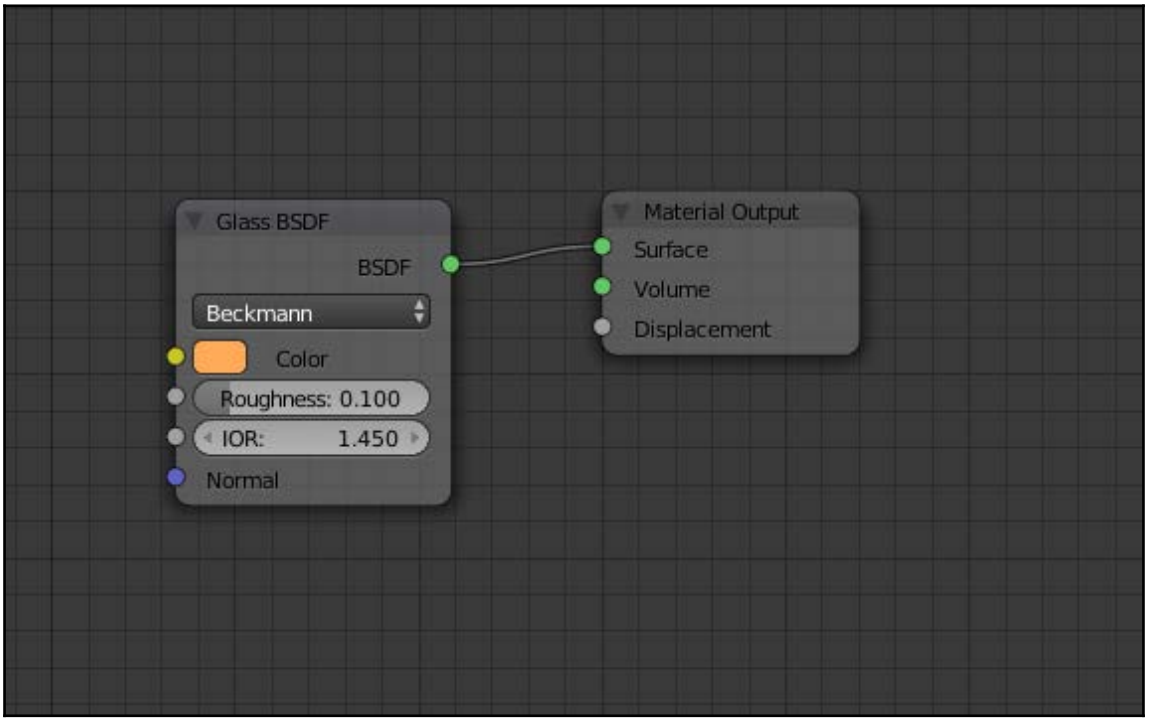


For our rubber material, we'll just combine a couple of darker colors. Note that we're using the GCX setting for the glossy shader-this is better suited to plastic or similar materials that aren't as reflective as metal.

We'll also add a slight noise texture to give us a little distortion in the surface of the rubber:



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.

Adding decals with UV maps

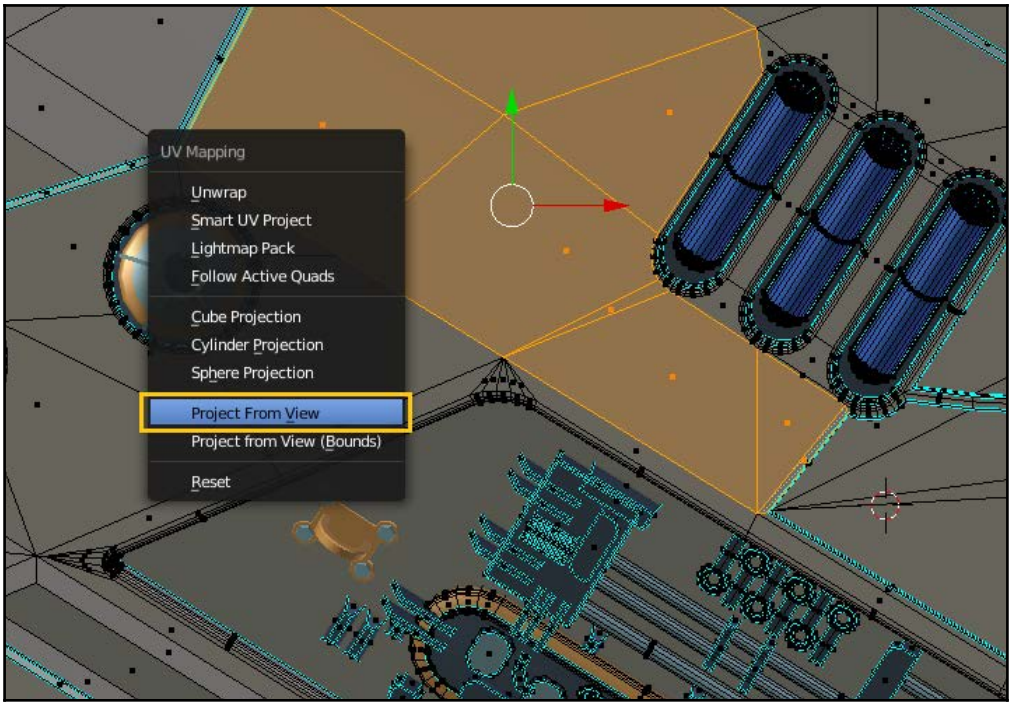
Next, let's add a few decals. Most large vehicles tend to have words, symbols, or logos on them. We can do this easily using **UV maps**.



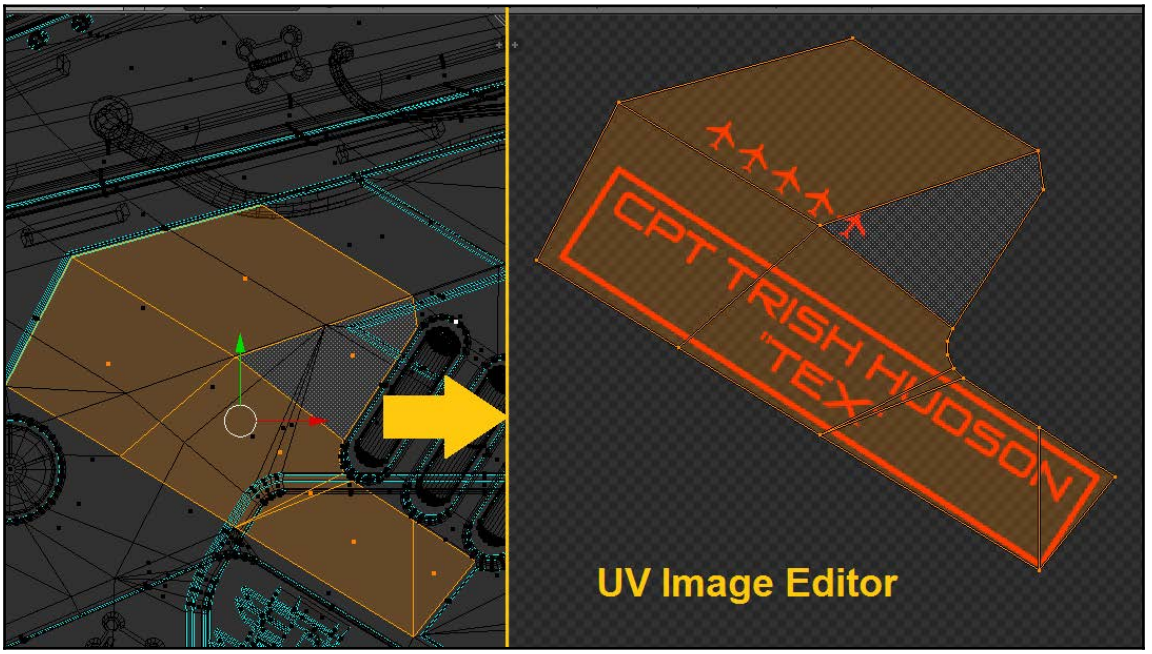
UV mapping tells Blender how to apply a two-dimensional image to three-dimensional objects.

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:**



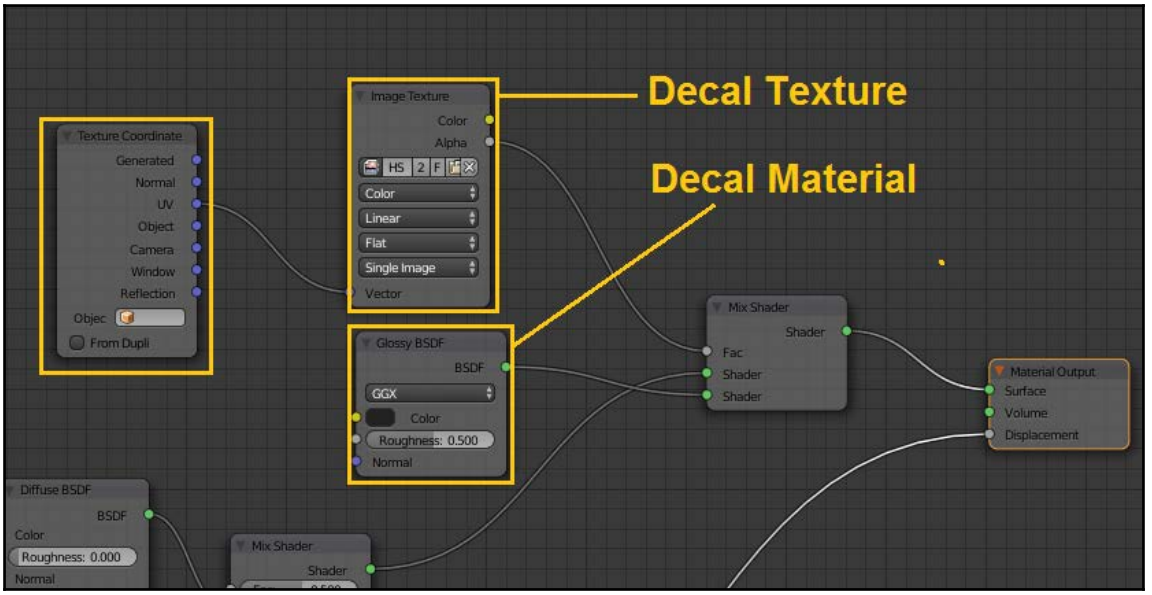
In the **UV/Image Editor**, you can now align those faces with an image of your choice:



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.

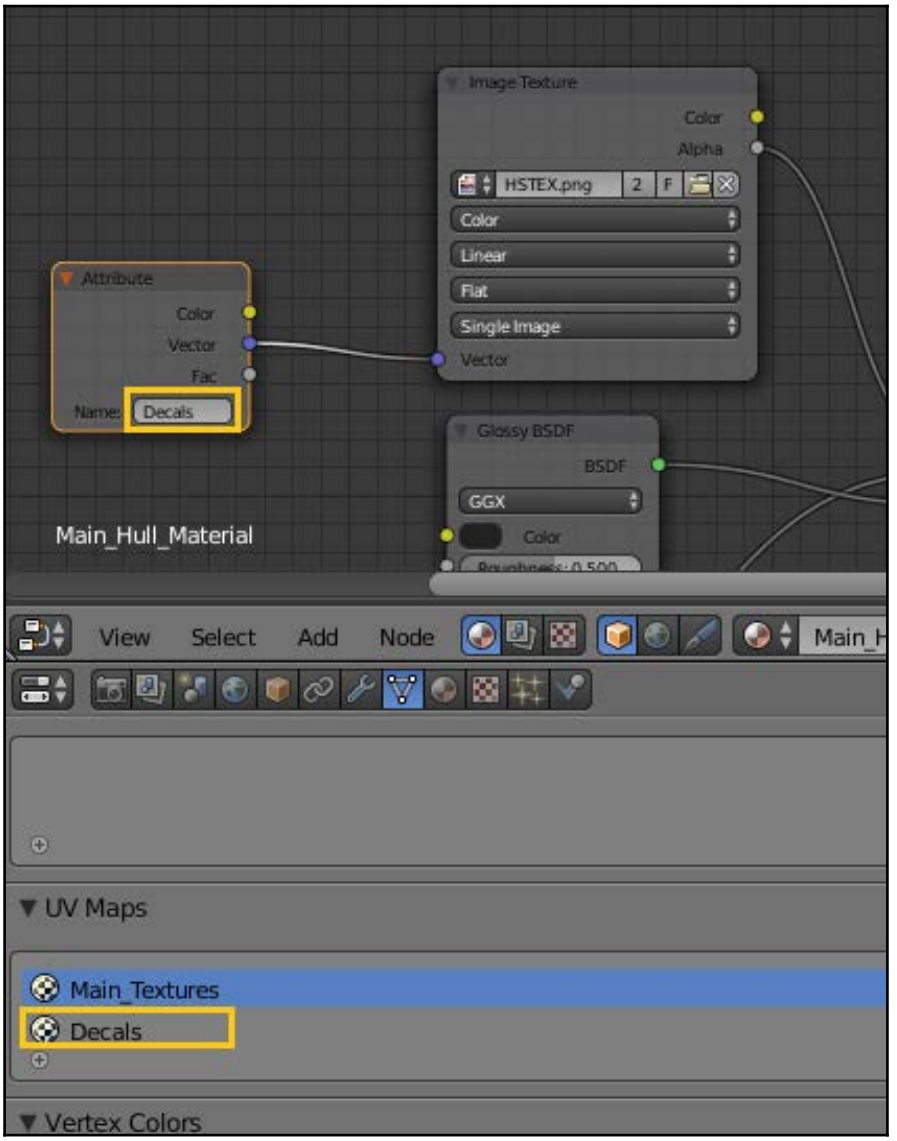
Once you've lined up your UV maps with an image, we'll need to add that image to our material. In this case, we'll ignore the color of the image. Instead, we'll just use its alpha or transparency value to determine which areas of the model will be affected by it.

For our decal color, I'm just using a low-gloss gray color.



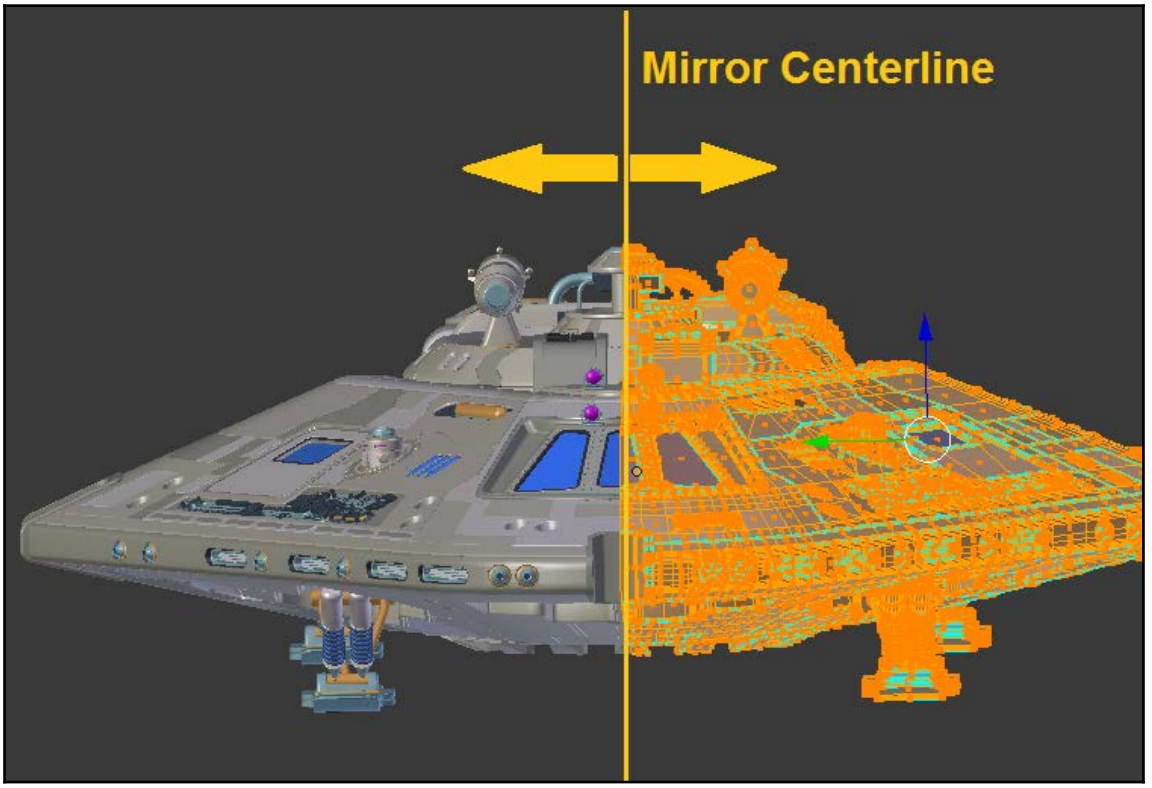
You can see here that I'm using a **Texture Coordinate** node (found under the **Input** submenu). This tells Blender to use the UV coordinates to align the image.

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.



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.



Other possibilities

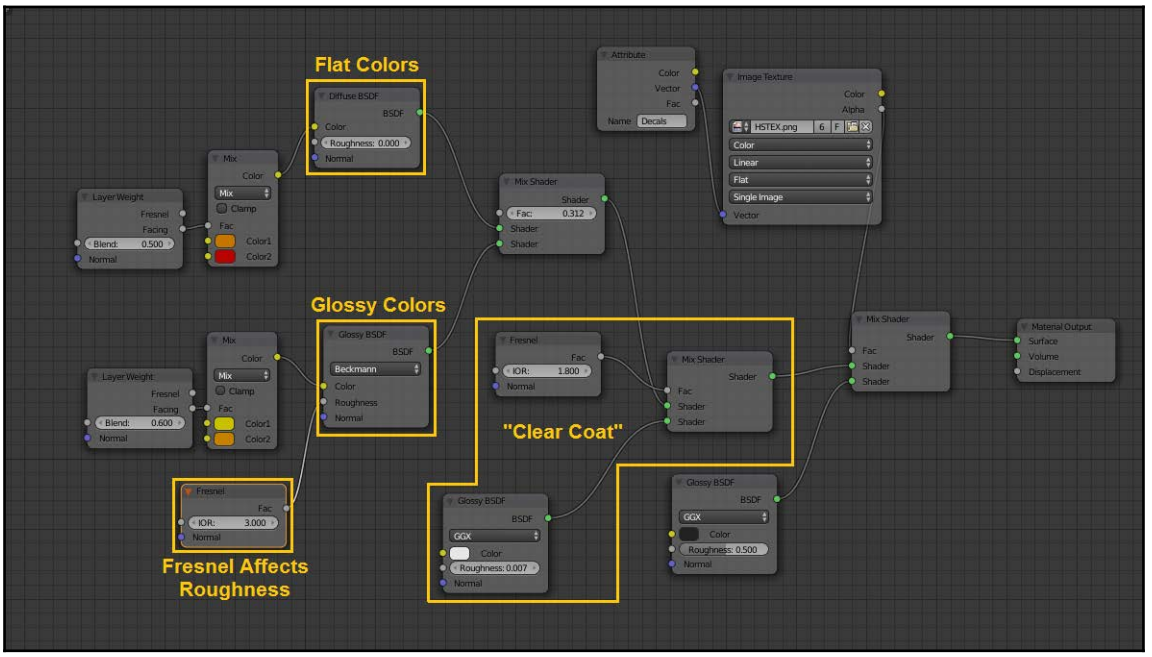
With a few decals, here's what my final ship ended up looking like:



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.

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**:



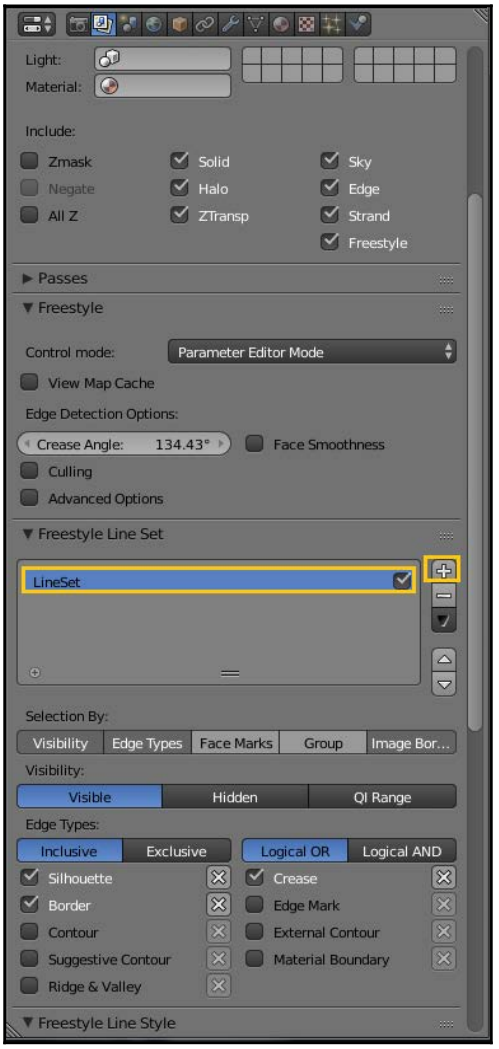


Freestyle is now supported by Cycles as well, but for a long time, that wasn't the case. You could make the argument that Blender Internal (or "BI") is more suited to NPR work. Besides, we haven't used the internal render engine yet in this book, so this is a good opportunity to play with it.

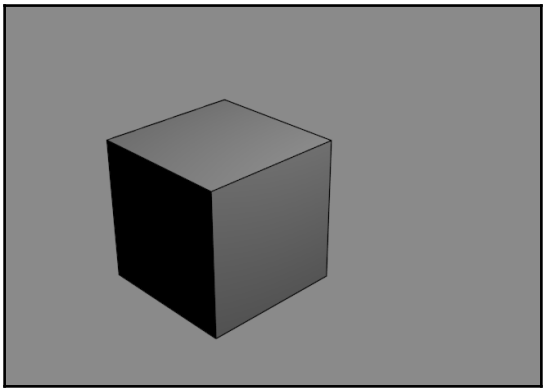
Next, you'll have to check **Freestyle** under your rendering tab:



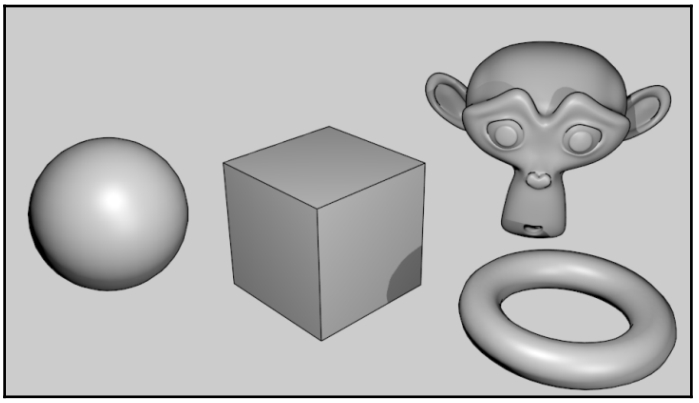
Then, under the **Render layers** tab, you can add a new **LineSet:**



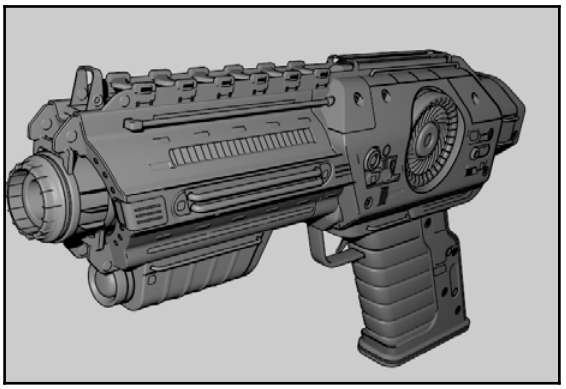
Now, when you render a scene, you will see the freestyle edges:



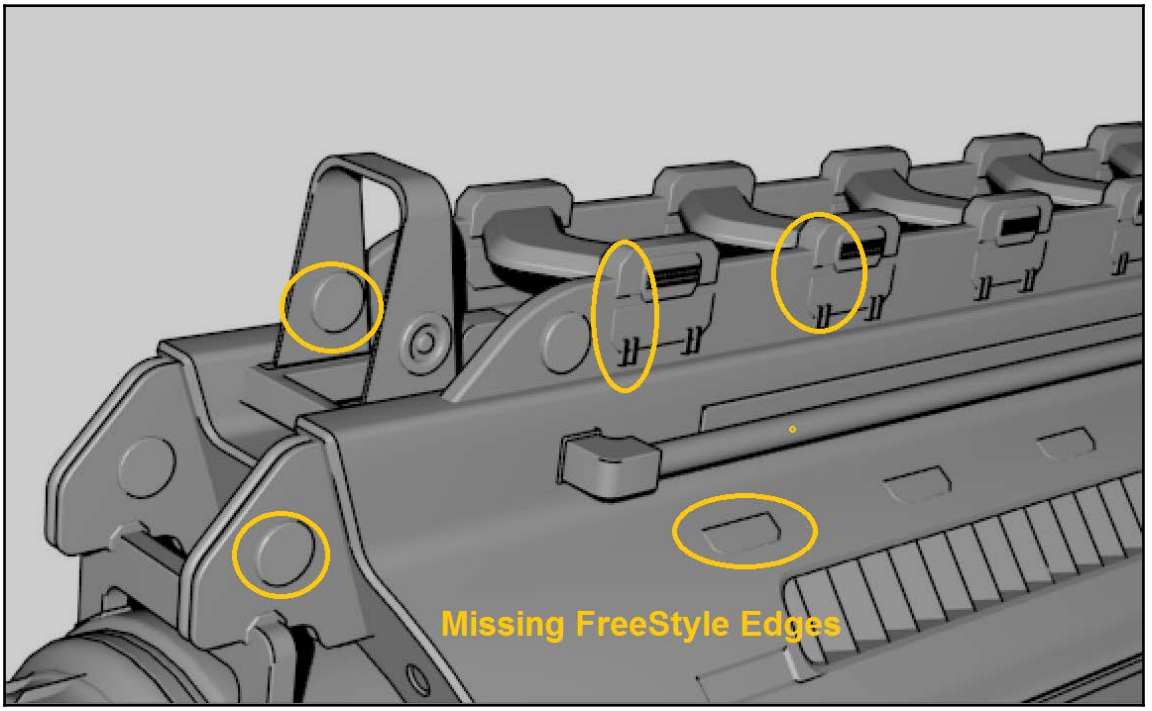
For simple objects like a cube, this is pretty straightforward. Blender knows where the edges are, and quickly fills them in with freestyle lines. Try this with various objects and see how it looks:



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:

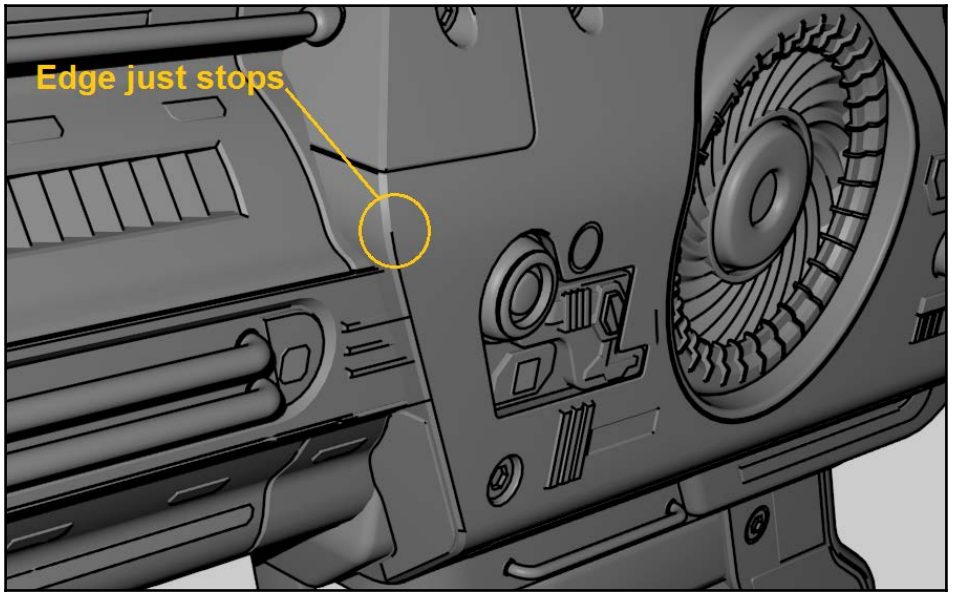


It does not look great. First of all, you can see one of the key problems here:



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.

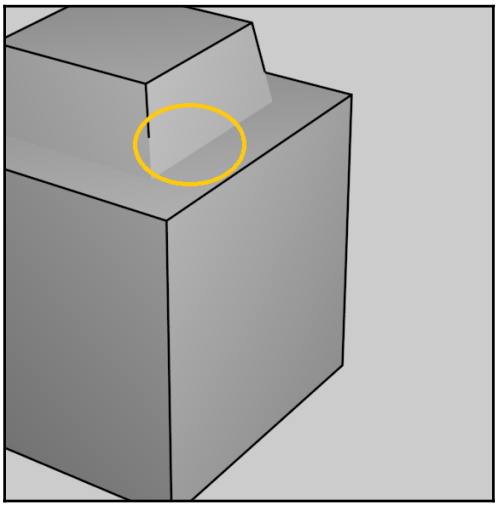
There are other problems as well. For instance, beveled edges don't work well with Freestyle either:



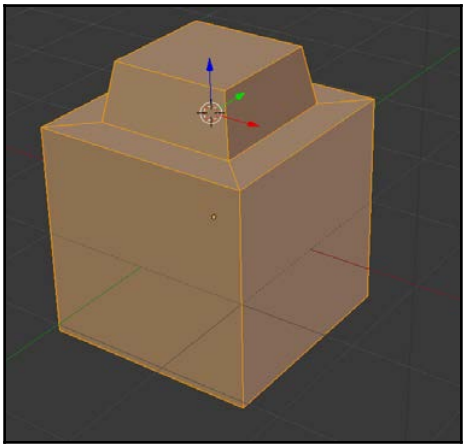
This leads us to a key point:

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.

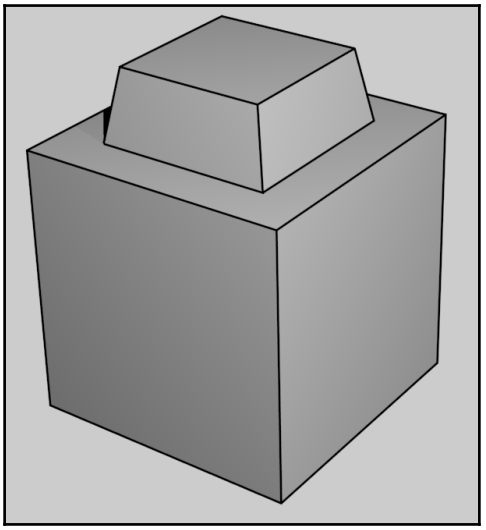
So let's talk about how to do that. First, we'll address the problem of meshes that penetrate other meshes:



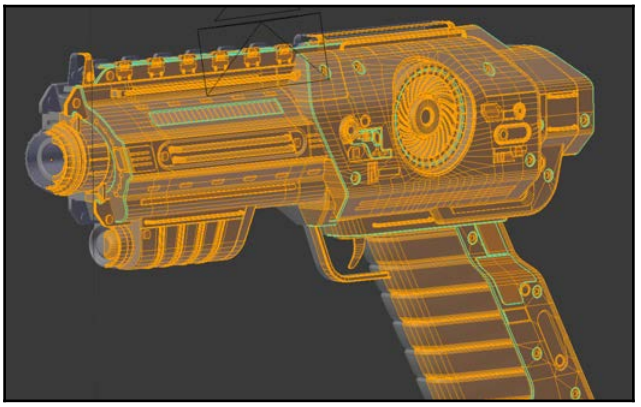
One solution is to connect all the vertices in a mesh, creating *watertight* or *non – manifold* geometry:



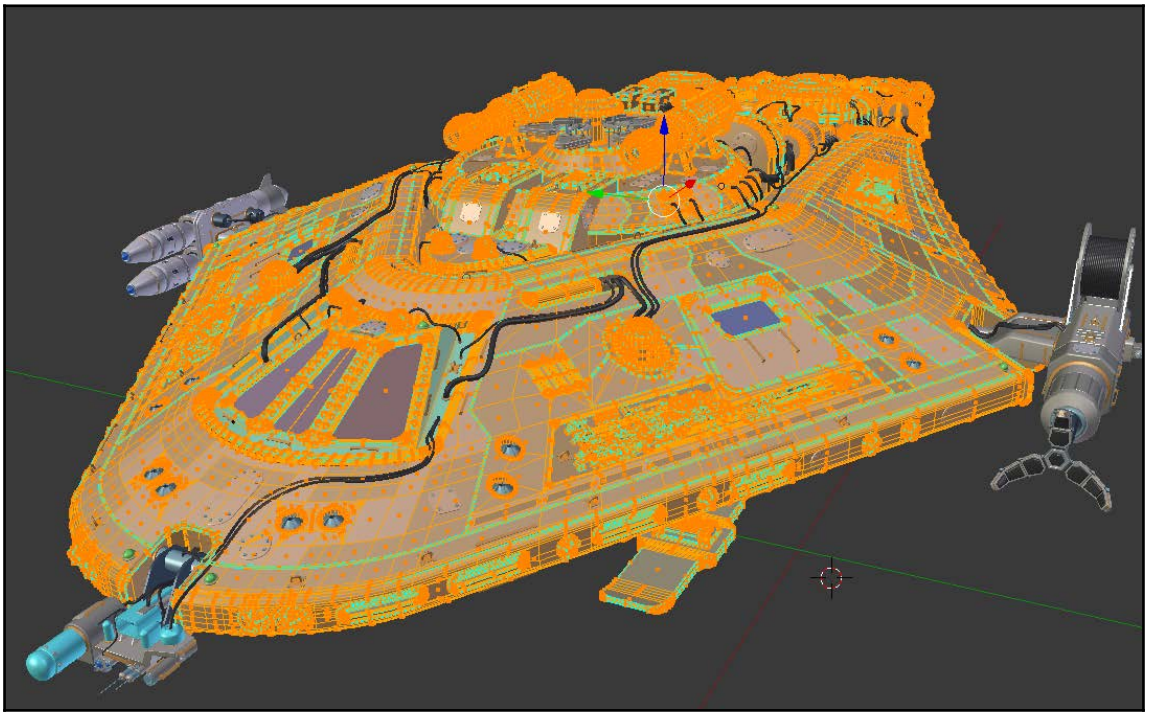
This will certainly solve the problem:



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:

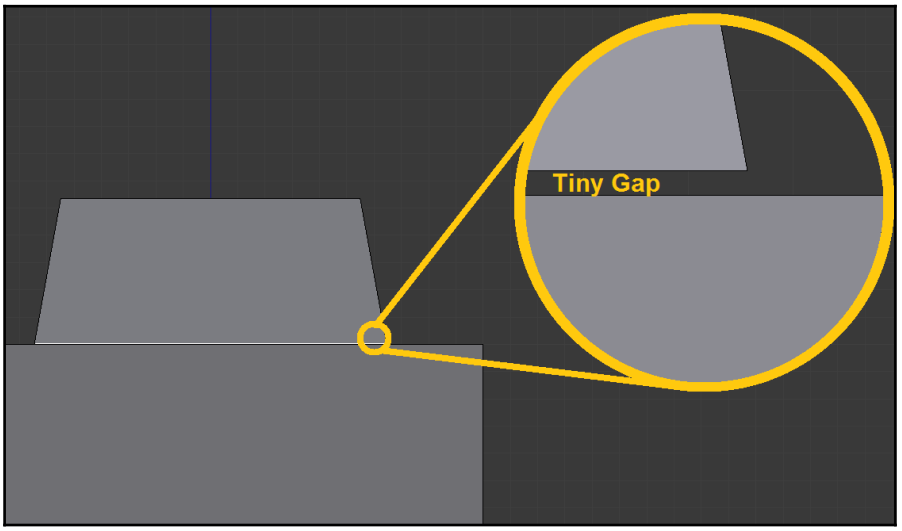


It would be a huge amount of work, and it would drive the polygon count through the roof. You would probably quadruple the geometry in the process. Imagine an even more complex model, like our spaceship:

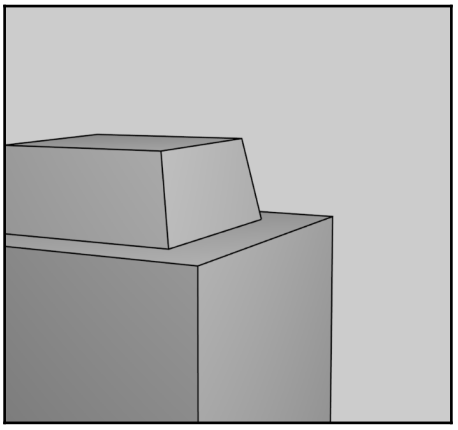


It might require millions of polygons. If we had multiple ships or other objects in a scene, we could quickly overwhelm Blender's ability to handle it. Even if you could handle it, renders would take forever (possibly multiple hours).

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:

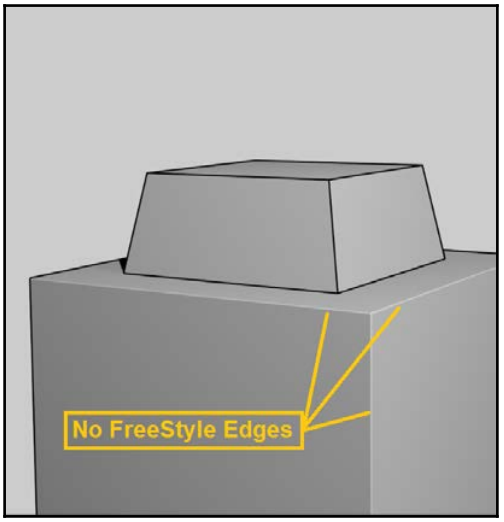


A very small gap will be obscured by the freestyle lines themselves and not even noticeable from most angles:

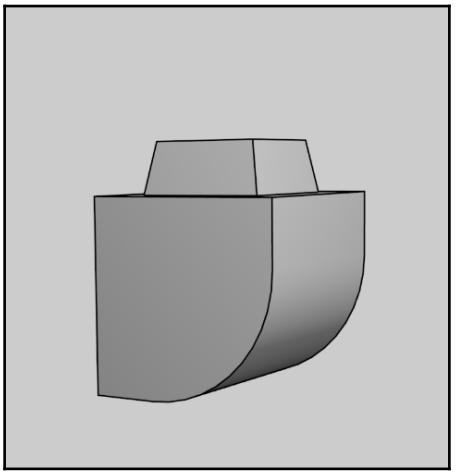


That's a much quicker way that doesn't require so much work (and so much extra geometry).

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:

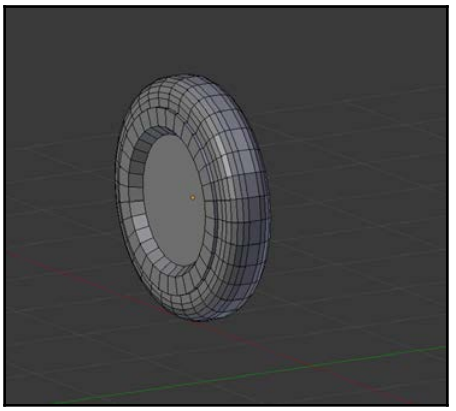


Blocking out your robot

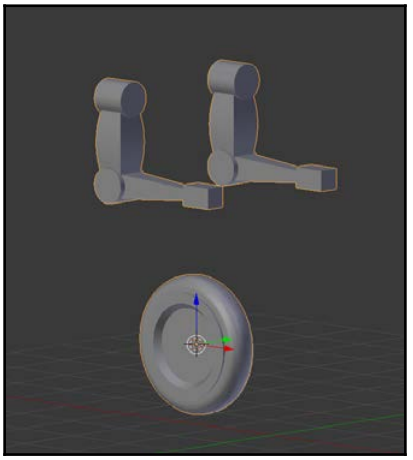
So now, let's move on to building our robot model.

Since this is going to be stylized (not realistic), I'm not going to go crazy with the detailing.

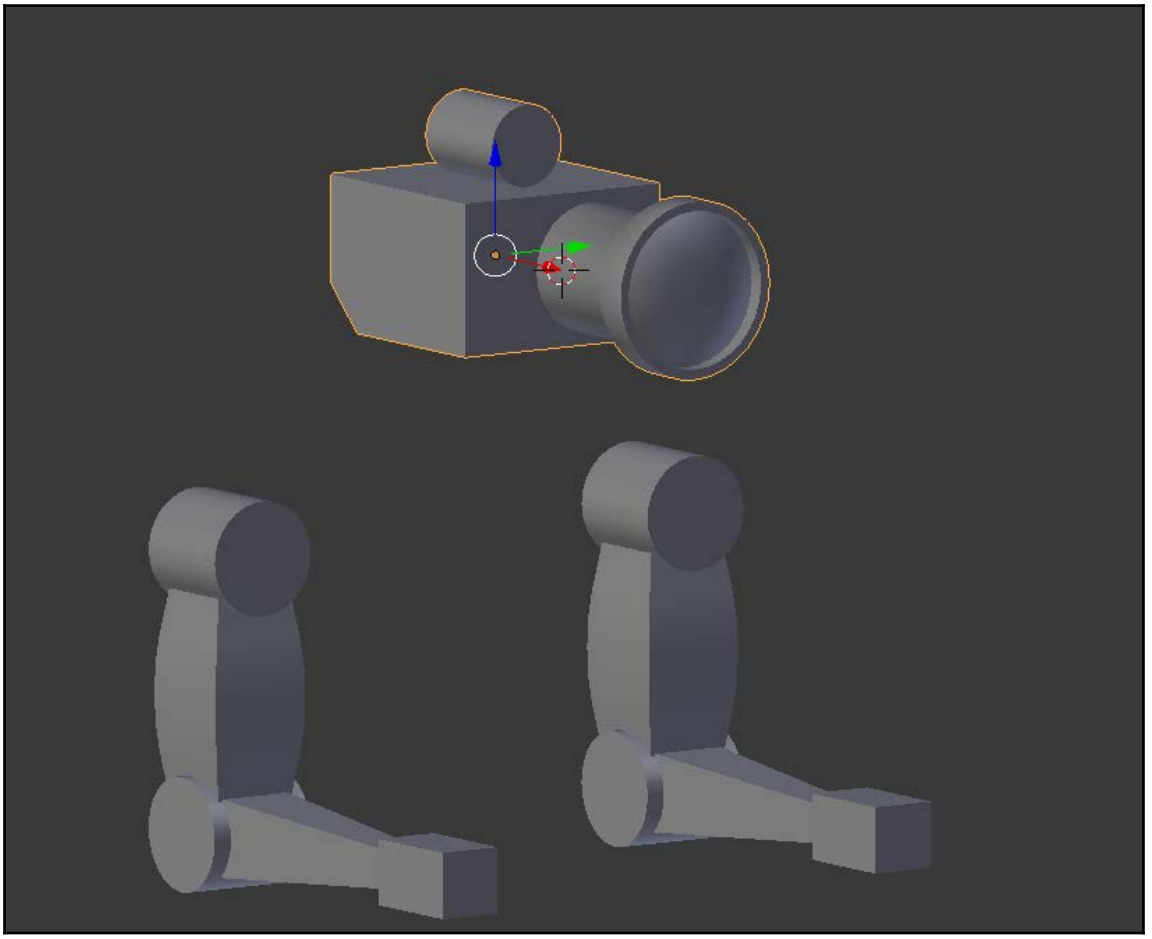
I'll start by blocking out some basic shapes. We'll use a wheel for the robot to roll on:



Some basic arm shapes:



And a head that looks somewhat like a video camera:



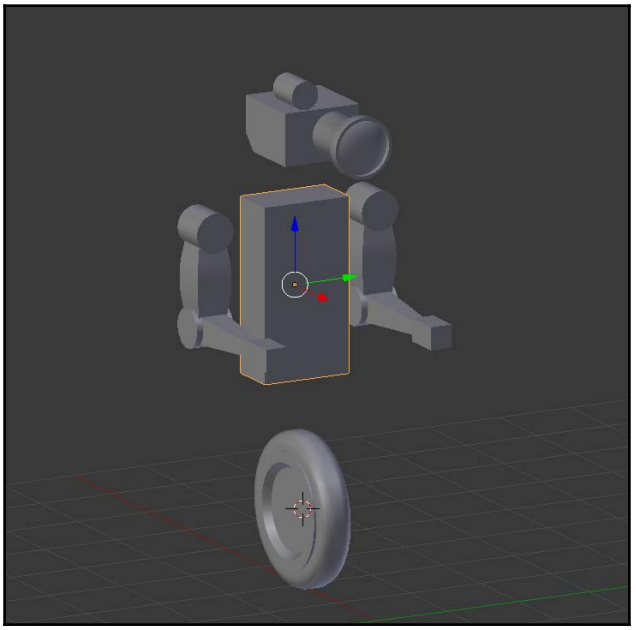
Creating the body with Subdivision Surfacing

For the body, we're going to use a modifier that we haven't talked about yet—**Subdivision Surfacing**.

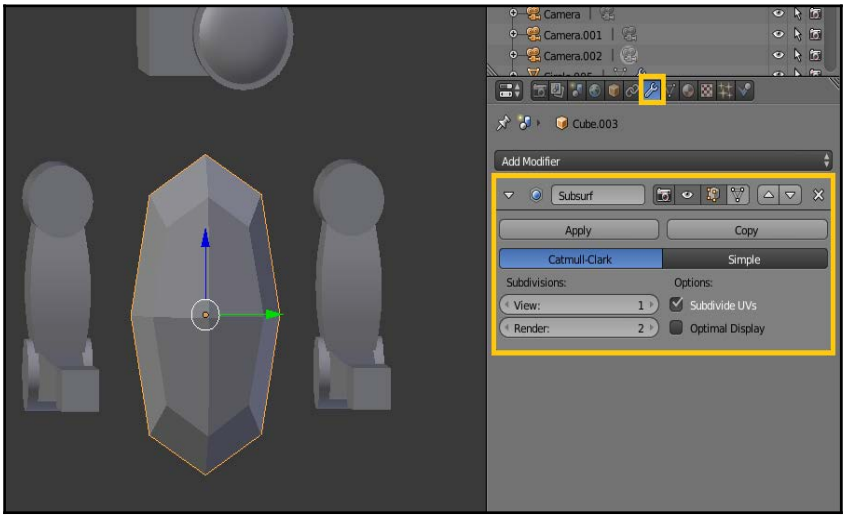
More experienced users may find it odd that we're waiting until this part of the book before using Subdivision Surfacing (or **Subsurf**, as its commonly referred to). After all, it's a tool that many people use frequently.

So, let's talk about that.

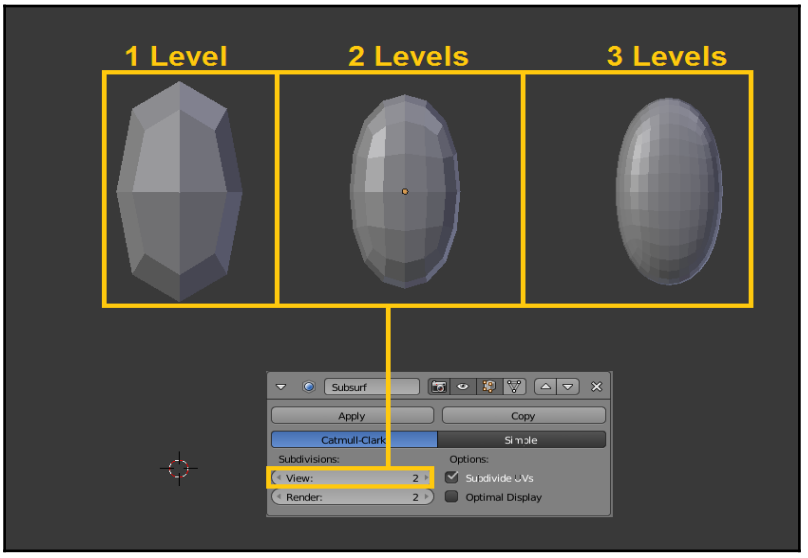
Subsurf works by dividing and smoothing a mesh. For instance, let's add a basic **Cube** shape for the body of our robot:



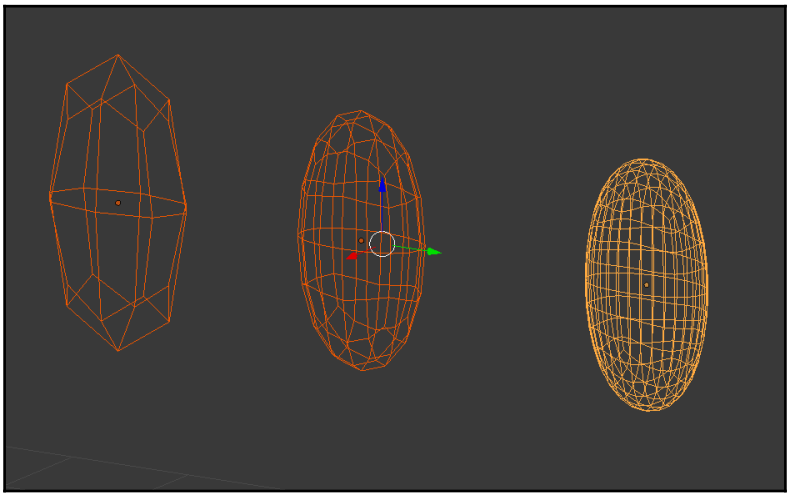
Next, let's add a **Subsurf** modifier to it:



Subsurf has divided and smoothed our cube. The more layers of Subdivision Surfacing you add, the smoother it gets:

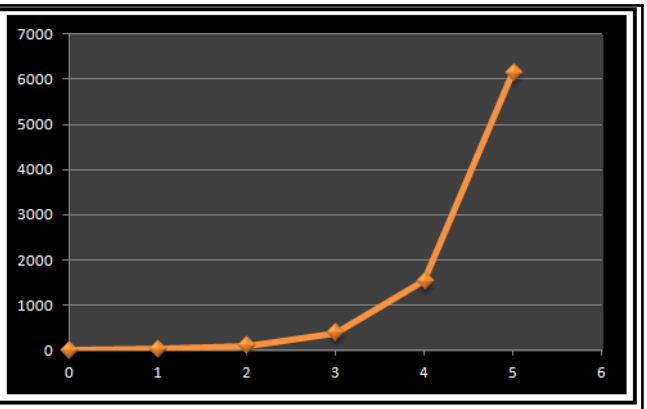


A very important note, however, is that with each layer of subsurfing, the polygon count goes up quite a bit:



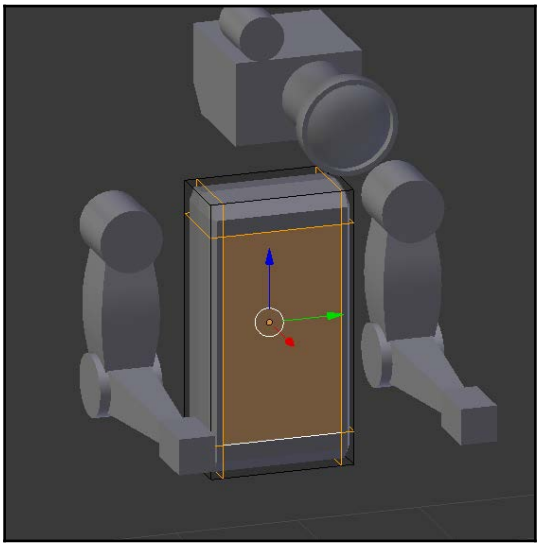
This can be quickly illustrated with a graph:

Levels	# of Polygons
0	6
1	24
2	96
3	384
4	1536
5	6,144

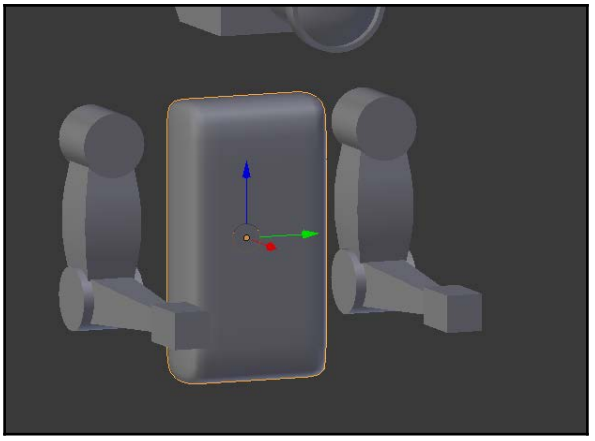


In practice, it would be rare to add more than two or three levels of Subdivision Surfacing. Still, the polygon count is one of the main drawbacks to it. **Subsurf** is a quick way to get smooth, flowing shapes—but, at a cost.

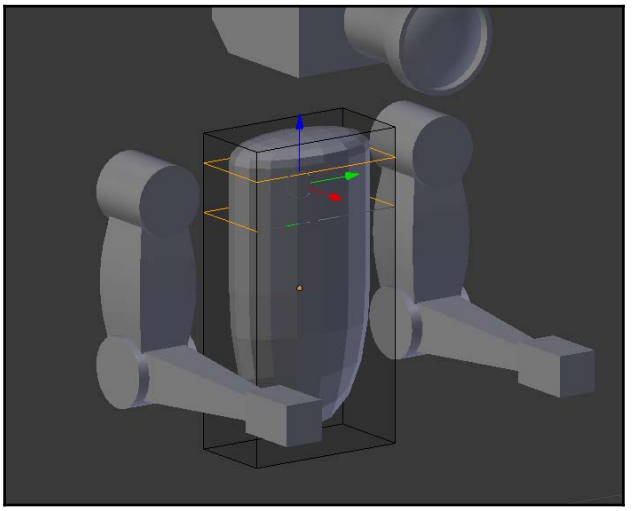
In order to add some definition, we can *Tab* into **Edit mode** and add a few loop cuts:



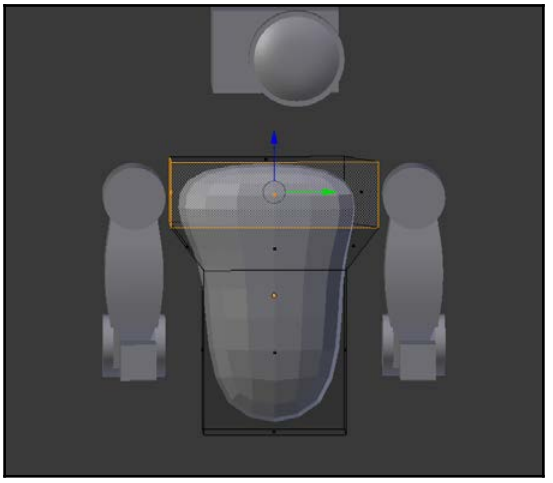
You can see the effect that this has:



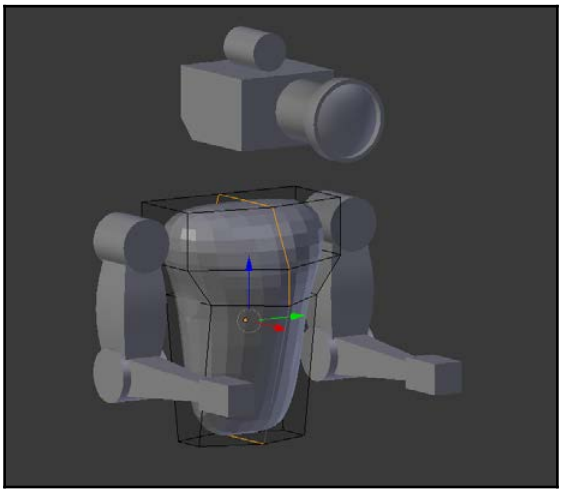
Let's remove those edge loops—it was just for demonstration—and continue building our body. We'll run a couple of edge loops around the upper torso area here:



And widen the torso:

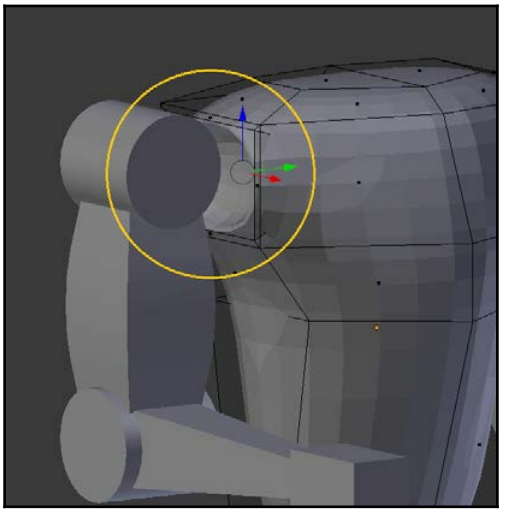


Then, we will run another loop around the body and start to refine the shape a little bit more:

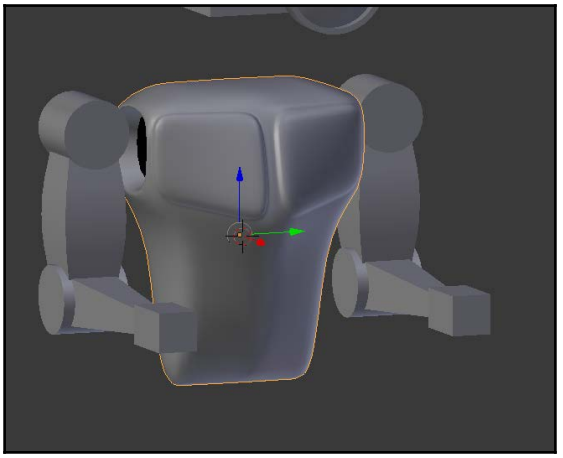


Remember that you don't need to do much (if any) beveling here. We're basically creating a "frame" or "guide" for the **Subsurf** modifier to create our body.

So, I'll next add some arm holes:



And maybe a bit of detail here, like chest plates:

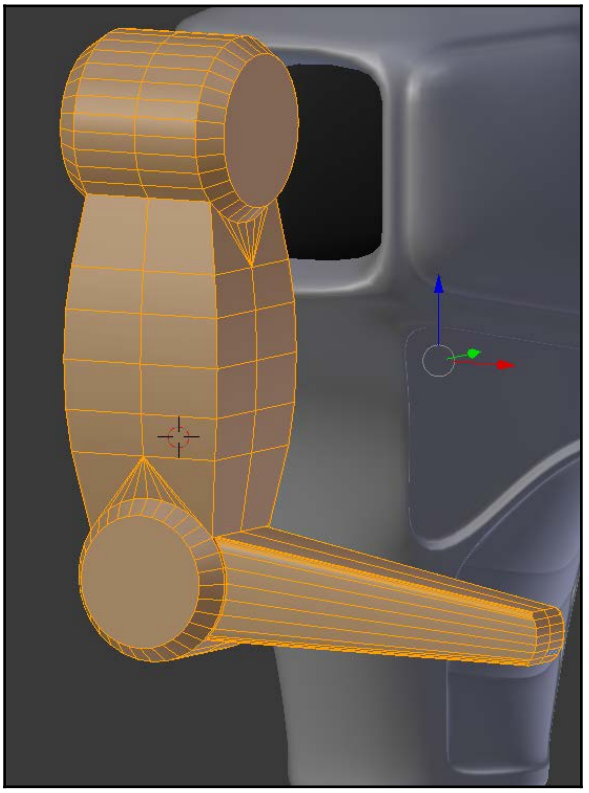


Just continue to add loop cuts and basic extrusions, until you get a body shape that you're happy with:

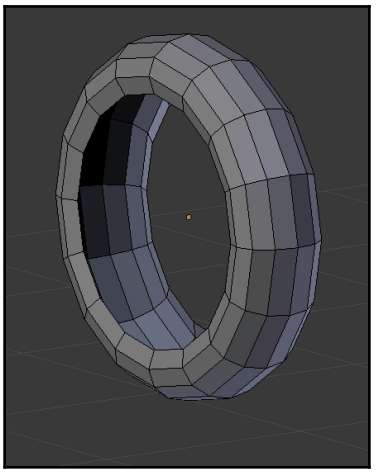


Finishing the robot

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:

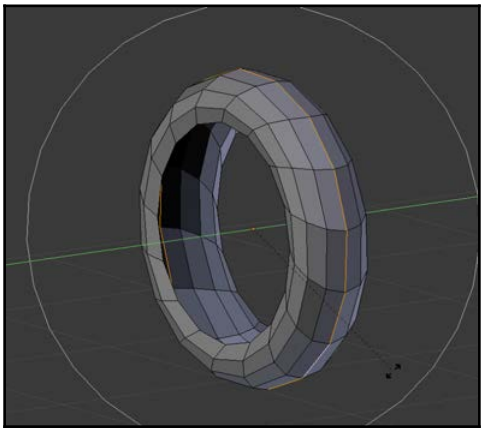


For the wheel at the bottom, we'll use **Subsurf** to create a sort of futuristic tire. We'll start simple:

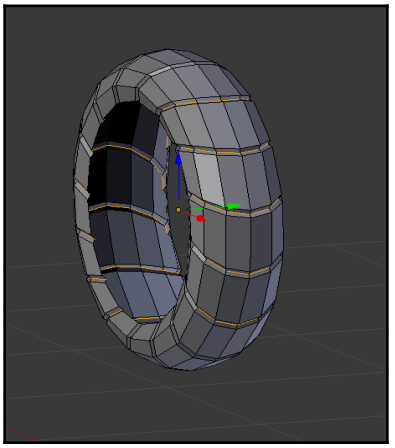


Remember, **Subsurf** will divide and smooth your mesh, so don't use too many sides to your circle originally.

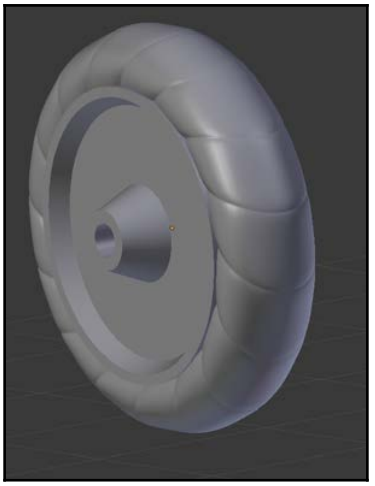
Using proportional editing, I'll spin the central edge loop around the circle to create sort of an “arrow” effect to the tire:



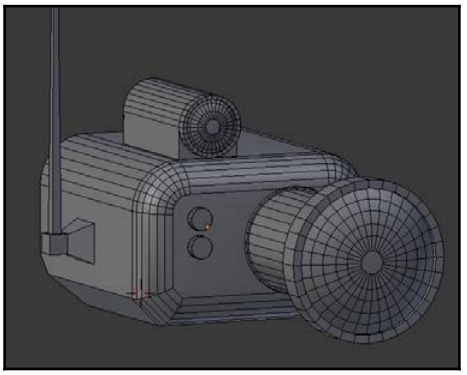
Then, using the **Bevel** tool, I'll create some basic tread:



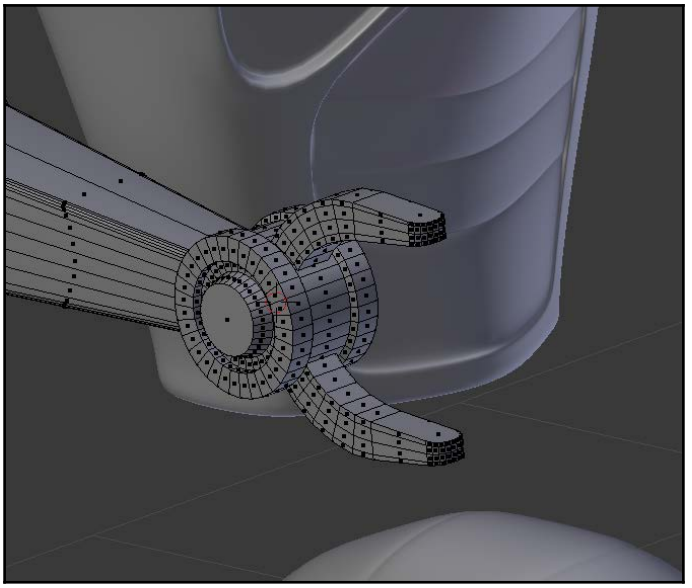
Then, I'll add **Subsurf** and create a basic wheel:



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:

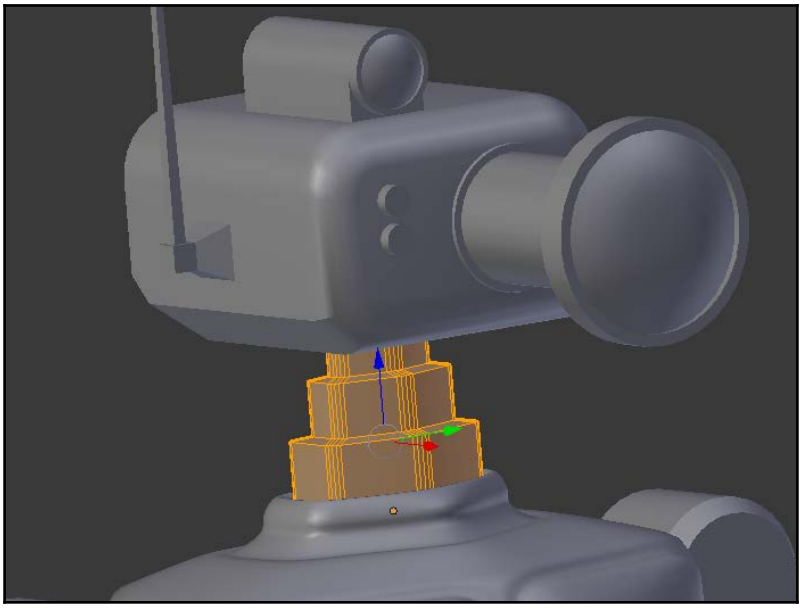


I'm not adding a lot of detail as I do this. When you're going for more of a cartoon or NPR look, that can sometimes be counterproductive.

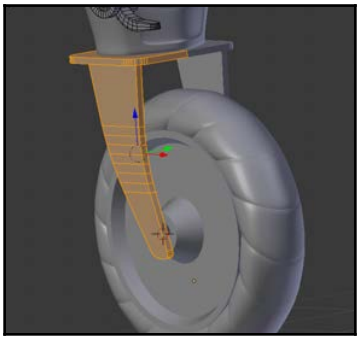


In previous chapters, we've been pretty careful about our use of polygons. We've tried to use as few as possible because there's a multiplying effect. If you have 100 extra polygons that you don't need on some object, and then you have 100 copies of that object, you'll be stuck with 10,000 extra polygons. 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:



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:



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.

8

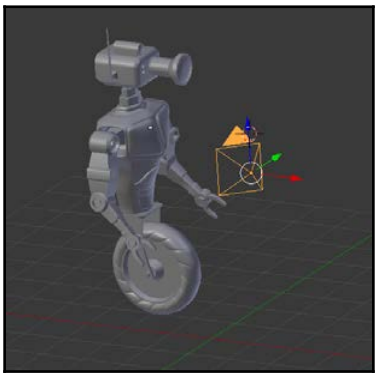
Robot - Freestyle Rendering

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:

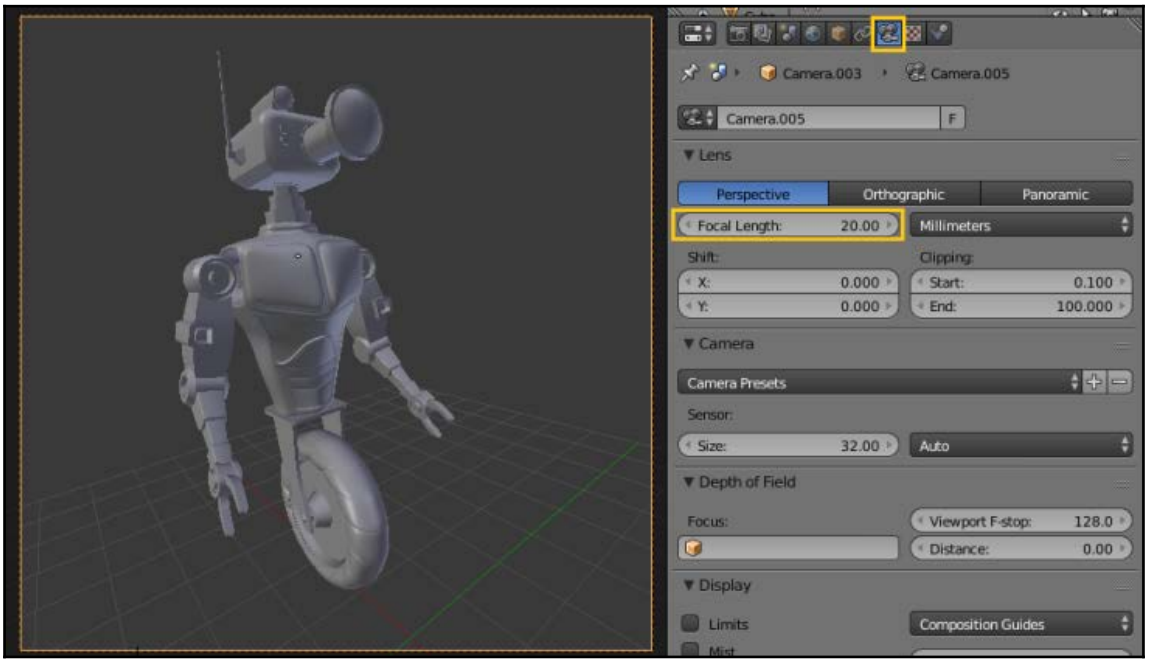
- Preparing the scene
- Marking Freestyle edges
- Creating materials
- Rendering and material options
- Conclusion

Preparing the scene

Before we render our robot, we'll need to set up the scene. First, we'll add a camera:

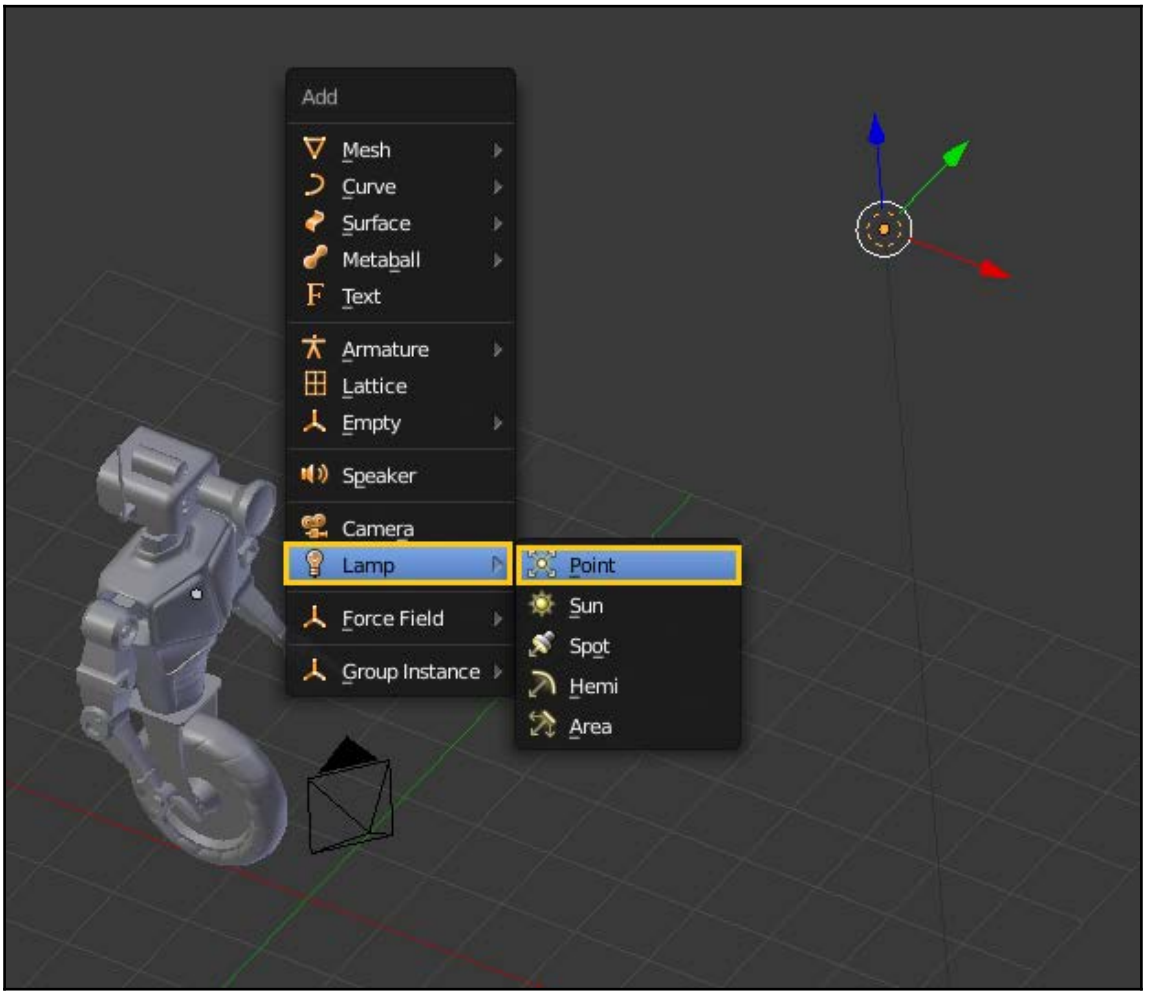


We'll position the camera for the render we'd like to create. If you're doing NPR renders, sometimes it helps to adjust your focal length down quite a bit:



This can help simulate the exaggerated proportions you might find in a comic book or other NPR products.

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.



You can adjust the **Energy** of your light in the **Lamp** panel:



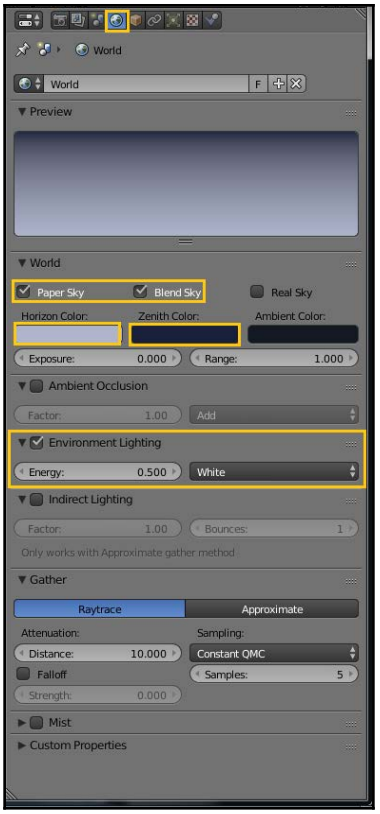
You can also choose either **No Shadow** or **Ray Shadow**. This is different than Cycles, but pretty self-explanatory. You are just choosing whether or not you want your lamp to cast shadows.



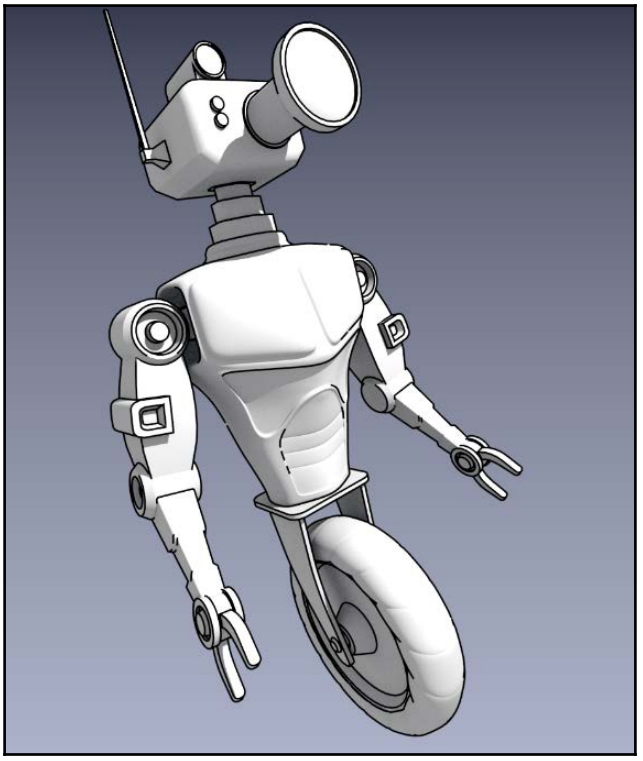
The term **Ray Shadow** is generally only used with Blender Internal. That's because you have the option (with other types of lamps) to choose something called **Buffer Shadow**. A Buffer Shadow is a non-raytraced shadow, which means it's an approximation not based on the actual calculation/reflection 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.



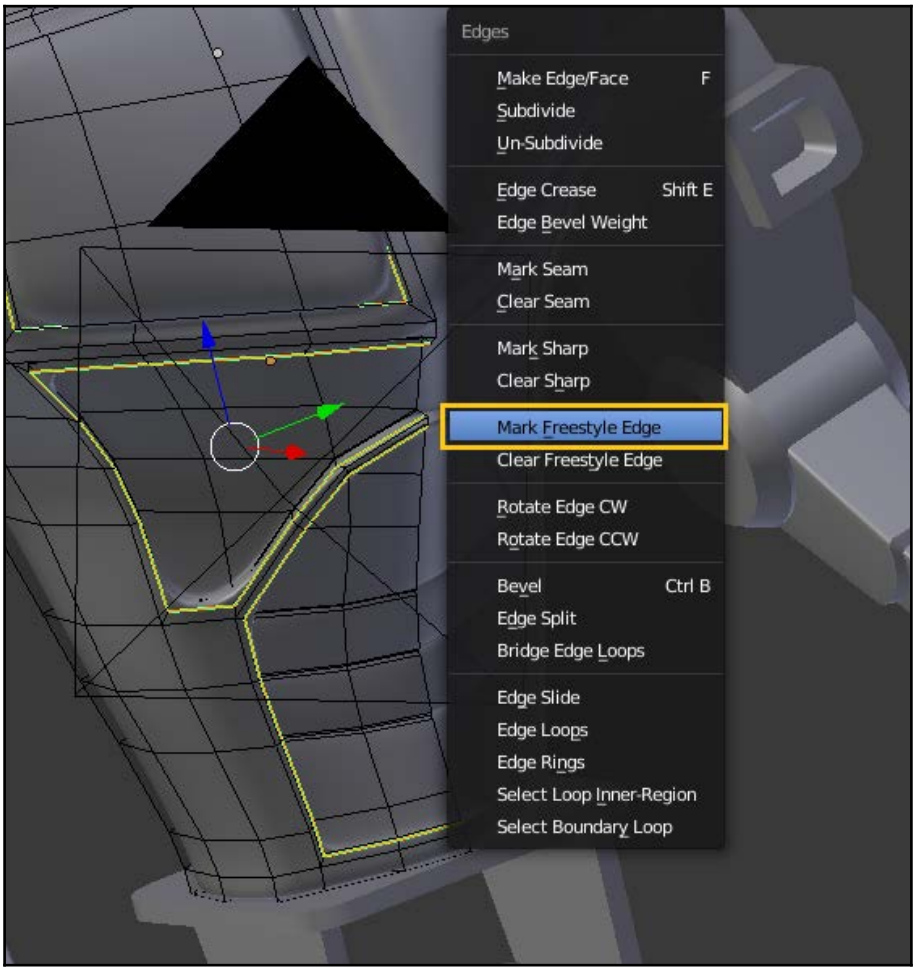
With these settings selected, let's try a Freestyle render of our robot now:



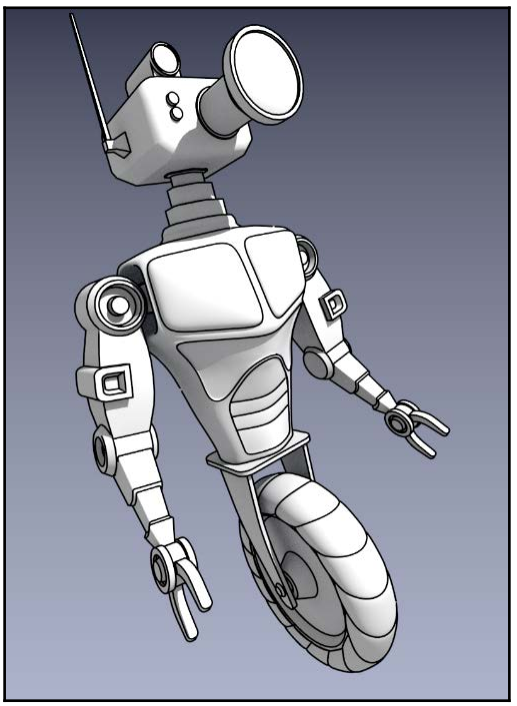
It looks okay, but obviously we've got a few problems. Despite the fact that we modeled it correctly (specifically for Freestyle), Blender still hasn't highlighted all the edges that we want it to. That's okay—there's an easy fix.

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**:



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

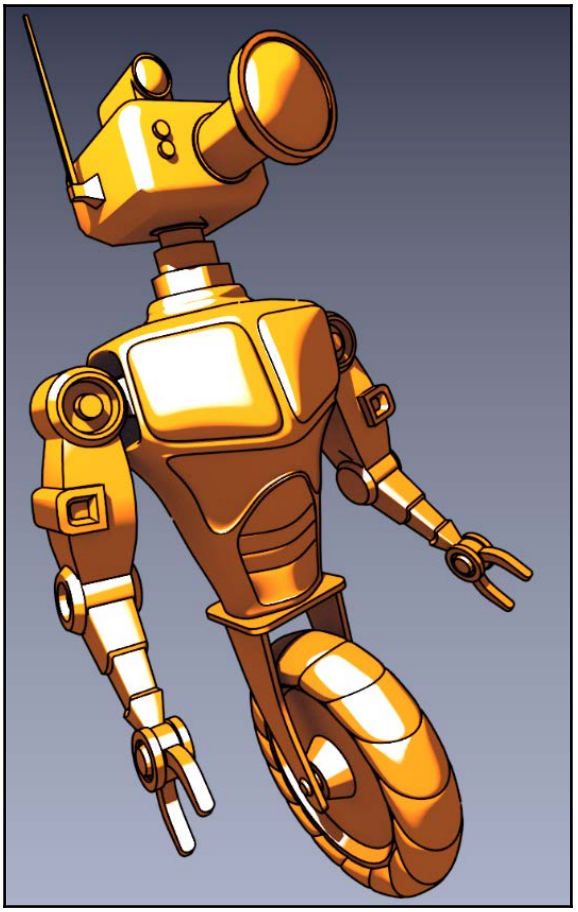
Materials in Blender Internal work a lot differently than they do in Cycles. In some ways, it's easier and more straightforward.

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.

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:

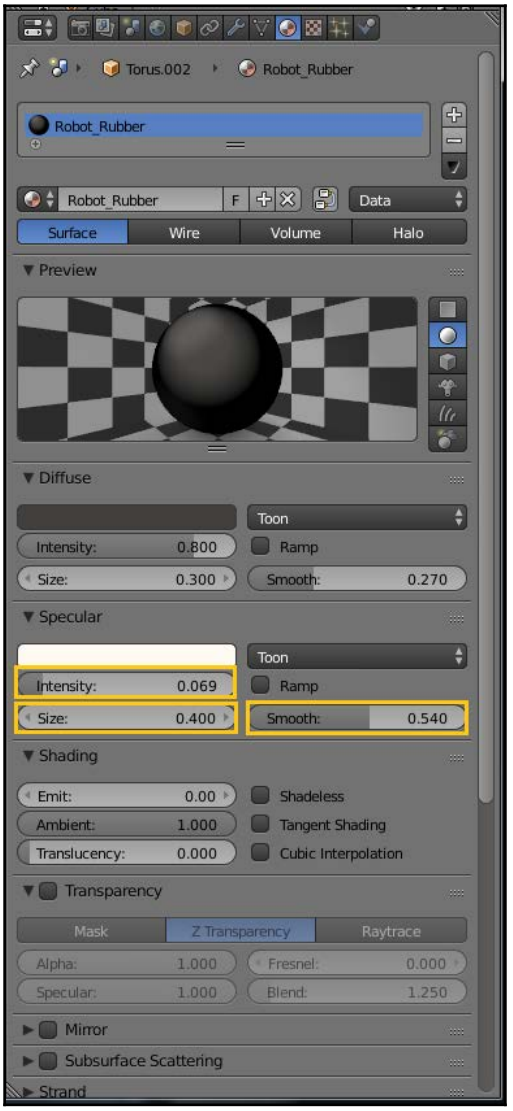


When you've got a decent setup, go ahead and try a quick render:

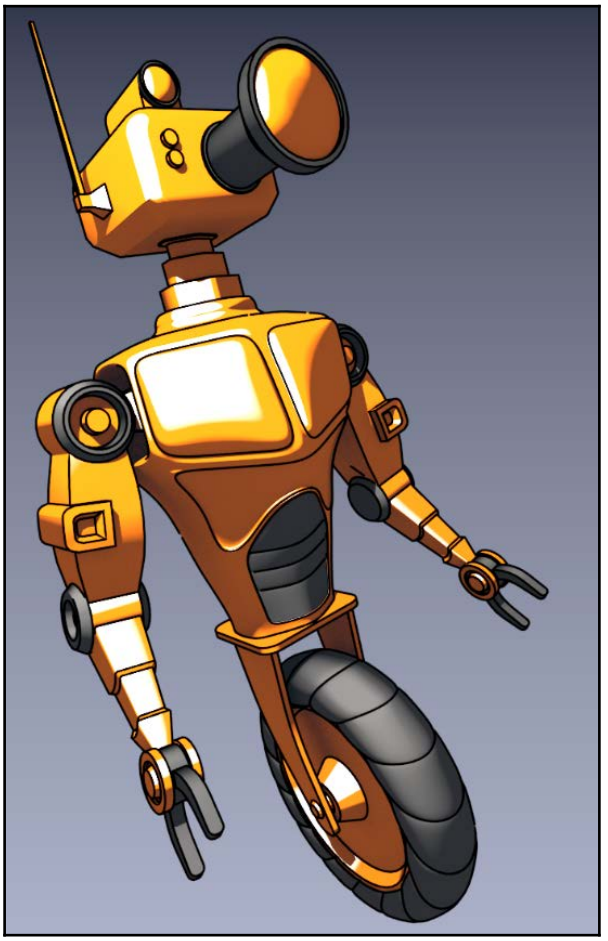


One thing you may notice here is that the scene renders significantly faster than it did in Cycles. We're not getting the added realism that comes with the Cycles rendering engine (because we don't want it), and as a side effect, it takes Blender much less time to produce an image.

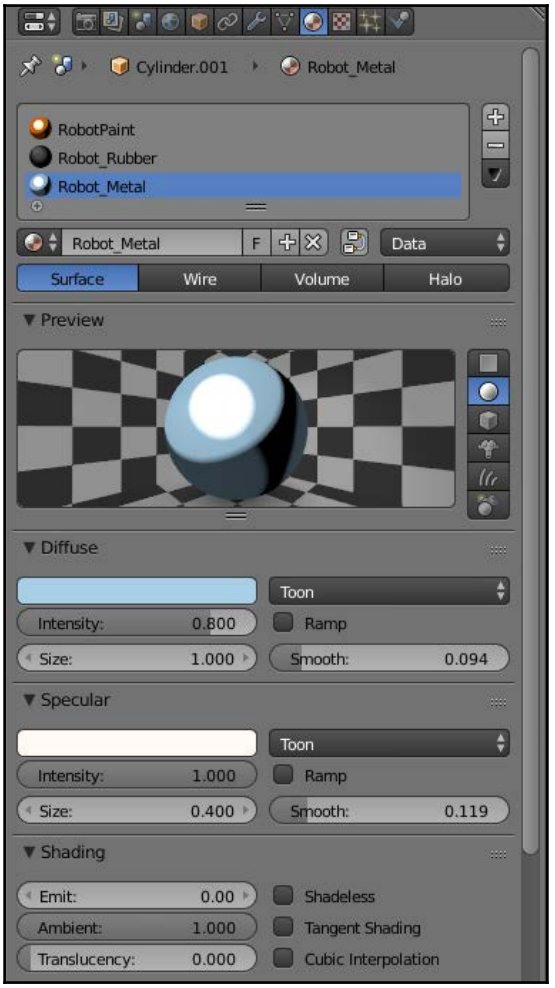
At this point, you can adjust your various material settings the way you'd like to. When you're happy with the orange paint (or whatever color you're using), let's create a second material for rubber:



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:



That's starting to look better. I'll also create a metal material to highlight a few parts:

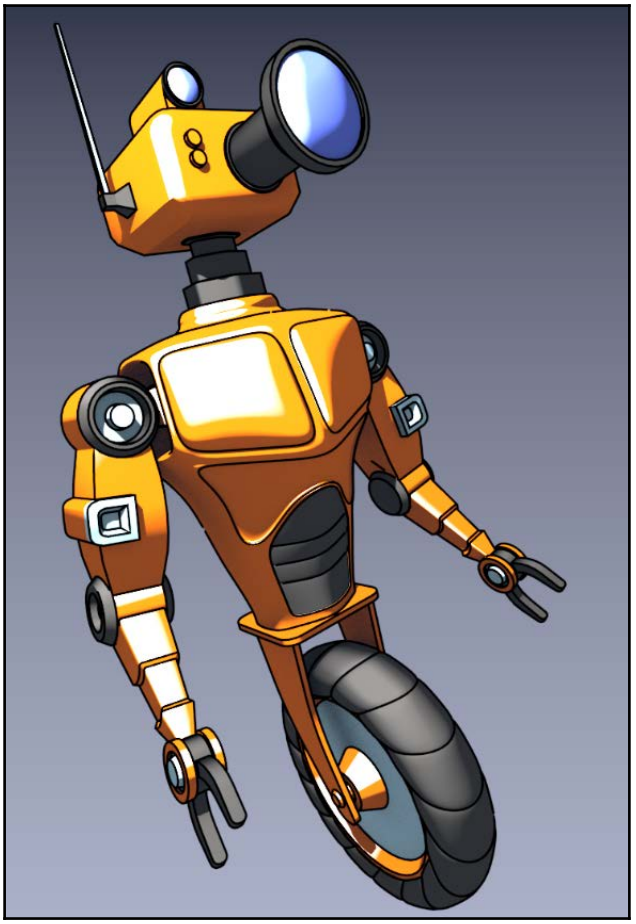


Finally, we need a glass material for the two camera lenses. Note that this isn't actually glass (in the Cycles sense), since it has no reflection or transparency. That's what we want for an NPR render though, particularly since there are no other objects in the scene (and thus, the glass would have 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.

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:



After we've assigned our new materials, let's render it again:

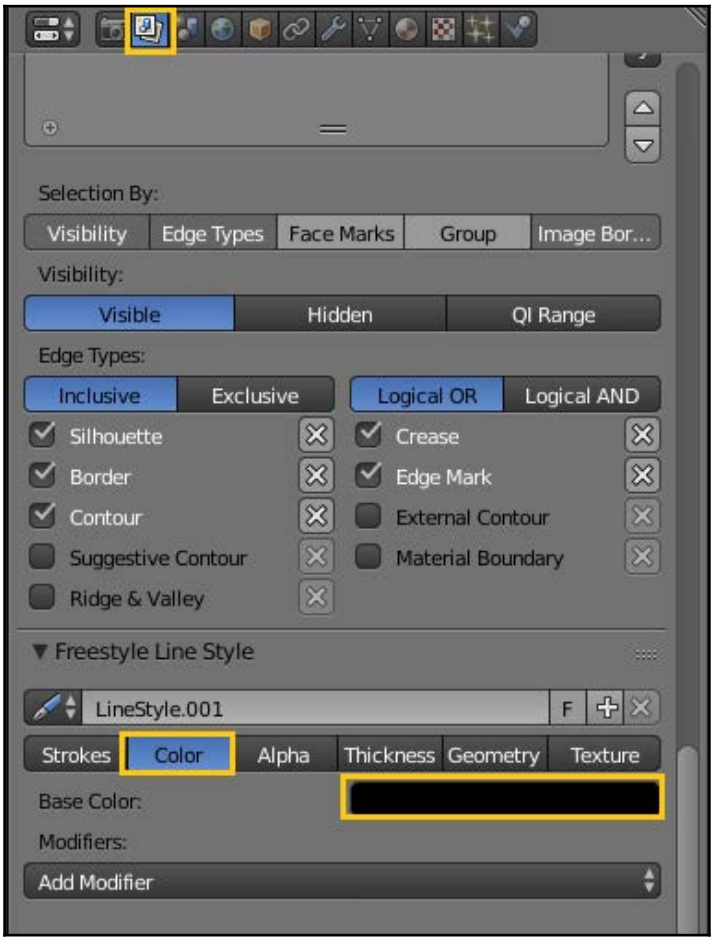


That looks pretty decent for a cartoon robot.

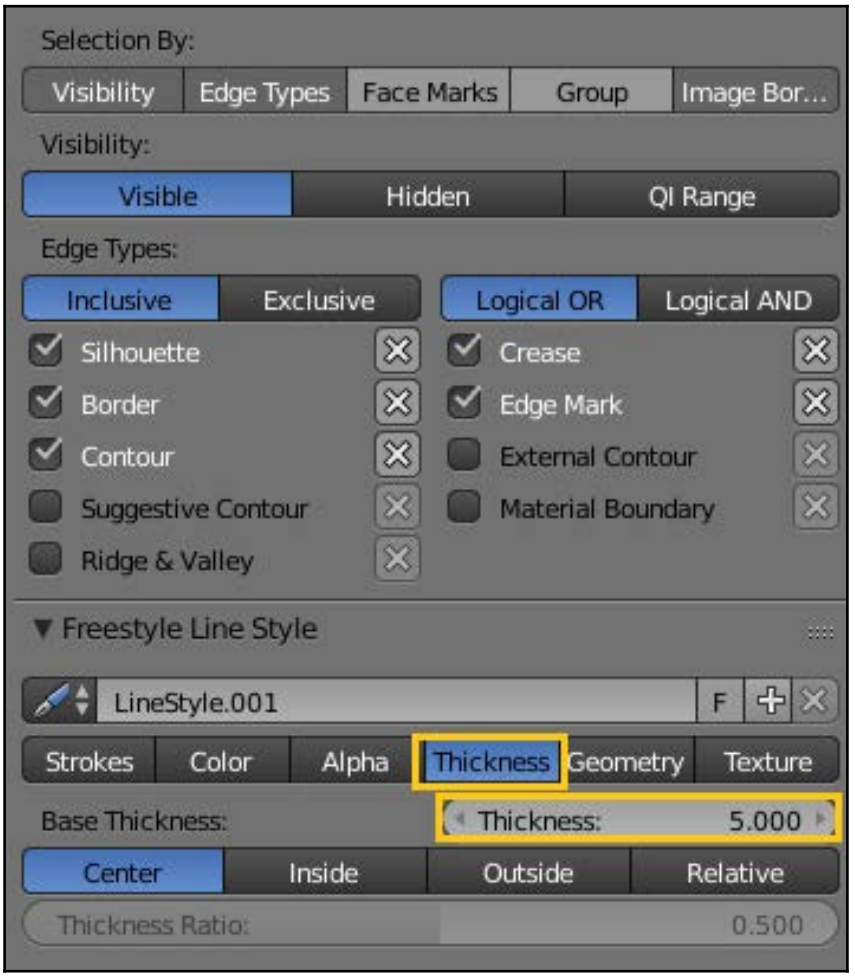
Next, let's take a look at a few options that we have.

Rendering and material options

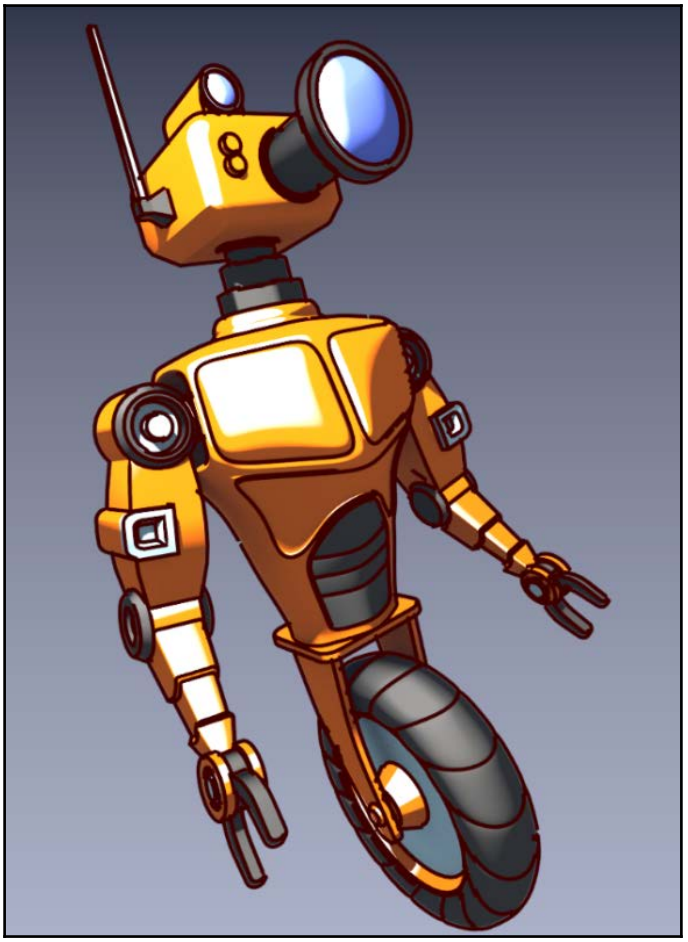
In the **Layers** tab, let's take a look at our **Freestyle** options. One thing we can easily adjust is the color of our lines here:



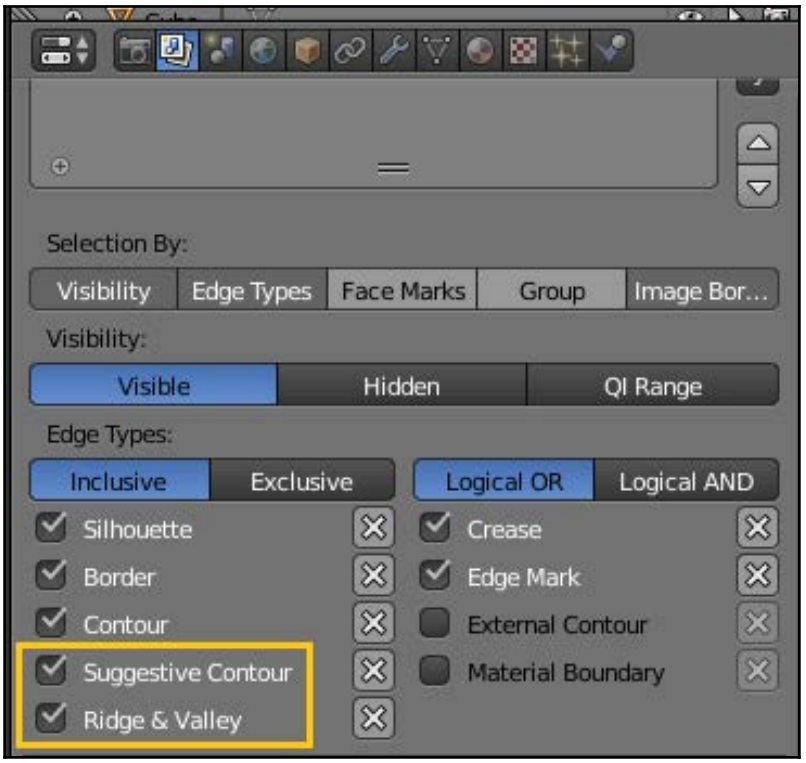
Black is more traditional for cartoons, but you can certainly experiment. Another thing we can easily adjust is the thickness of our lines:



When we render with thicker red lines, here's what it looks like:



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:

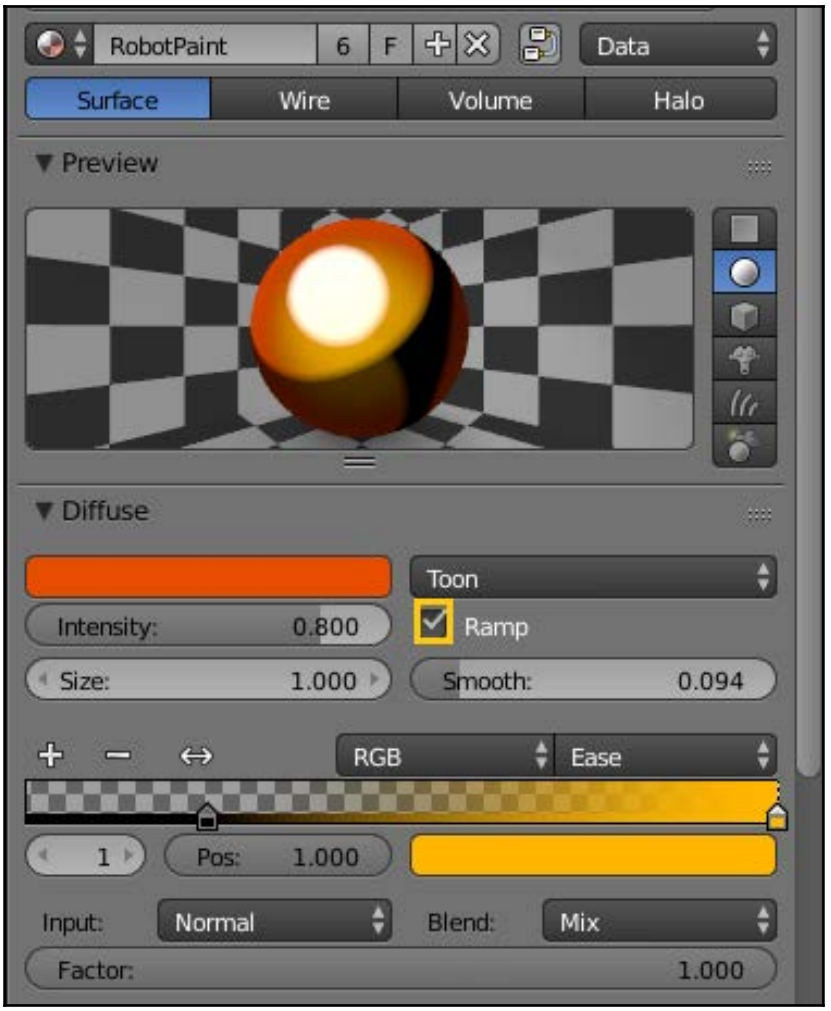


As you can see, this has quite an interesting effect on our render:

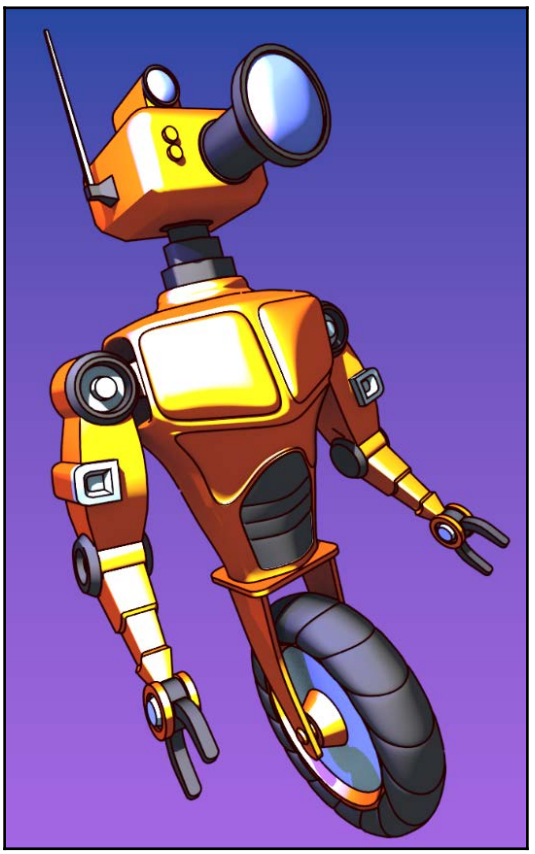


For our robot, it just looks messy—I don't think we'll use it. But if you were doing some type of landscape or other smooth, flowing shape, it might help draw out a bit of detail.

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:



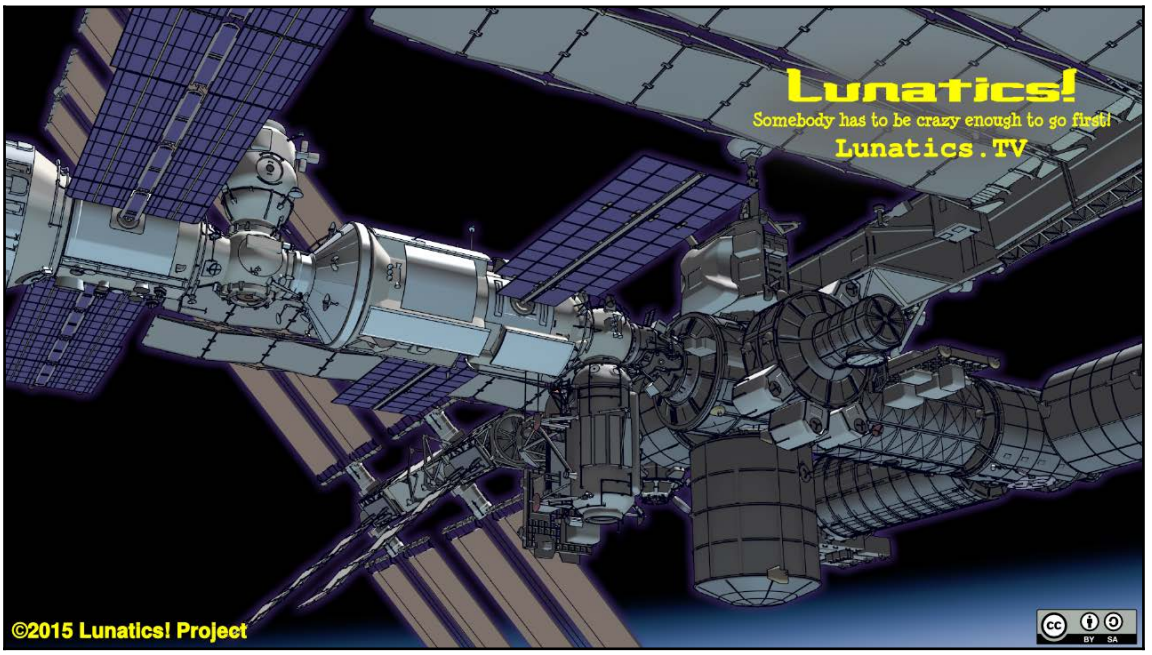
It can be used to simulate color changing paint, glass, or anything else where the color varies with the angle. The setup shown is pretty basic, but I think it makes the robot look a little bit better:



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.

Just as an example of what you can do, here's a Freestyle render of the International Space Station from the web series *Lunatics!* (`http://lunatics.tv/`, 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.

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.

9
Low-Poly Racer – Building the Mesh

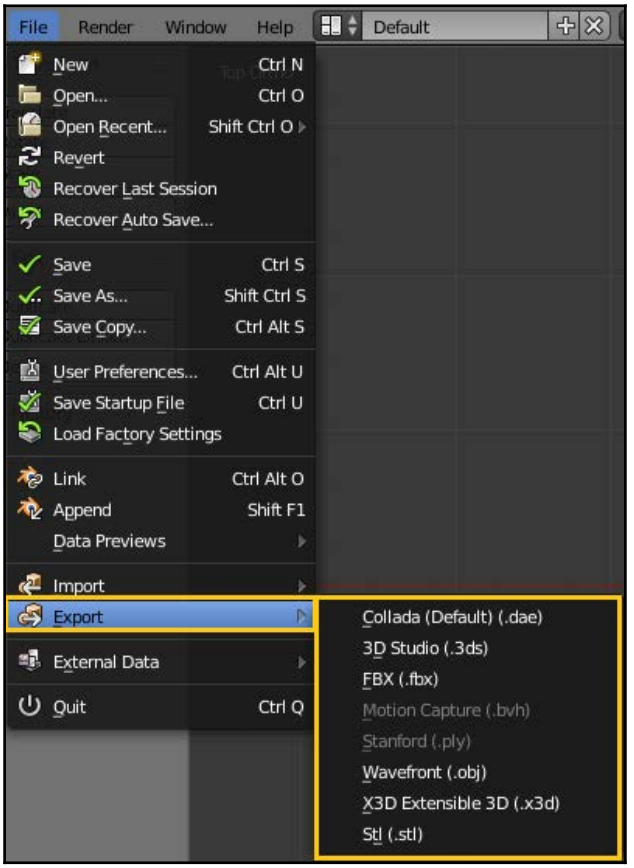
In this chapter, we'll build a low-poly model for use in a racing game. We'll look at several ways that game models differ from the projects we've created so far. We'll also look at some pros and cons of various methods, and important considerations when building these types of models.

- 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.

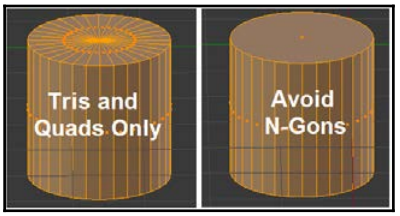
For instance, Blender is an excellent tool for creating 3D game models. You can build and texture the model in Blender and then export it to other formats. Let's take a quick look at our export menu to see what options we have:



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.

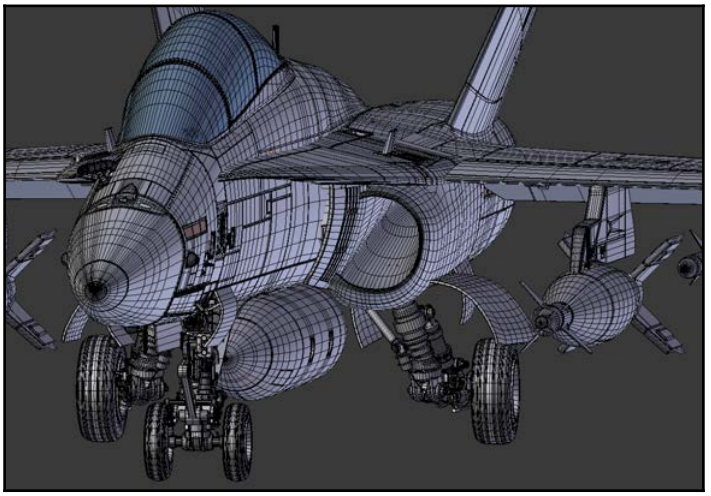
When you're talking about export and conversion, it's important to note that not everything carries over from Blender. You'll want to create your models in certain ways to prevent data loss and maximize compatibility.

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:

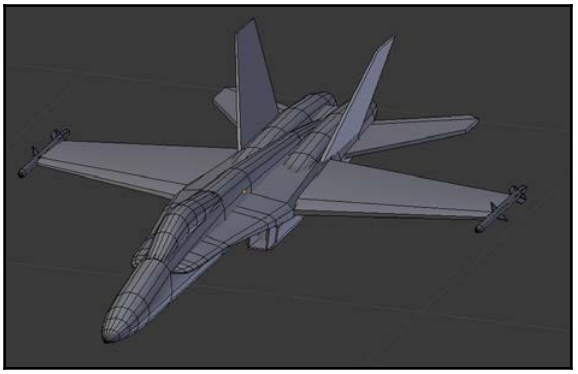


The polygon count is very important for these types of models as well. If you're familiar with flight simulators, you'll know that a typical aircraft looks far less realistic in a game than it does in a Hollywood movie. Both are 3D models, but they have very different requirements and uses.

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:



By contrast, here's another F-18 model I did for a simulation:

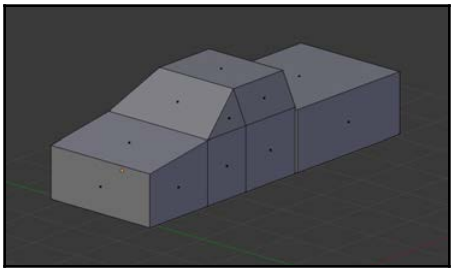


You can see that it's far less detailed because it needs to be rendered in real time (in other words, it's constantly rendered as the user "flies" the airplane). So as we build our model, we'll need to keep the polygon count as low as possible.

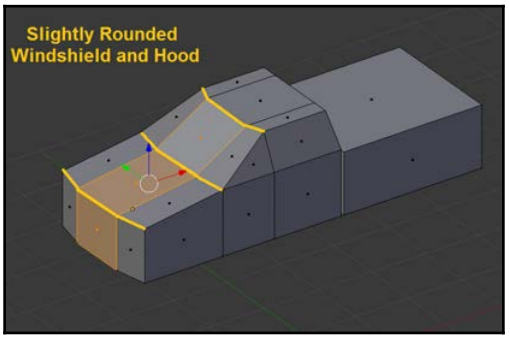
How low does it need to be? Well, that really depends on the application. Some truck models (like the kind we'll be making) may only be few dozen polygons, while others can reach into the tens of thousands. It all depends on your application. For this project, we'll shoot for around 5,000 polygons (or 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

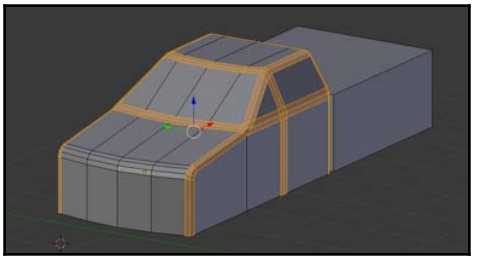
Let's start by creating the basic shape of our vehicle. We'll make this an off-road racing truck, so let's just block out the basic shapes. We'll start with the rough dimensions of the body:



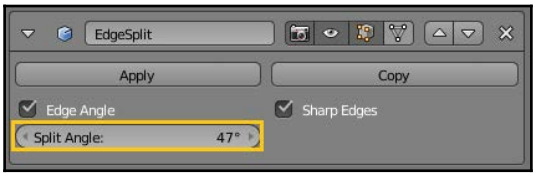
Then, I'll just run a couple of loop cuts around the cab and add a slight curve to the front of the truck:



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.

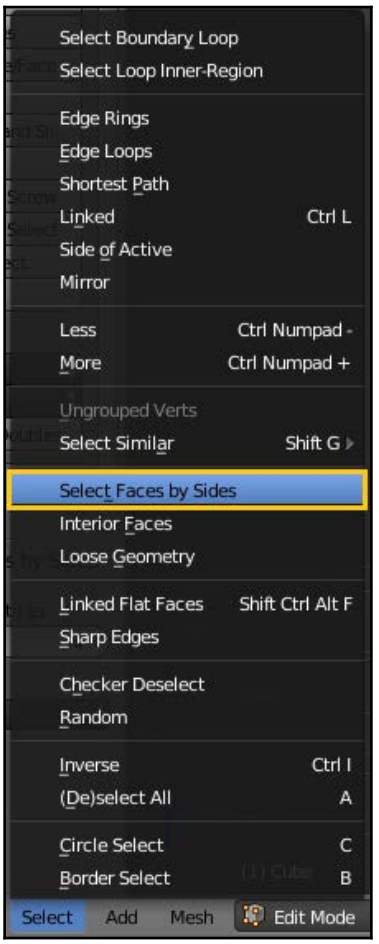


Next, I'll add an edge split modifier to the mesh. We'll use a split angle of 47 degrees or so. I wanted to make sure all the 45 degree angles were smooth, so I just added a couple of extra degrees in case there was a weird corner in the mesh somewhere.

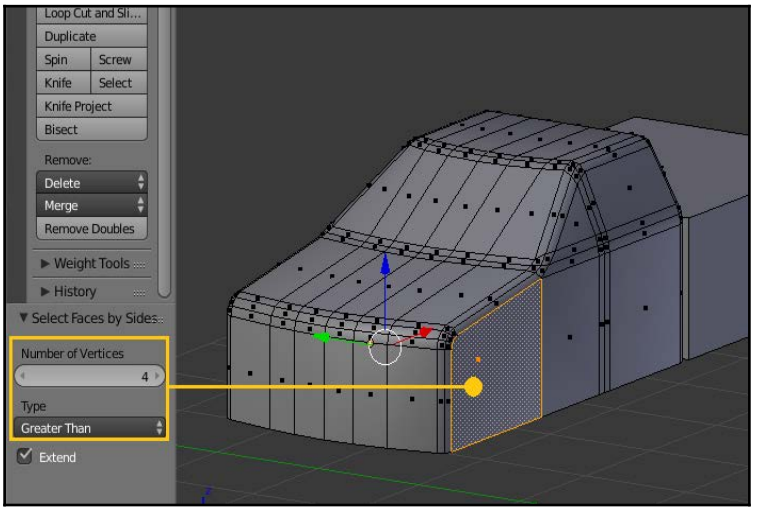


It's important to note that the **Edge Split** modifier is only temporary. Since the model is going to be exported, the smooth and flat shading will be determined by whatever game engine you end up using. It's still useful, however, as it can show us when corners are too sharp (or not sharp enough), and we can make corrections as we model.

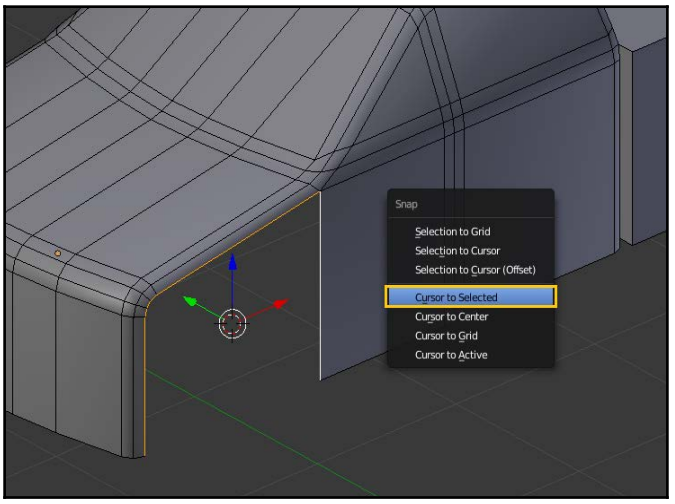
Earlier, I mentioned that we don't want to use N-Gons. However, sometimes you end up with them by accident or force of habit. At any time, you can go to your **Select** menu and pick **Select Faces By Sides:**



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:



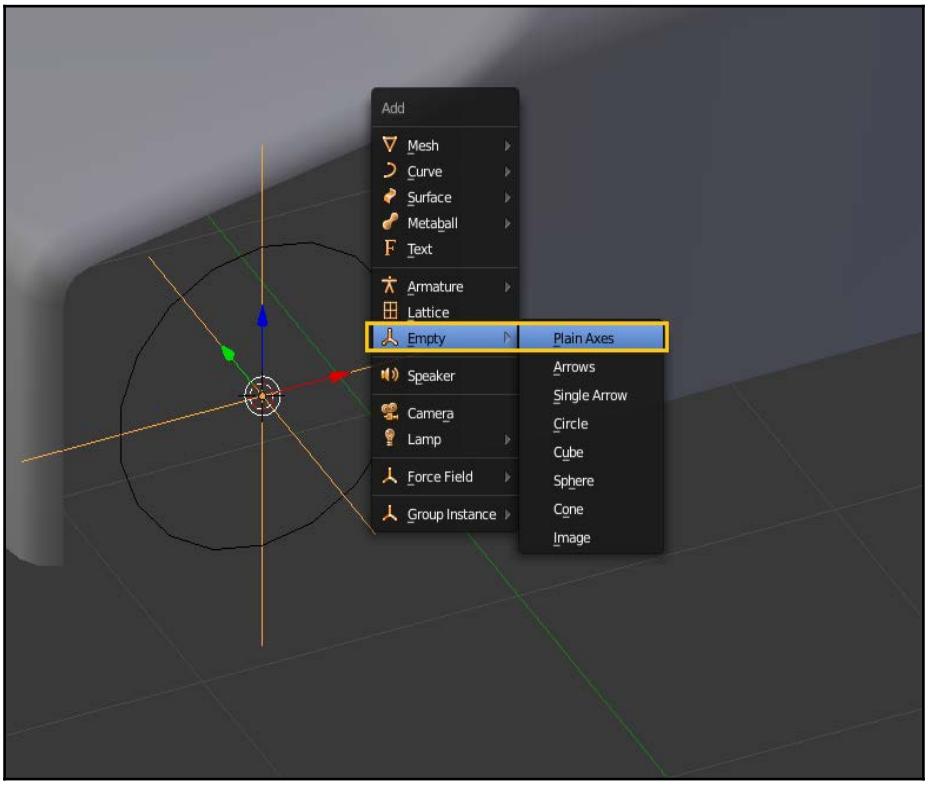
The front quarter panel would need to be redone. Of course, we need to cut out the wheel area anyway, so we'll just go ahead and delete that face. Then, we'll set our cursor to the middle of those edges:



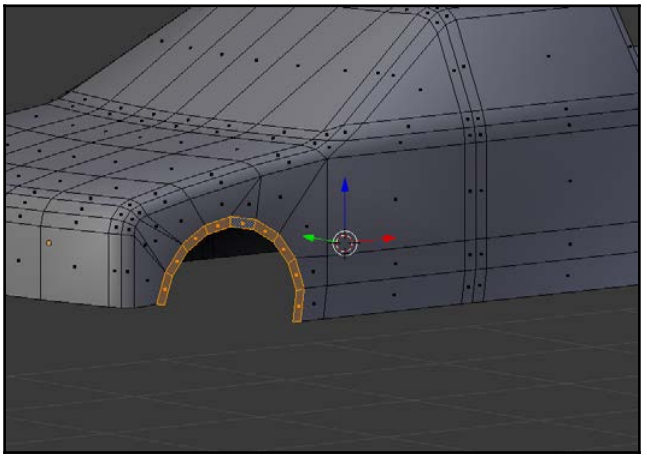
Next, we'll add two things.

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).

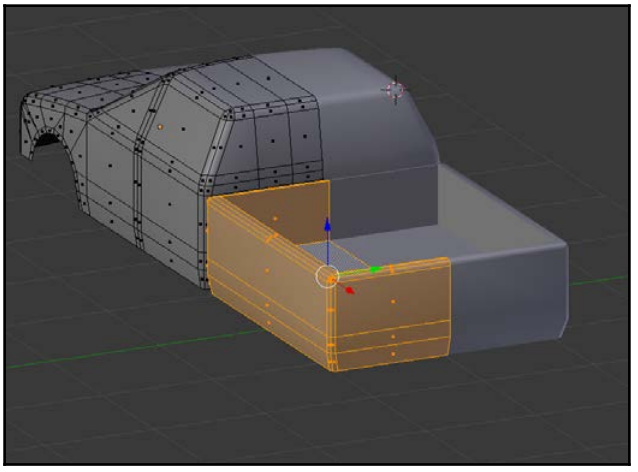


Next, we'll eliminate the parts of the circle that we don't need and build it into the mesh. I'm going to create a ring of faces around the outside of the circle, which will allow us to add a slight flare to the fenders.

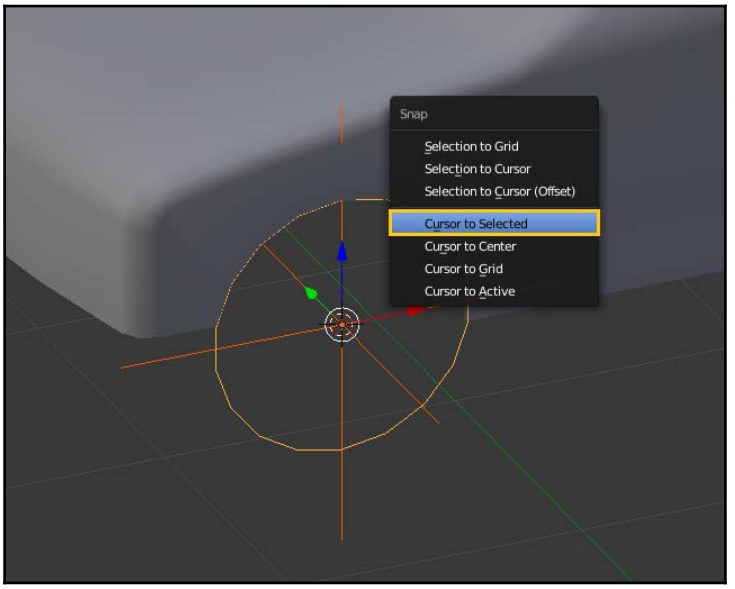


Before going any further, I'll add a mirror modifier to the model and delete half of it.

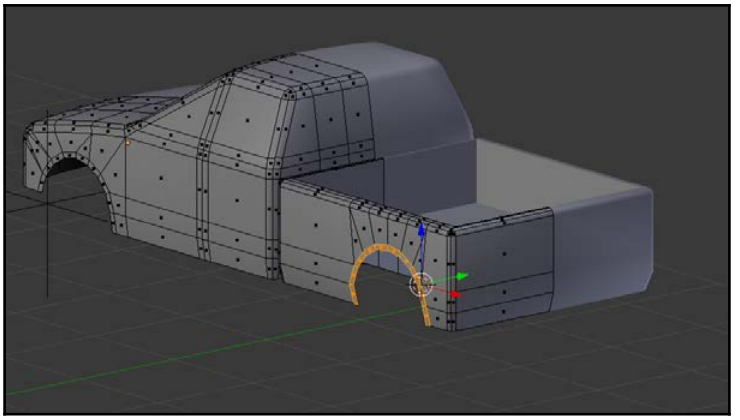
Next, we'll add a little detail to the bed of the truck:



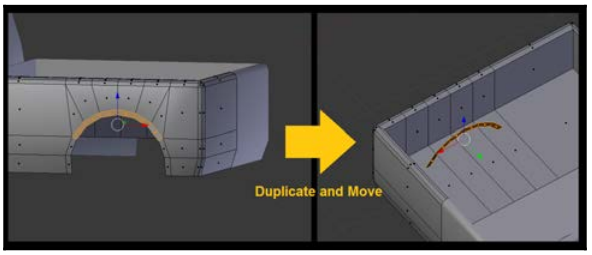
Now, I'll set the **3D Cursor** back to the **Empty** and add another circle. I can match the exact size and position of the original circle we cut in, and then move it back to create the rear wheel well.



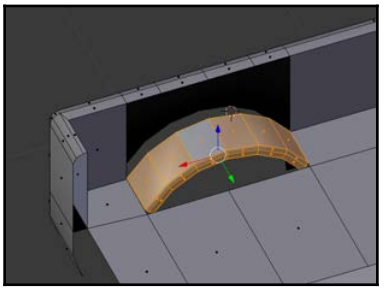
Using the same technique as before, we can create the cutout for the rear wheels:



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.



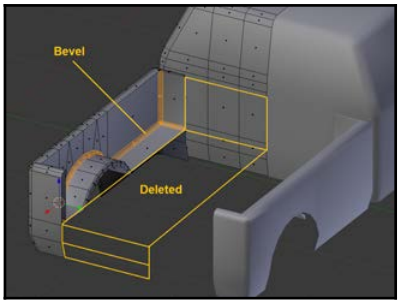
Then, we can delete the faces we don't need and build that wheel well into the mesh of the bed.



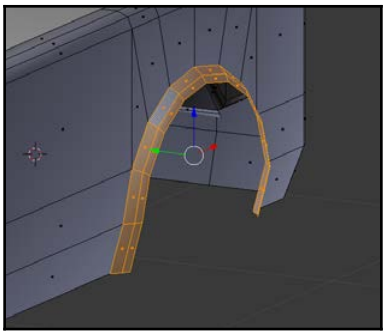
I'm just going to bevel the inside of the bed slightly.



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.



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.

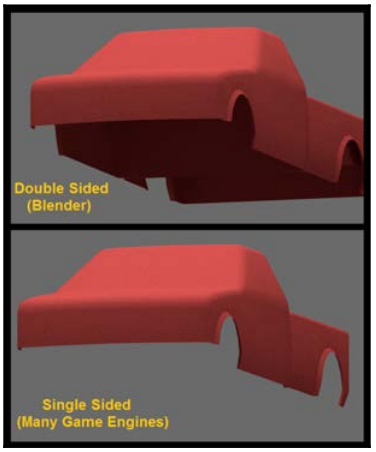


Manifold versus non-manifold meshes

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.

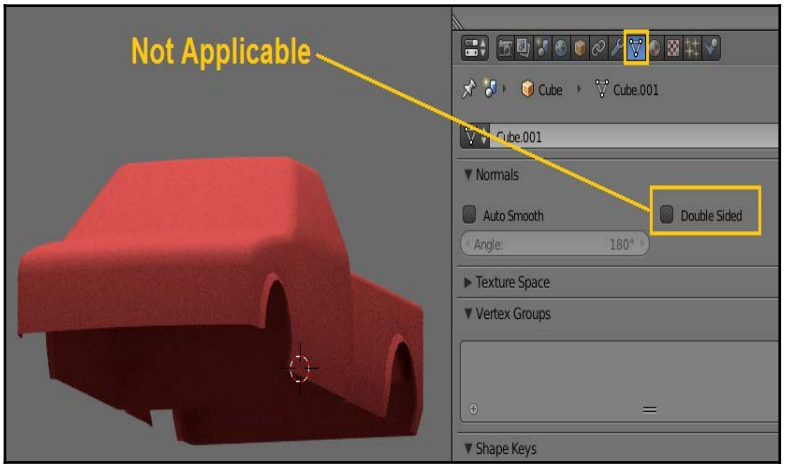
The problem is that some game engines aren't as forgiving as Blender. To increase rendering efficiency and reduce CPU/GPU workload, they simply don't render the back side of polygons. Instead, it will appear that those faces are missing from the model.



So a model that looks okay in Blender may not work in another program.

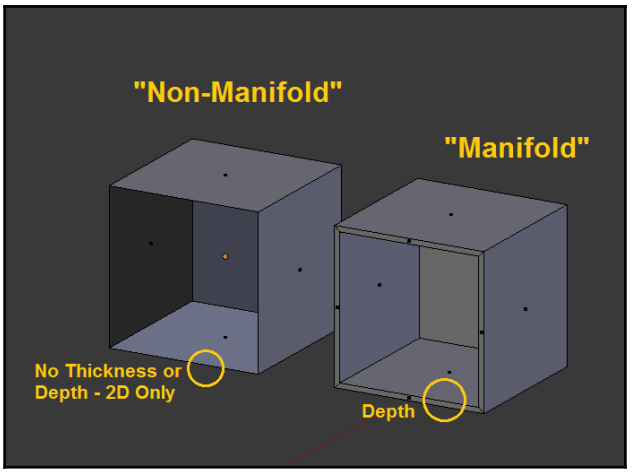


Just to clarify, this topic is only slightly related to the **Double Sided** option under the **Mesh** tab. Regardless of whether this option is checked, Blender will still render both the front and back of a polygon. The box just allows you the option to display materials and textures regardless of the direction of your normals.

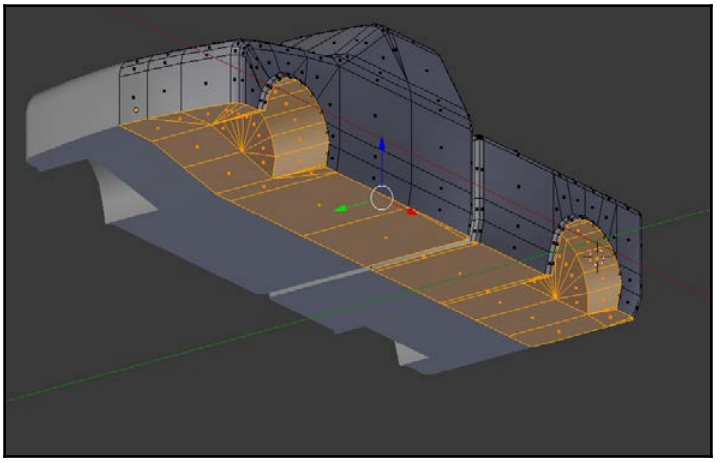


To make sure we don't have any problems with the back side of polygons, the best solution is to fill in the geometry so that we have no open or “non-manifold” meshes.

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.

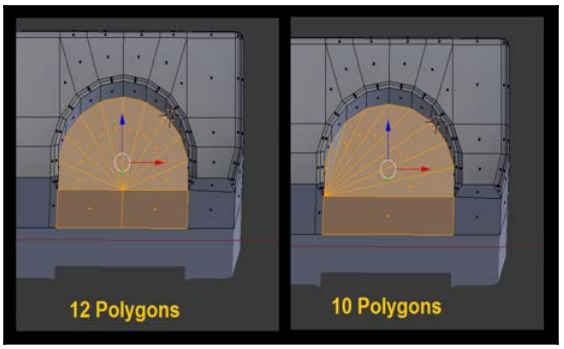


So, let's fill in the bottom of our truck to avoid this issue.



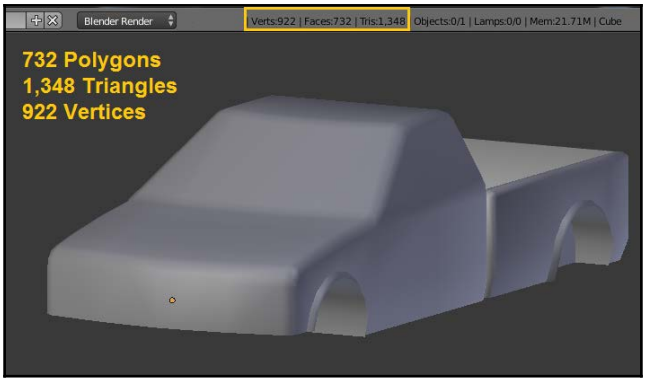
Adding details

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).

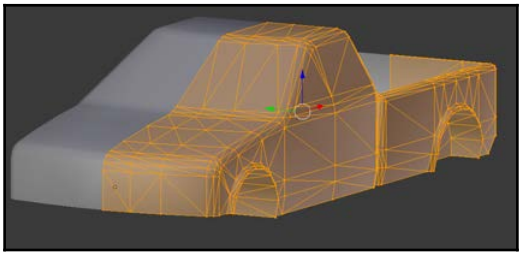
At this point, we've completed the basic shape of our truck body. You can see where we're at in terms of polygons and vertices.



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.

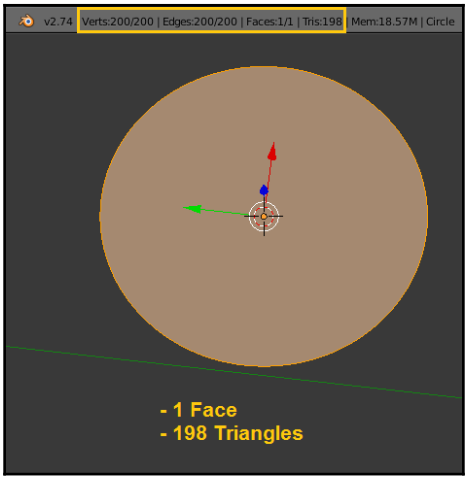
Quads and N-Gons don't really exist at render time—they're just a modeling tool to create more efficient collections of triangles.

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.

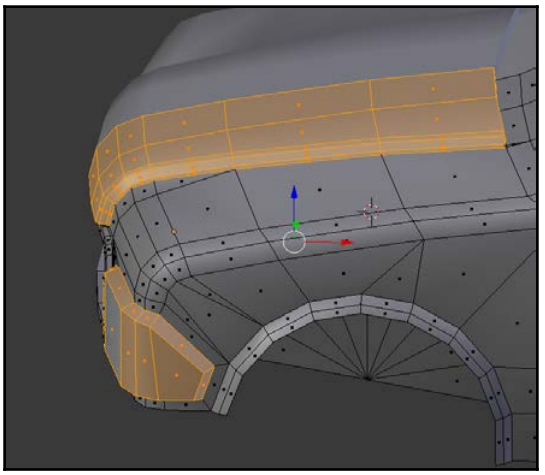
Let's say that you're building a model for a game engine that does accept N-Gons. You might add a circle with 200 sides and fill it in with one face:



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.

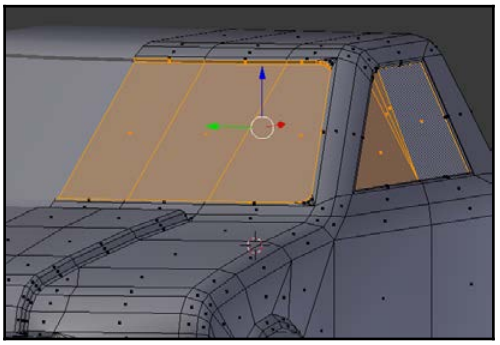
Again, this doesn't really affect us on this project, but it's something to keep in mind if you're going to be building a lot of game models for export.

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:

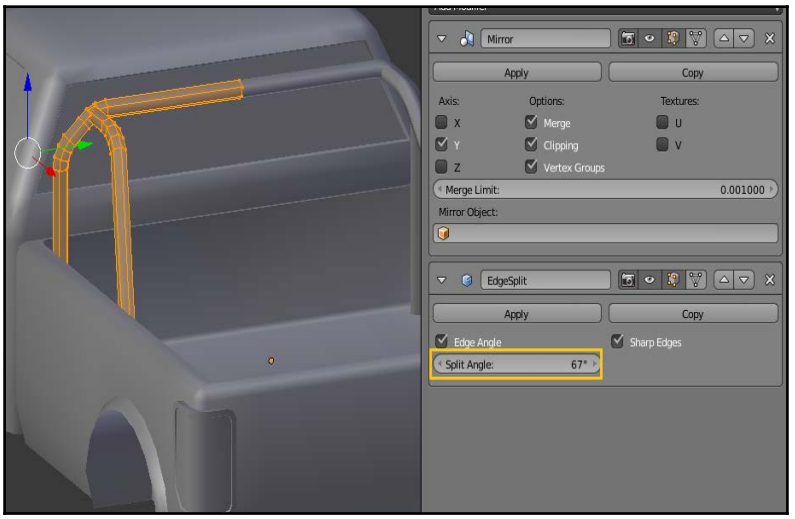


Next, I'm going to extrude the windows in just a bit. This is another area where we could actually save polygons. If we didn't extrude the windows in, we could just texture them later on. However, it doesn't cost us very many polygons to do it this way, and we can later add a "glass" material to just 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).

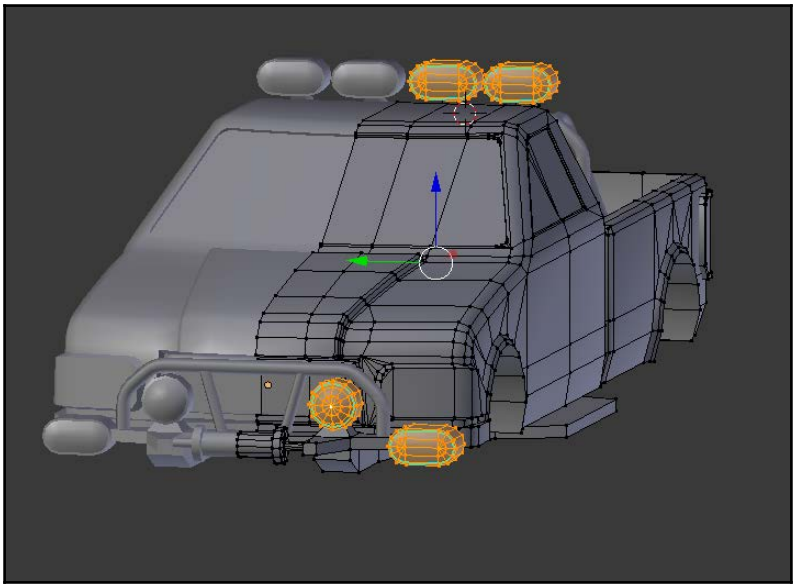
As you will quickly discover if you build game models, there are pros and cons to just about everything. Sometimes, we just need to make a choice.



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.



I'm also going to add in a few extra lights for our truck:



Next, I'll create some wheels (as a separate object):

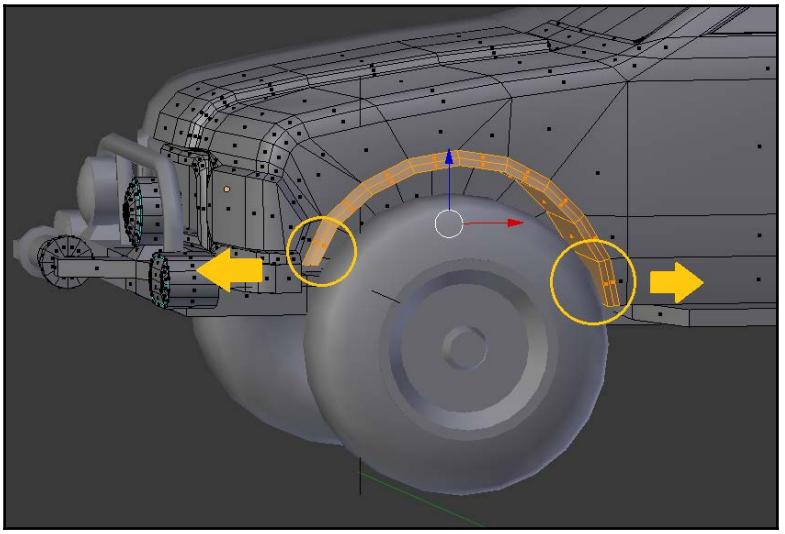
Remember, since the truck has four wheels, every polygon you add is being multiplied by a factor of four. You can add as much detail to the wheels as you'd like, but just keep an eye on how much geometry you're adding.

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.

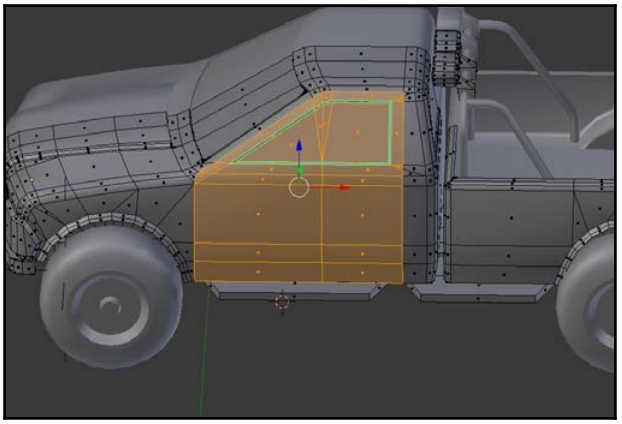
On the other hand, it will add polygons to your model and require additional work when you set up your game. You'll have to make sure that the various parts all fit and move correctly.

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:

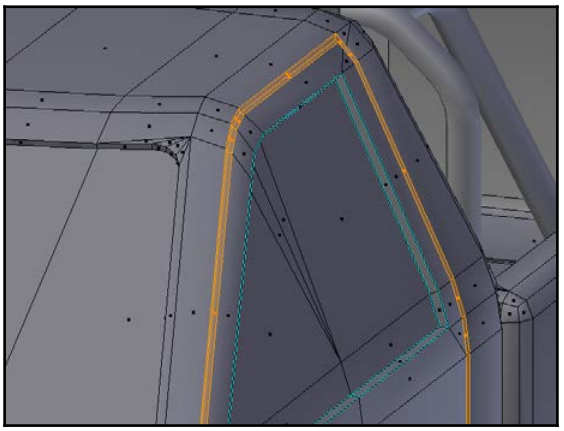


The next thing I'll do is select the faces that make up the door area.



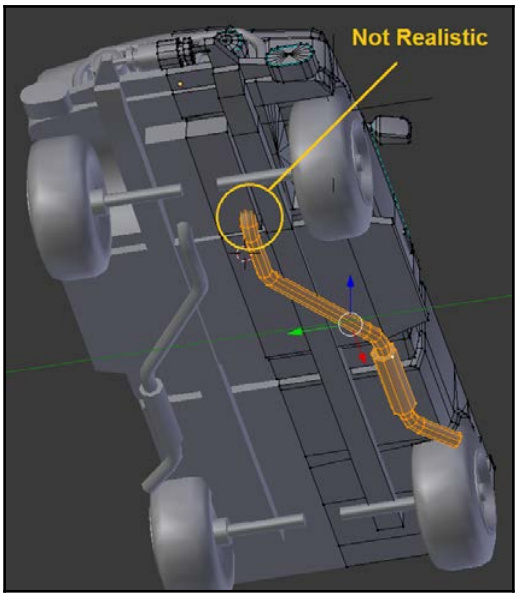
Using the **Inset** tool, we can create a slight panel line where the door connects to the body. This is another case where we don't technically need to do that (it uses polygons), but there may be a benefit.

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.



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.

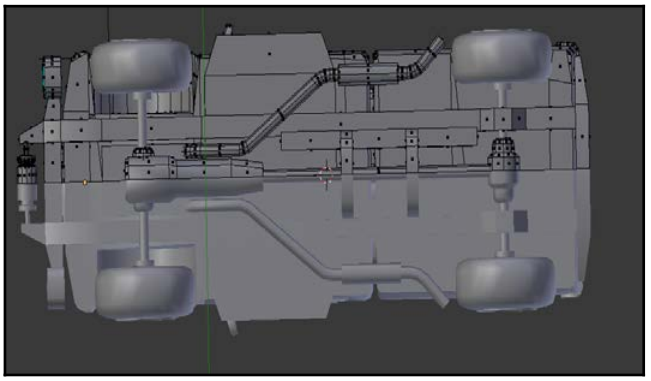
For instance, I've added an exhaust pipe that runs along the underside of the truck, but it just disappears into the engine area.



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.

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.

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.

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.

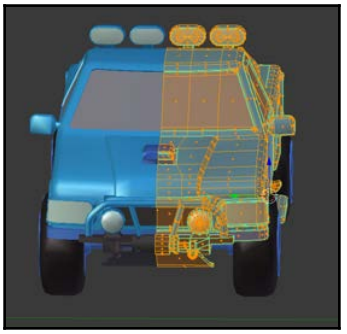
After we have our image-based texture map, we can tweak it in an external image editor and add details.

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.

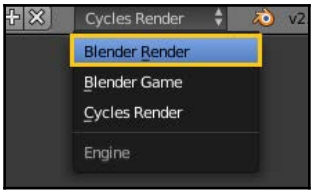
So, let's get started.

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:

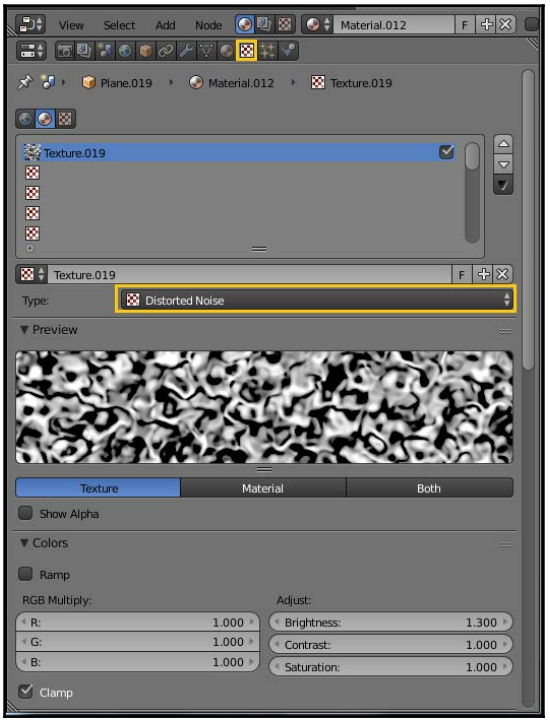


The primary material, of course, will be the truck's paint. We'll use Blender Internal materials to create a stylized paint scheme. First, switch to the internal rendering engine:



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.

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:



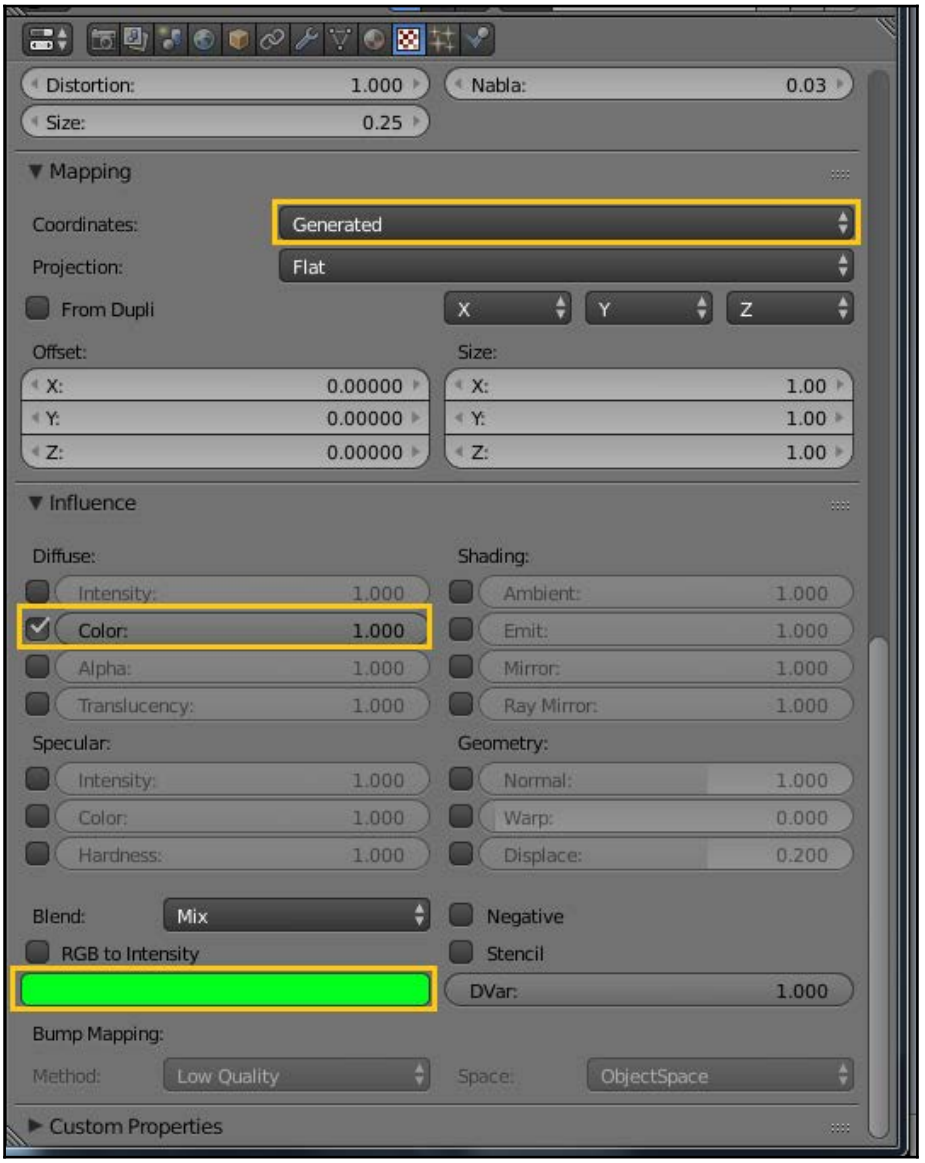
You can see that there are plenty of settings to adjust, but we'll leave those alone for now. Below the pattern itself, you'll see options for **Mapping** and **Influence**.

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.

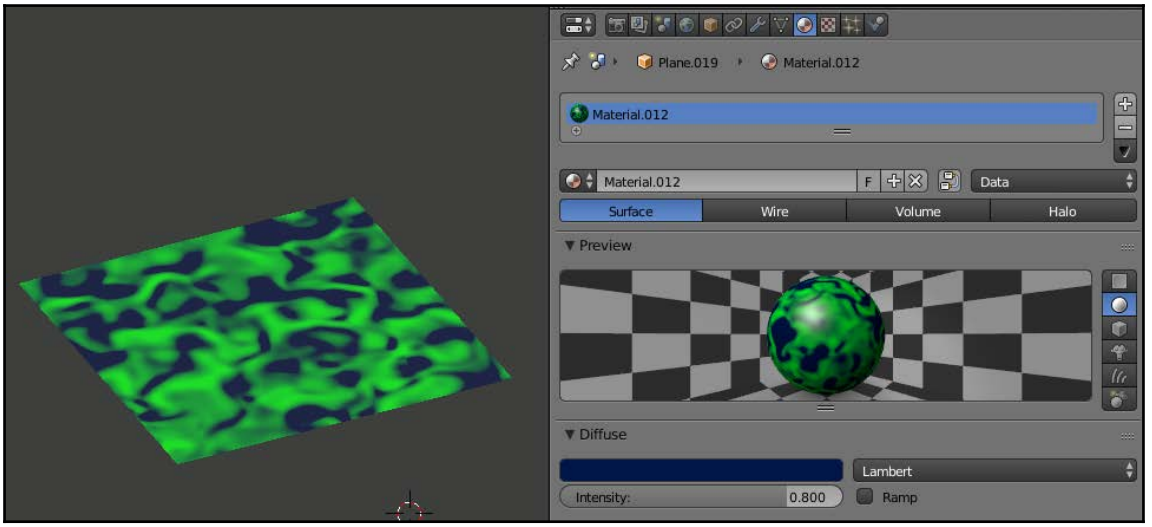


Textures that use mathematical patterns rather than image maps are generally referred to as procedural textures.

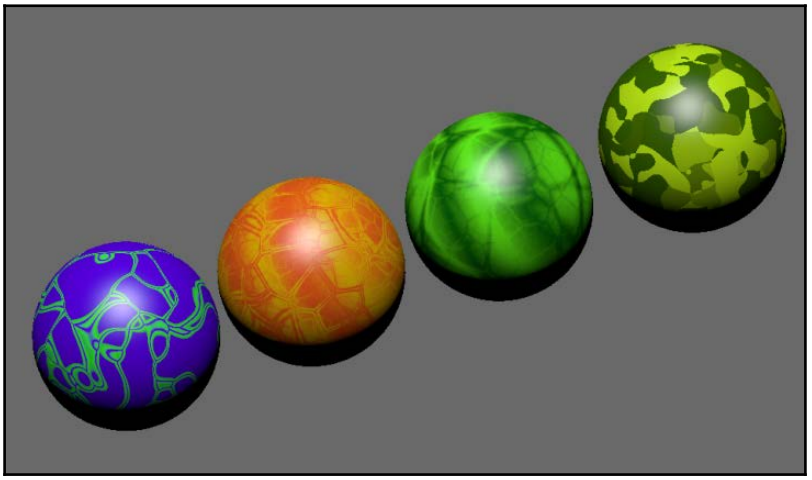
After setting our mapping coordinates, we'll the select the **Color** option under **Influence**. This means that the texture will affect the color of our material. You can set the color to whatever you'd like. In this case, I've selected green.



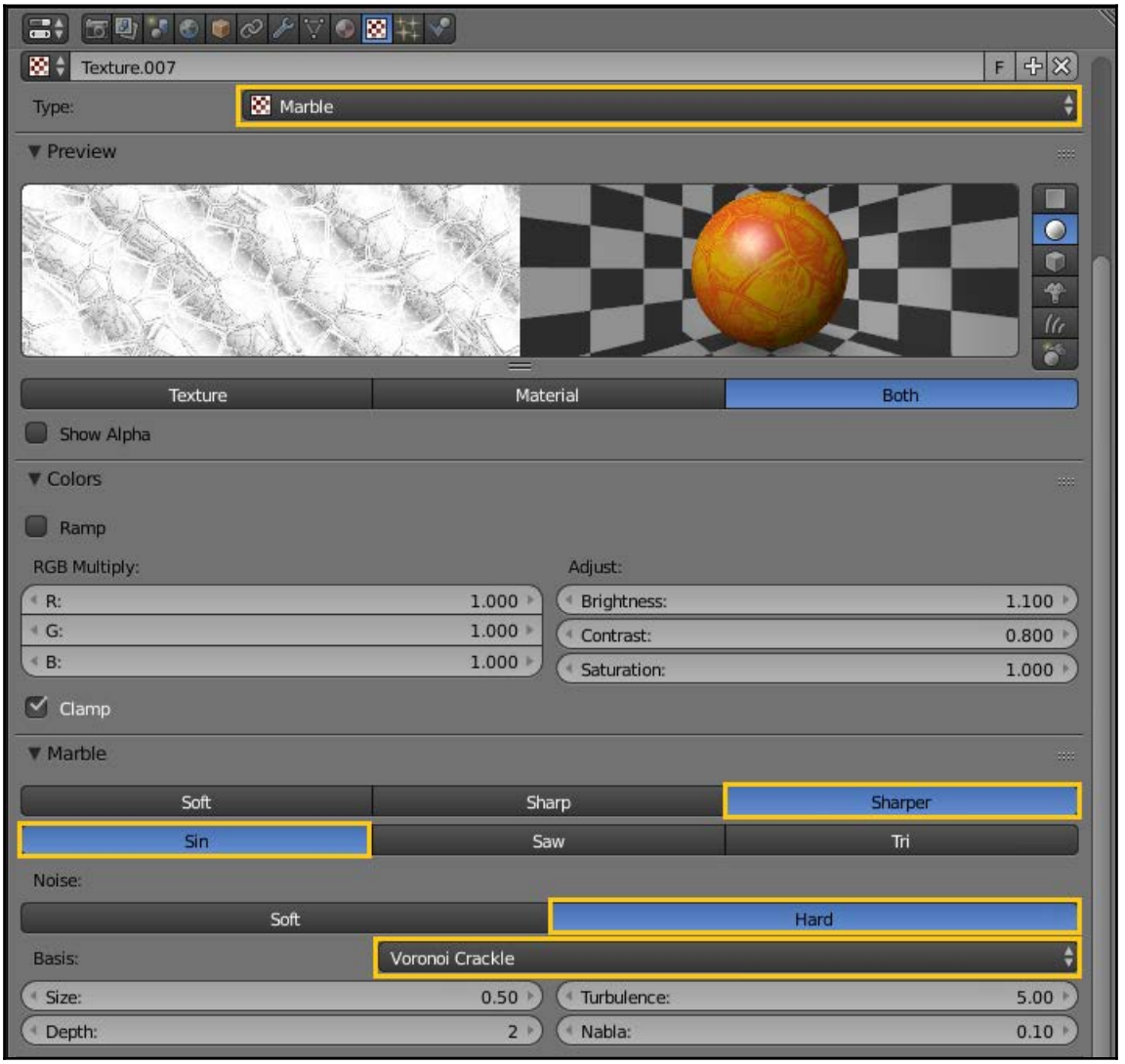
With a blue base material, here's how that texture looks:



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:



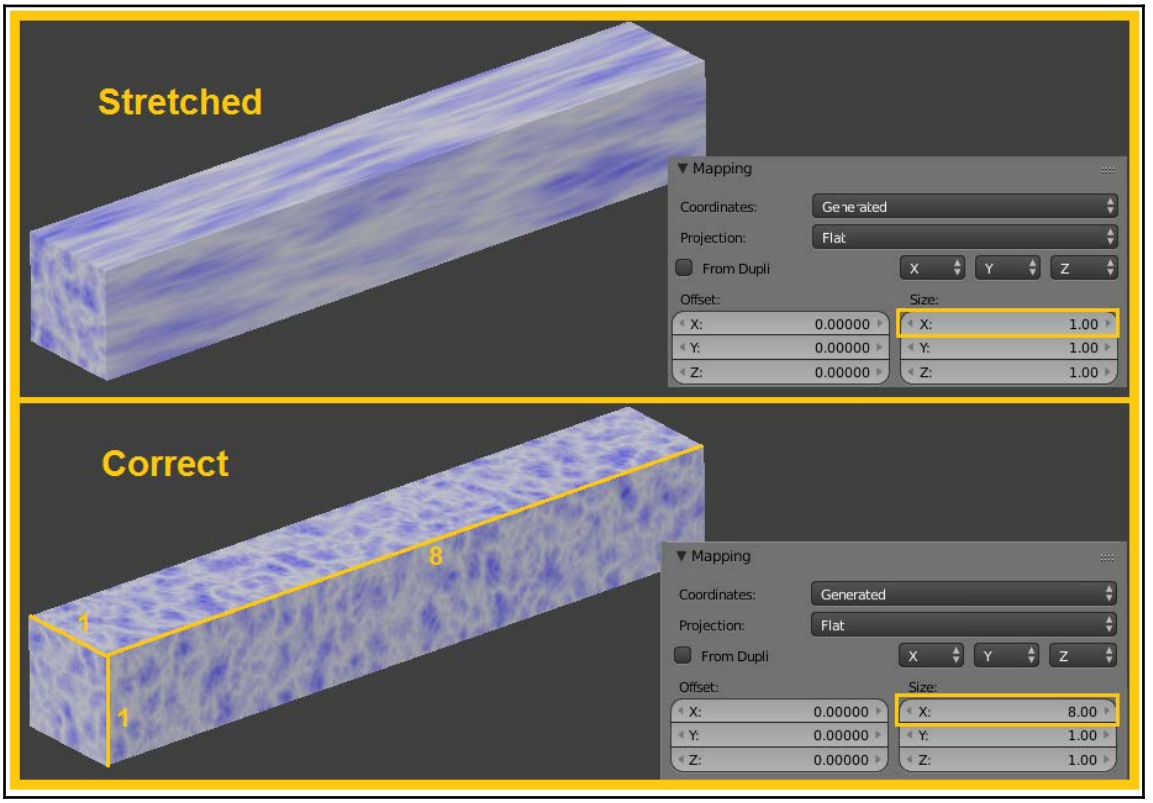
For instance, we can use a **Marble** texture and select **Voronoi Crackle** as our **Noise** Basis:



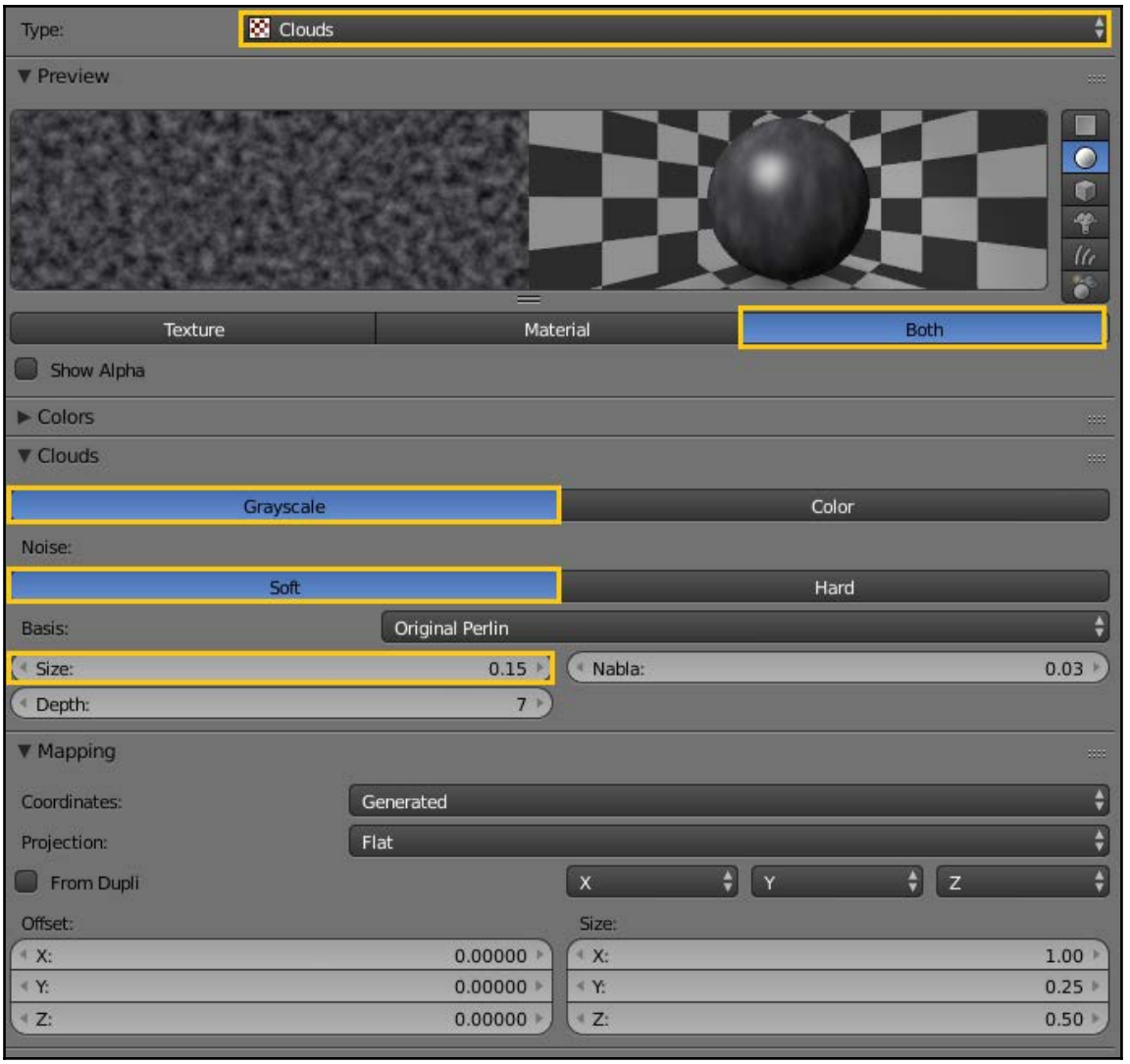
After playing with some settings, I think this will look pretty good for the custom paint job on our truck. Of course, you can try various textures (or combinations of textures), until you find one that looks good to you:



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:



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:



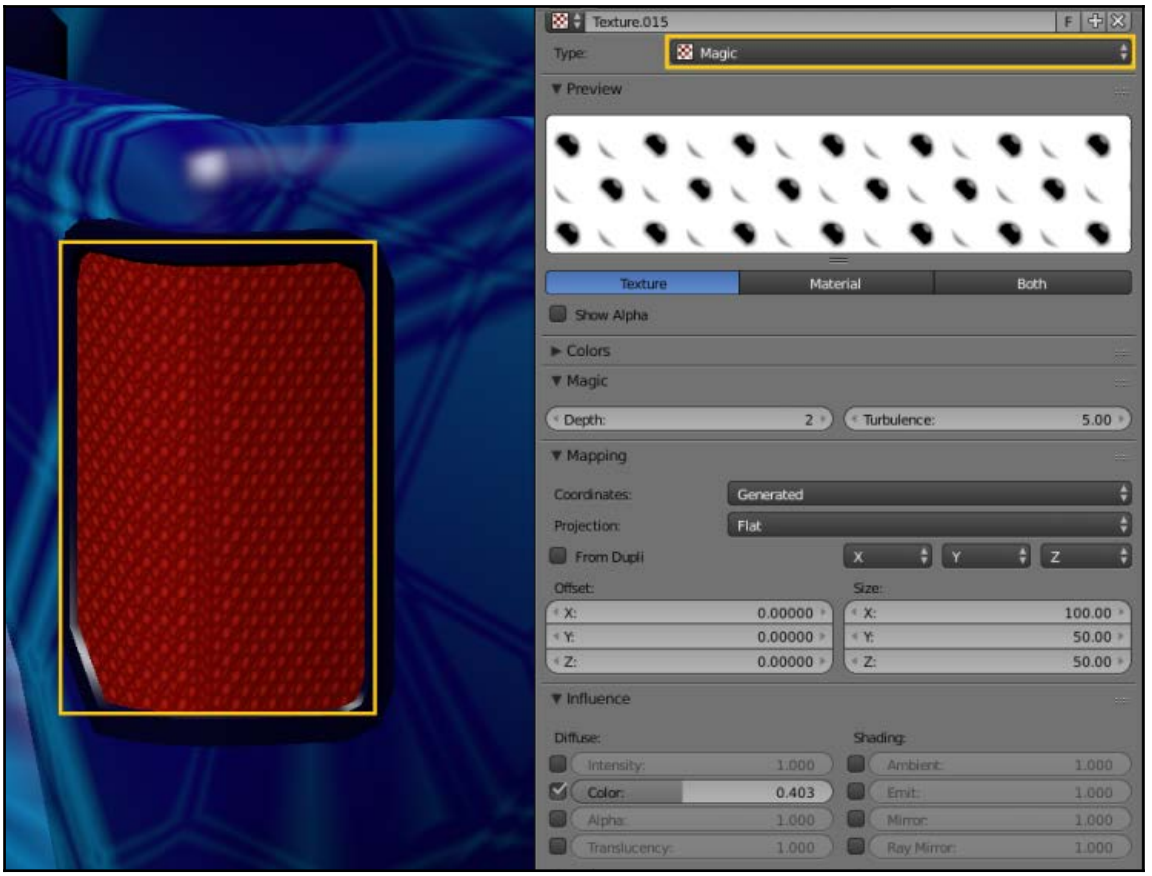
A simple cloud pattern is probably good enough for now. As you can see on the bottom of the truck, it doesn't look particularly realistic. However, it does look better than a flat color, and it's seamless—no gaps or edges:



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:



We'll repeat this for the various materials. For the taillight, we can use a **Magic** pattern to add a bit of detail:



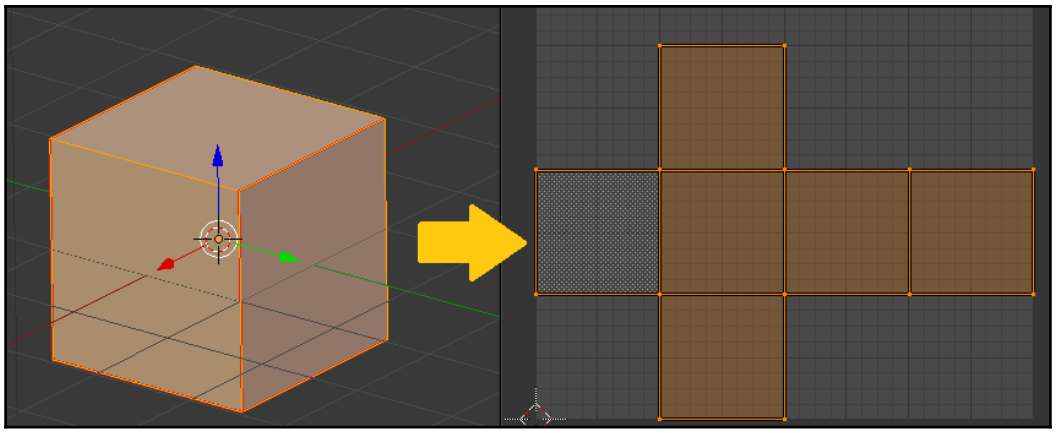
One thing I'll do at this point is delete some duplicate objects. For instance, each half of the truck has three sets of ovular (oval-shaped) lights. Since they're all going to look the same, we can just continue with texturing one light and then make duplicates later. As you'll see in just a minute, that will make things more efficient:



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.

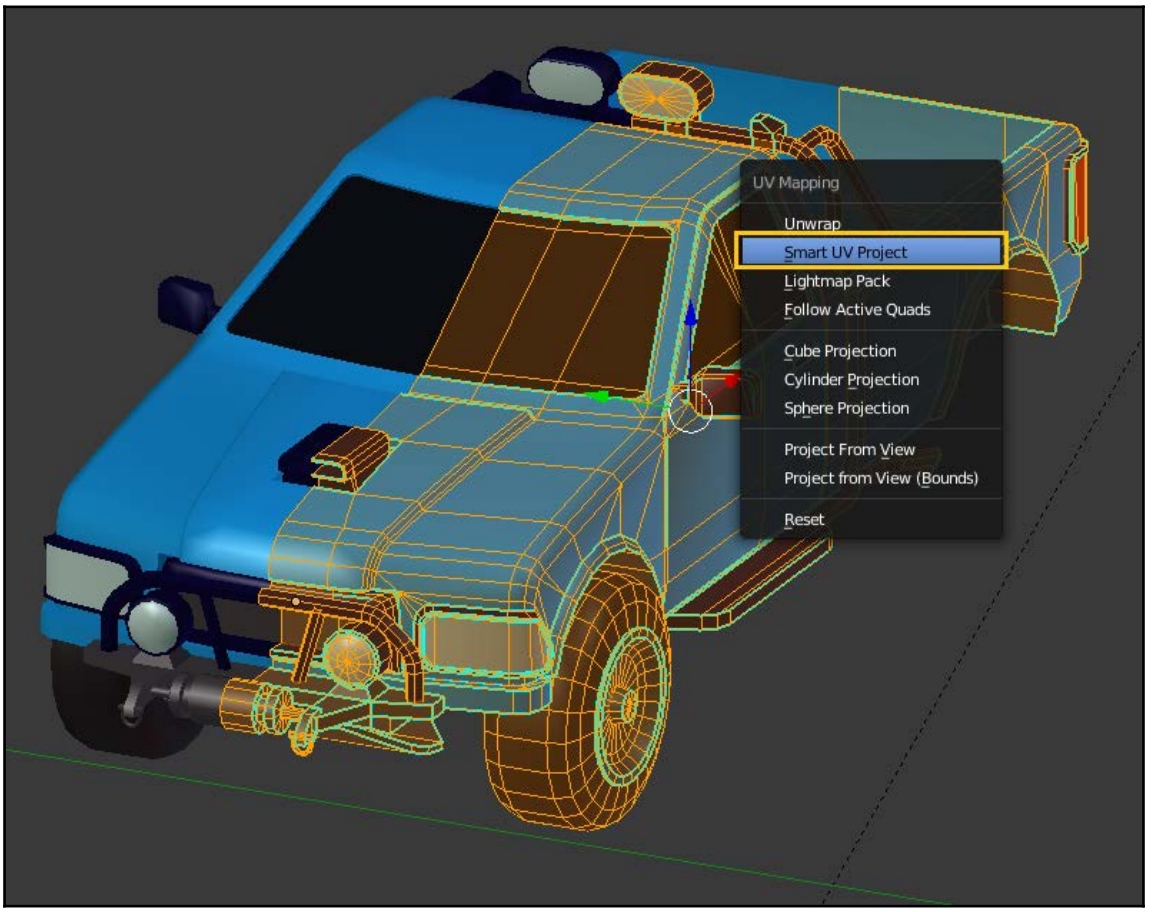
If you're not familiar with the idea of UV unwrapping, it's really pretty simple. Our model exists in 3D space, but it will be using a 2D image texture (eventually). We need to tell Blender how to apply that 2D image to our 3D object. When we talk about unwrapping, you can think of it as a paper model. We need to cut and *unfold* the model so that it lies flat:





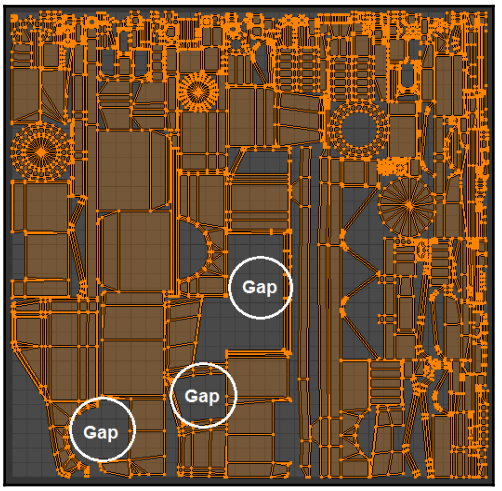
UV unwrapping is a non-destructive process. You don't change or damage your 3D model at all. The traditional coordinates (*X*, *Y*, and *Z*) still define the location of your vertices in 3D space. A second set of coordinates (*U* and *V*) will define how those vertices are mapped to a 2D image. This is why it's called a **UV Map**.

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.

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:



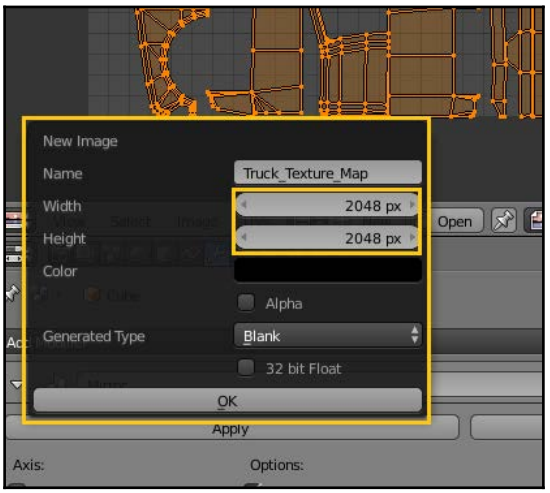
The second reason that this isn't ideal is that it's not intuitive. In other words, you can't look at the UV map and immediately identify the hood panel or the tailgate. If you're going to be painting decals on later, it's quite helpful to be able to look at your UV map and understand what the various 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.

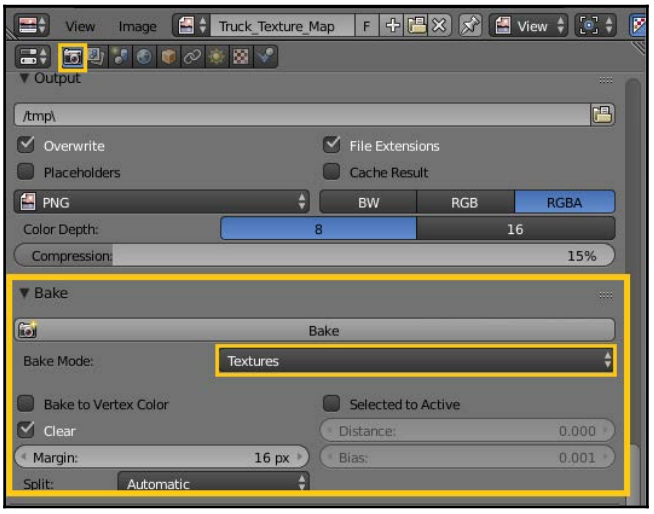
Let's create a new image in the image editor. I'll call it `Truck_Texture_Map.png` (but of course, you can name it anything you want). I'll make it *2048*x*2048* pixels:



An image that's *2048*x*2048* 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:

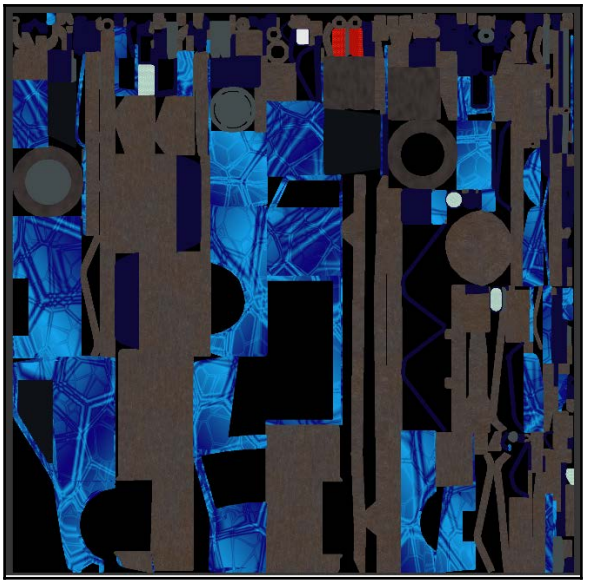


There are plenty of options here, but we'll keep it simple for now. Select **Textures** from your **Bake** mode drop-down menu. Then press **Bake**.

In your **UV/Image Editor**, you will now see that Blender has baked the procedural textures we created to an image.

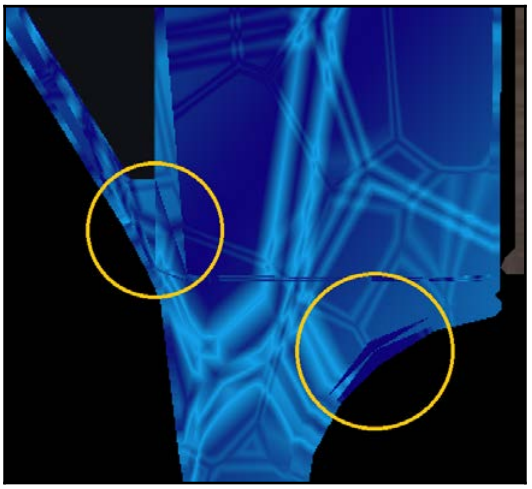
This is useful for many reasons and necessary if you plan to export models to another format. The procedural texture that we'd been using for the paint, for example, was based on a mathematical pattern (in this case, the **Marble** option). However, that particular pattern (and the generated coordinates 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:

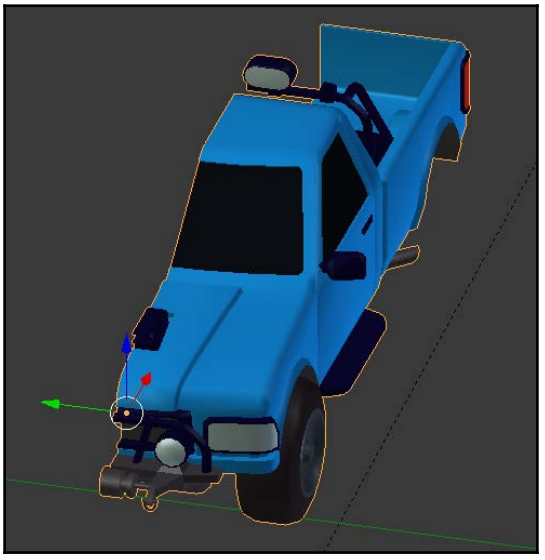


However, we do have a slight problem with the way we baked it. Because we used a mirror modifier on the model, Blender tried to bake both sides of the truck to the same 2D texture space. However, the paint pattern is not symmetrical (it's different from one side to another). There's no way to bake both halves of the truck to the same 2D space.

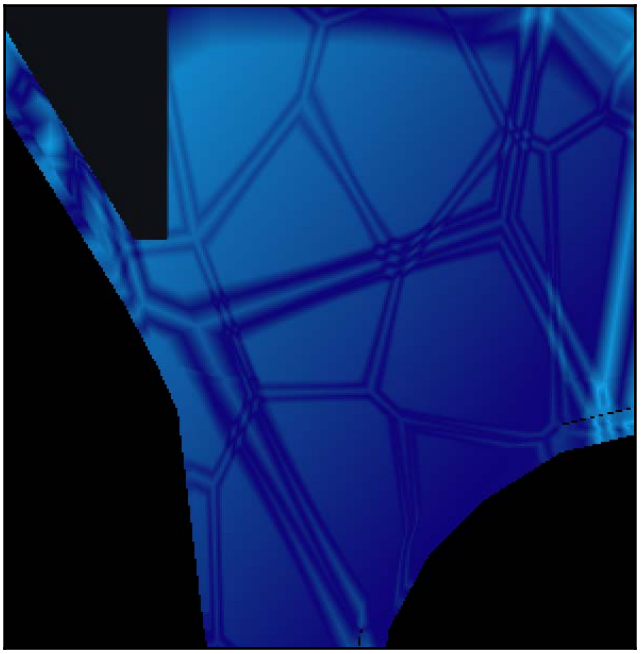
Therefore, you will see a number of overlaps and problems in the image as Blender tried to combine both sides:



Of course, one way to fix this would be to turn off the **Mirror** modifier on our truck:



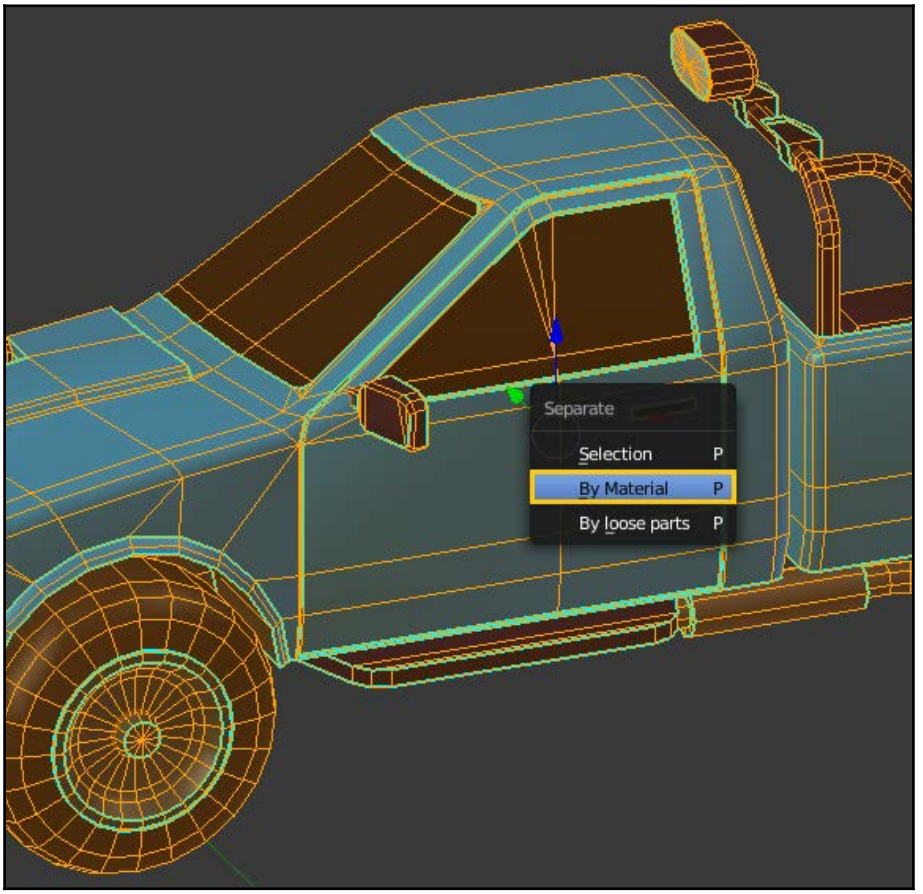
That will work, in the sense that you won't have odd artifacts during the bake:



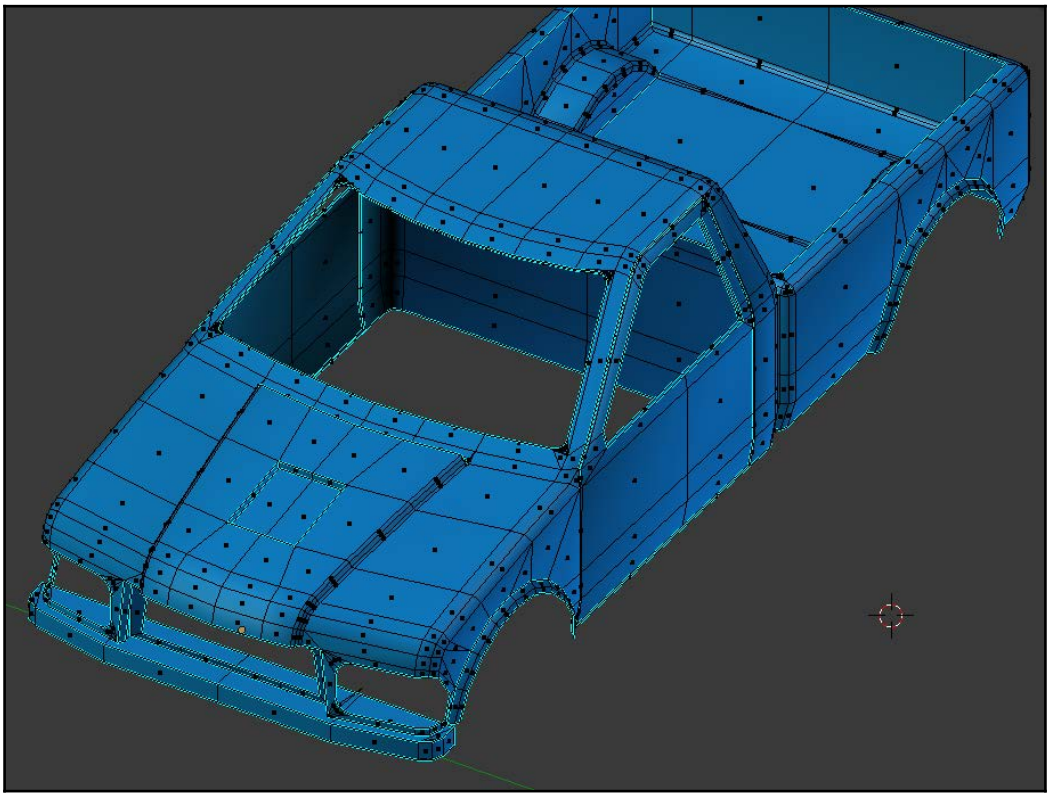
Of course, it also means that both sides of the truck will be identical, which wasn't the way we created it originally. It doesn't necessarily look bad, but it's just something to be aware of. You can get away with it when it comes to a paint pattern, but it will pose a problem in other areas.

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.

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.



For those parts that we don't want to mirror (like the main body), go ahead and apply the mirror modifier:

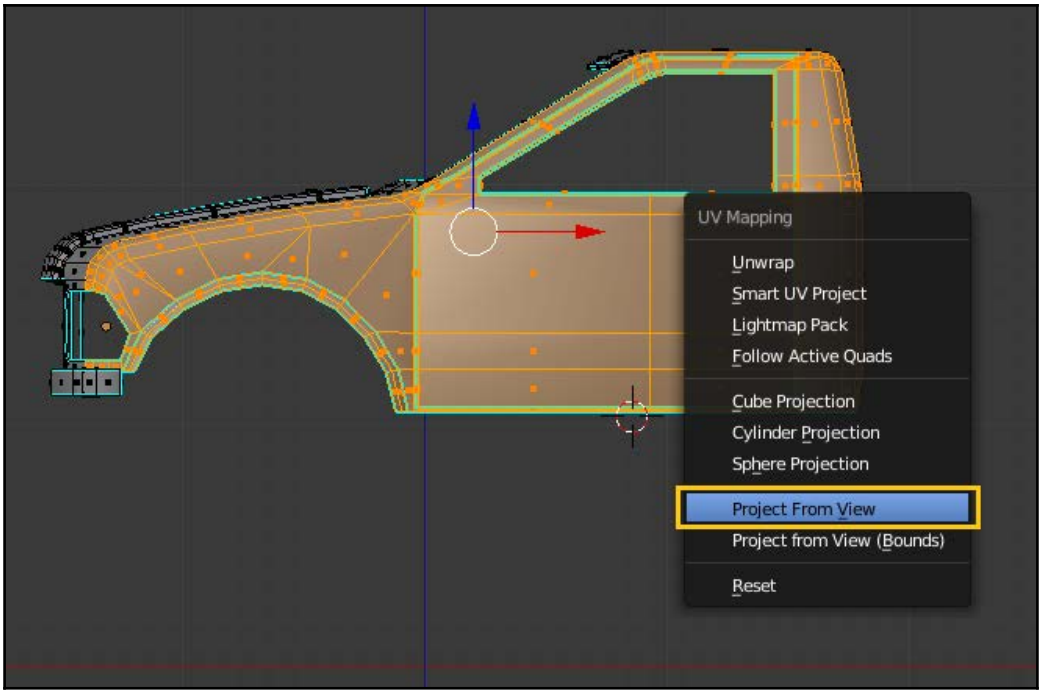


Now, we'll go through and unwrap these UVs.

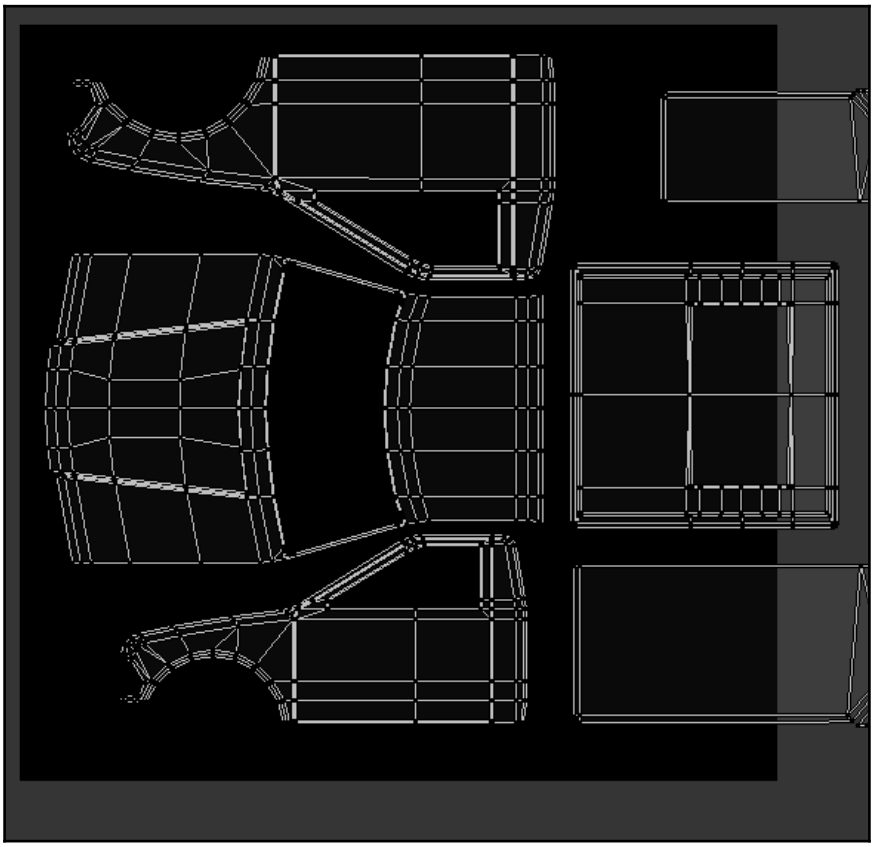
From the side of the cab, I'll pick all of the polygons that are directly facing us and press the *U* key again. This time, however, I'll pick **Project From View**. The selected faces will now be unwrapped exactly as you are looking at them.



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.

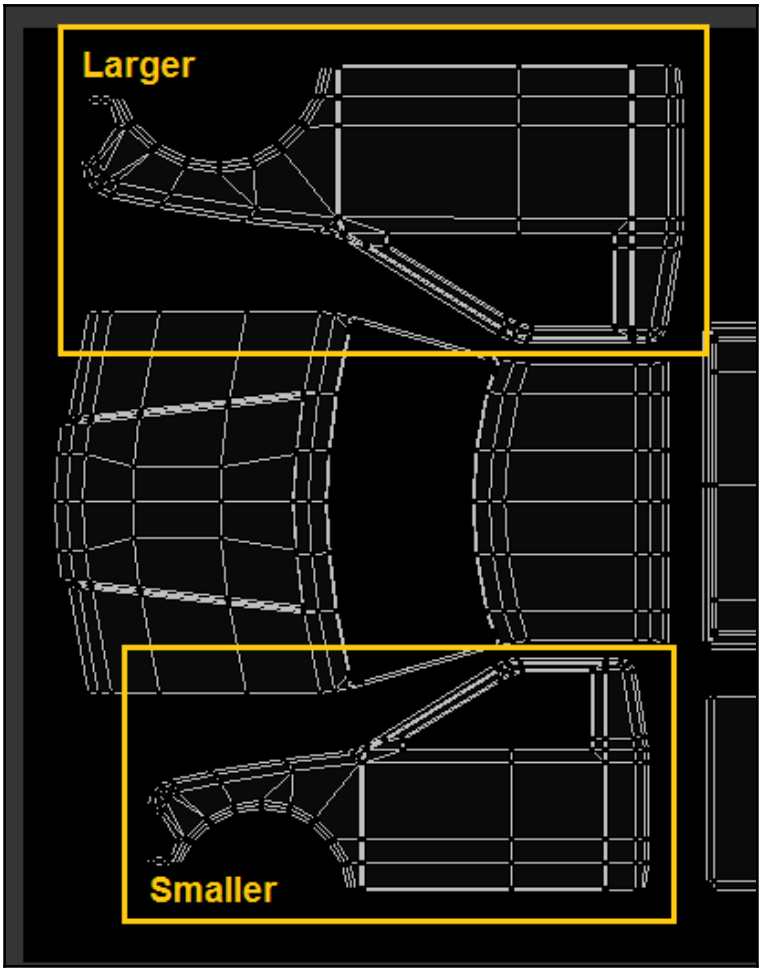


You can repeat this process for the other parts of the body and move the resulting UVs around until they form a rational pattern:

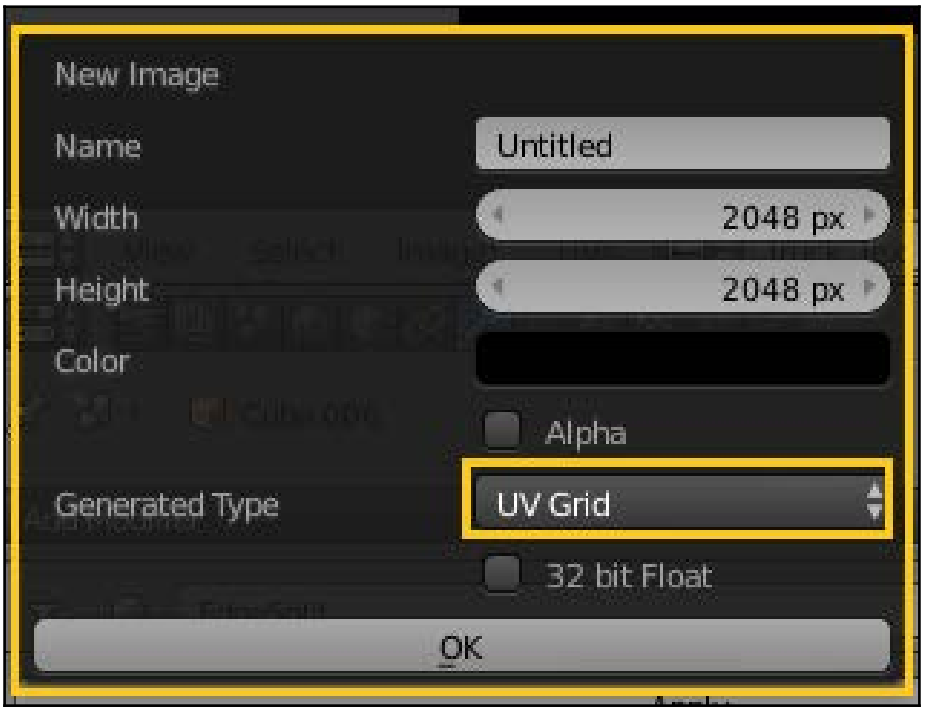


This result still leaves a lot of unused texture space, but it is far more intuitive than the **Smart UV Project** option we used originally. You can look at these UVs and immediately identify the various body panels. That will be extremely helpful when painting textures on later.

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:



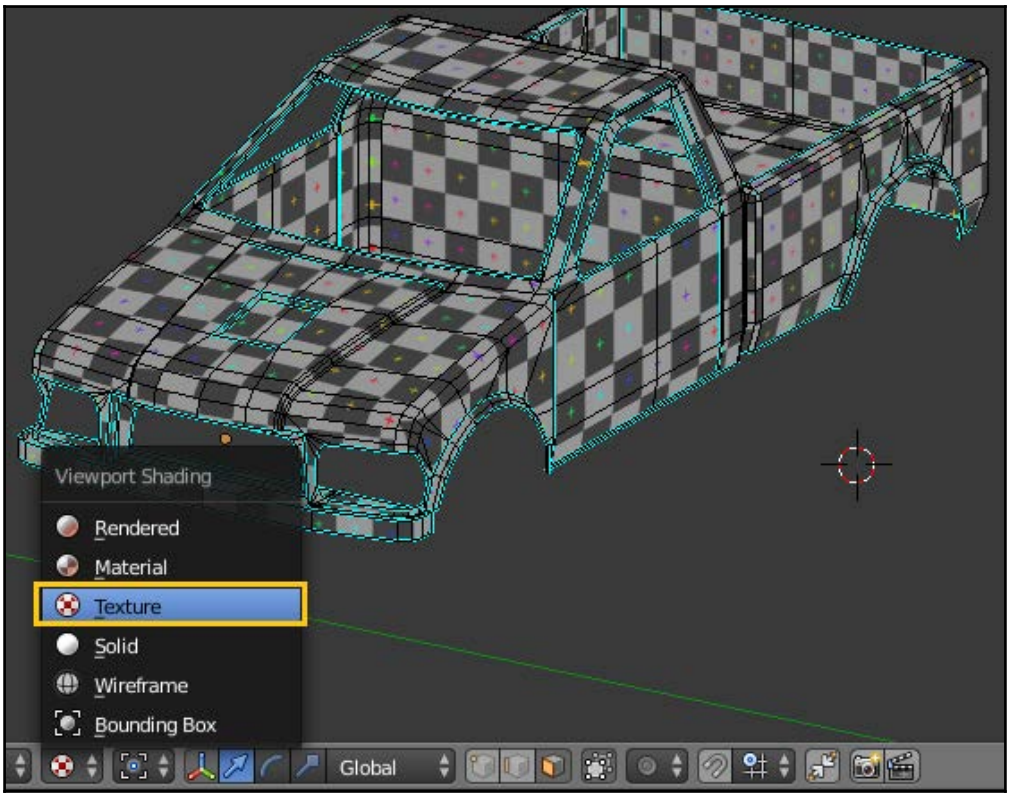
One way to fix this is to add a new image in our **UV/Image Editor**. Under the **Generated Type** option, we'll pick **UV Grid**:



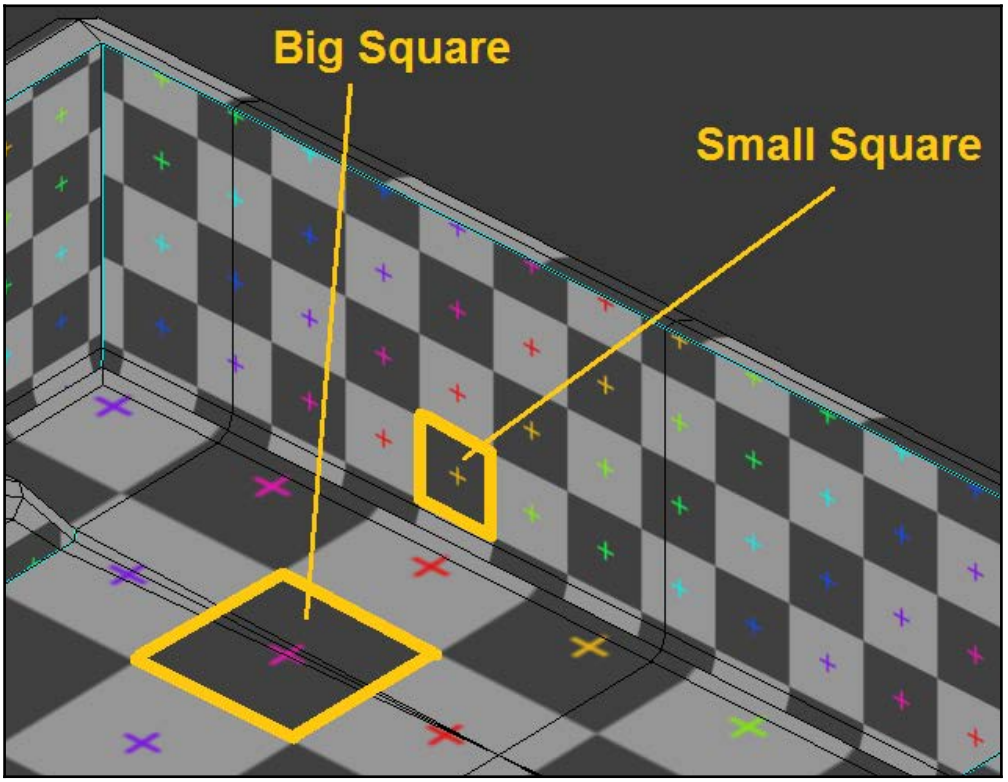
Now, you can switch your **Viewport Shading** to **Texture** and take a look at the UV grid. The purpose of the UV grid is to help you check your UV maps (to make sure they're not stretched or improperly sized):



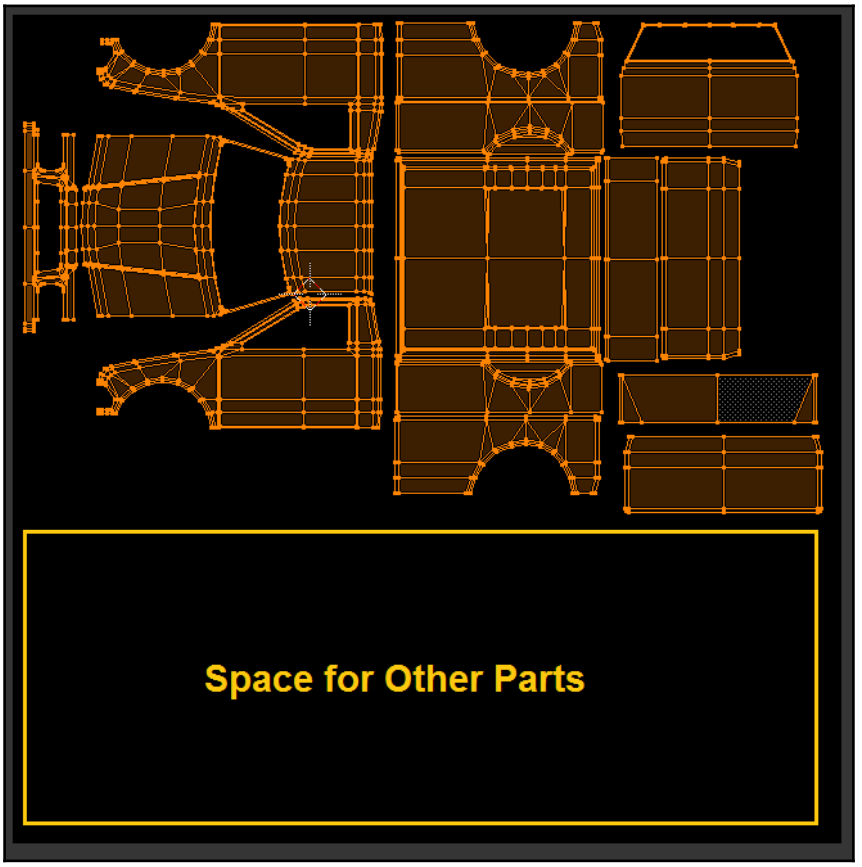
You can also use the **Area Stretch** option to do this.



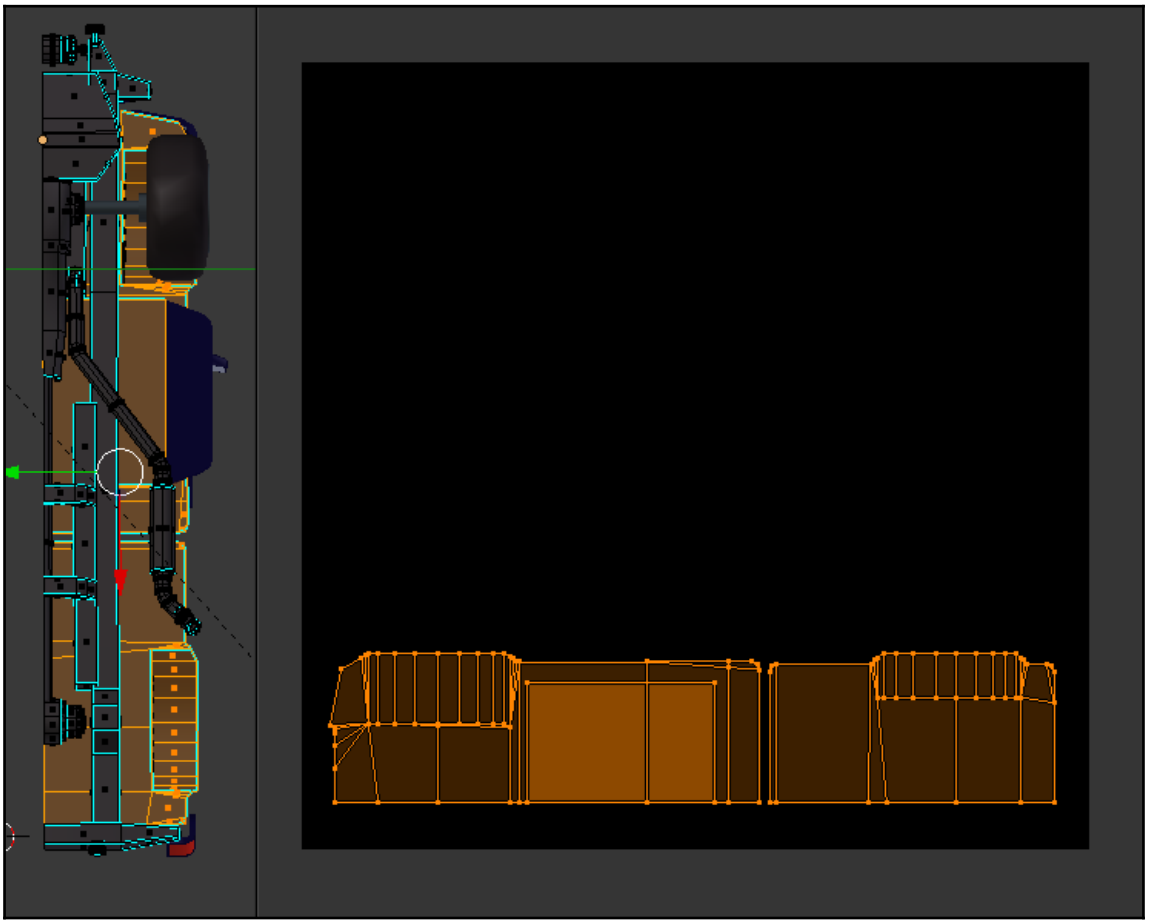
For instance, when I look at the inside of the tailgate and the bed of the truck, I can see that I have a problem. The tailgate UV map is too large by comparison, resulting in squares that are too small:



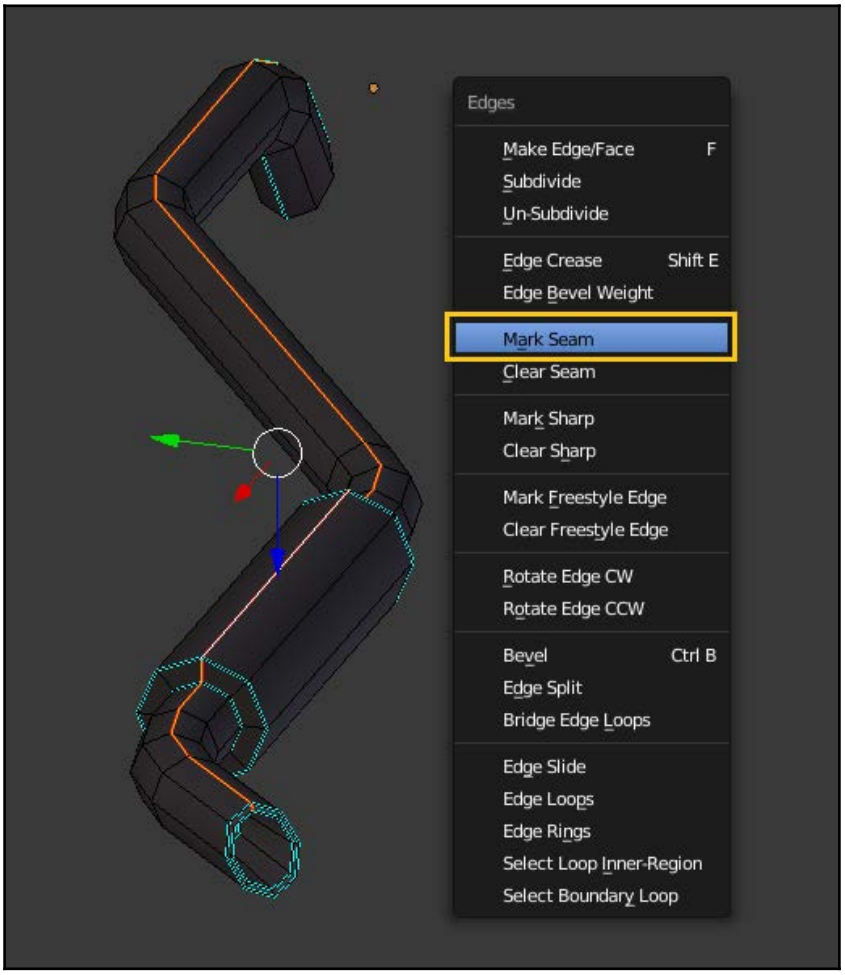
Using this method, you can adjust the size of your UV maps. You can also use the Average Island Scale option to do this automatically (*Ctrl* + *A* in UV editor). When you've got them all the correct size (or close enough), you can arrange them more logically. Make sure that you leave space for the rest of the truck inside of your square image window:



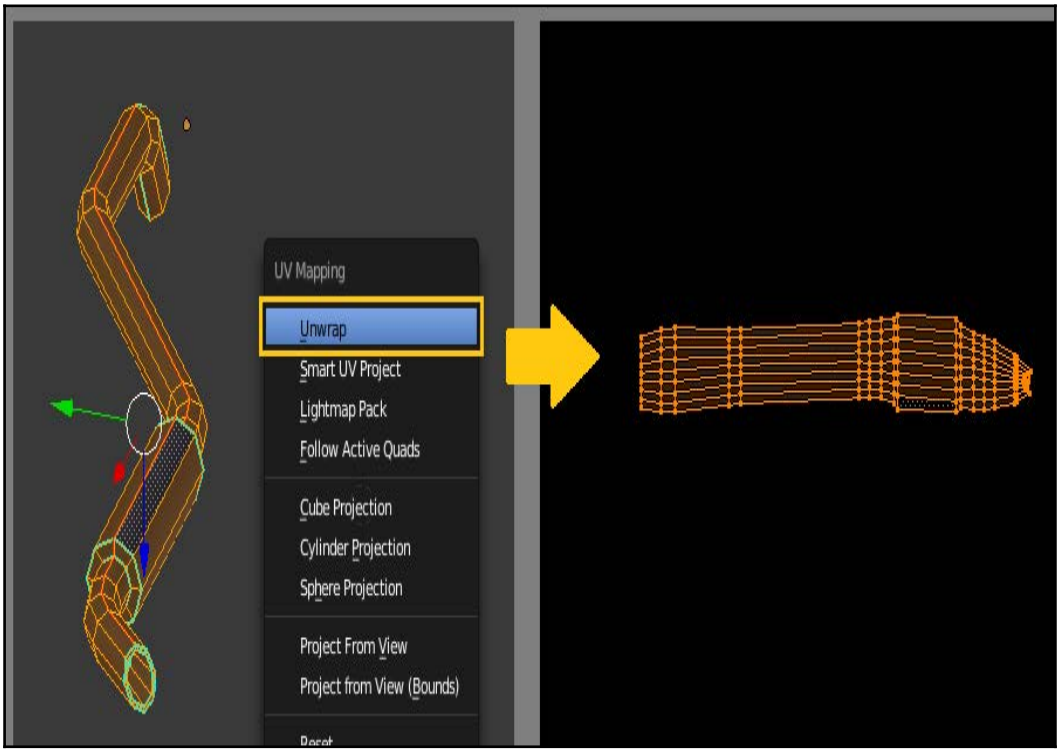
Next, we'll move on and unwrap the frame and mechanical parts at the bottom of the truck. First, we'll unwrap the flat part that makes up the actual bottom of the truck.



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**:



Now that you've marked your seam, you can unwrap with the first option (**Unwrap**):



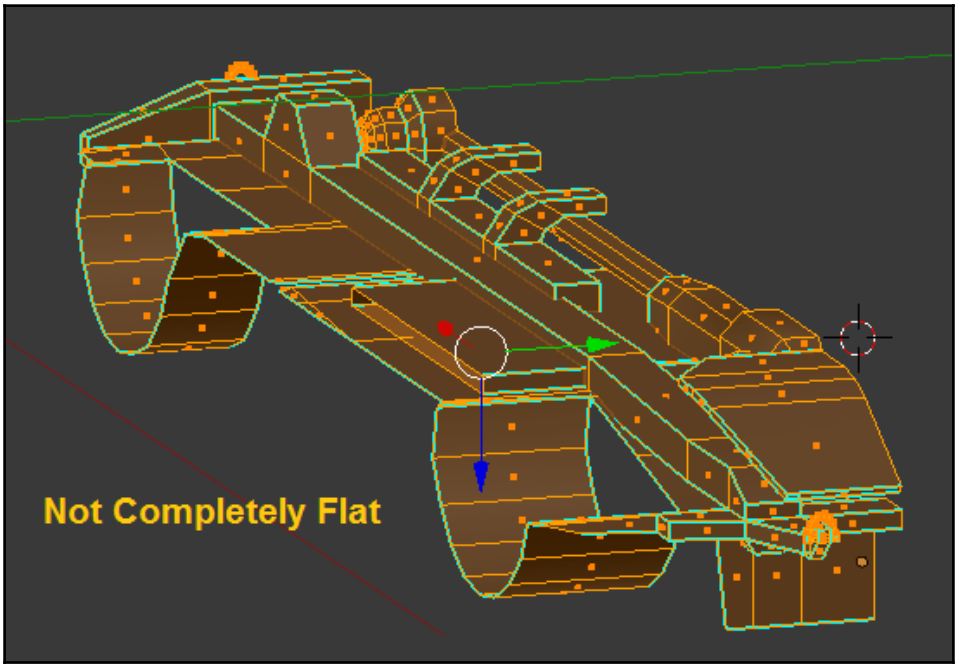
The resulting UV map may look a bit strange, but it does make sense when you think about it. Because the different parts of the exhaust have different diameters, the UVs will have be a little taller/wider in some places to make up for that.

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.

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:

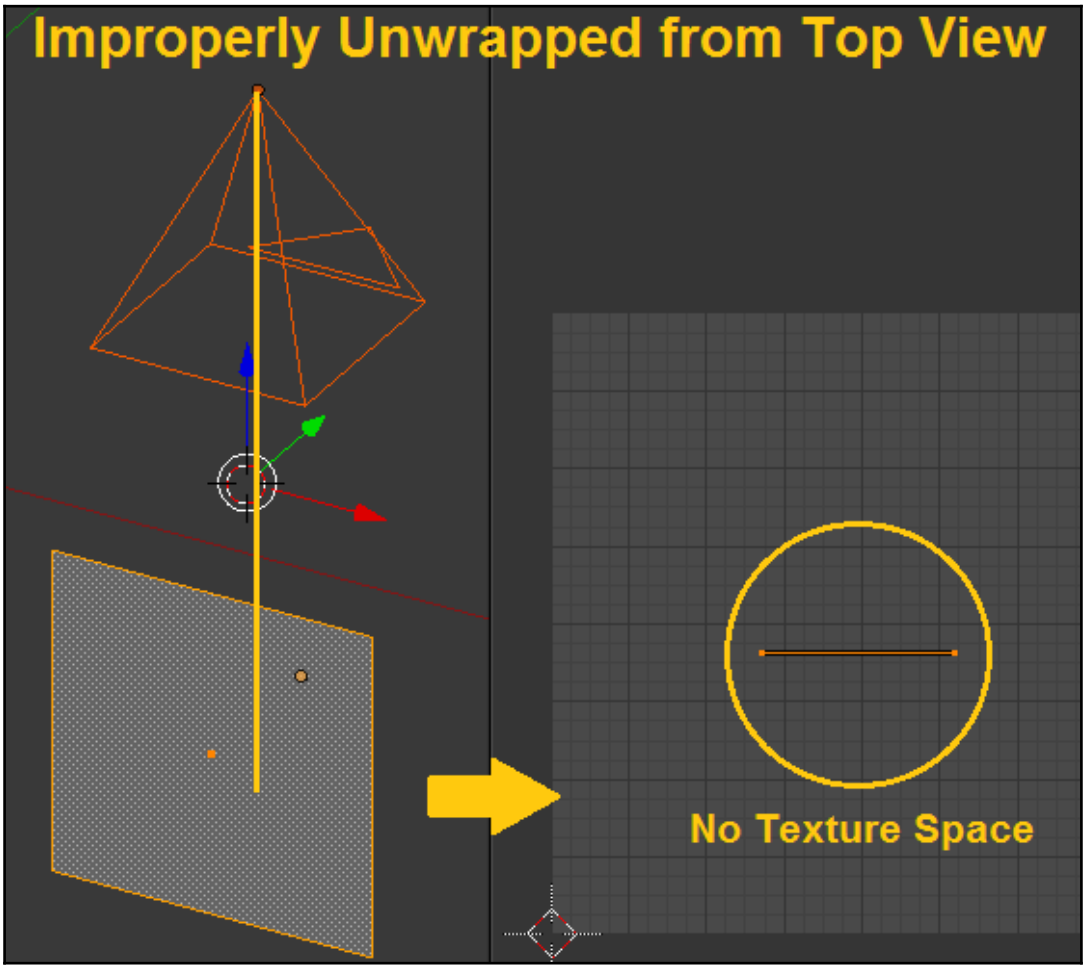


What I'm about to demonstrate would not be considered a good technique. It's a quick way to get the job done (and therefore worth knowing about), but I advise you to use it sparingly.

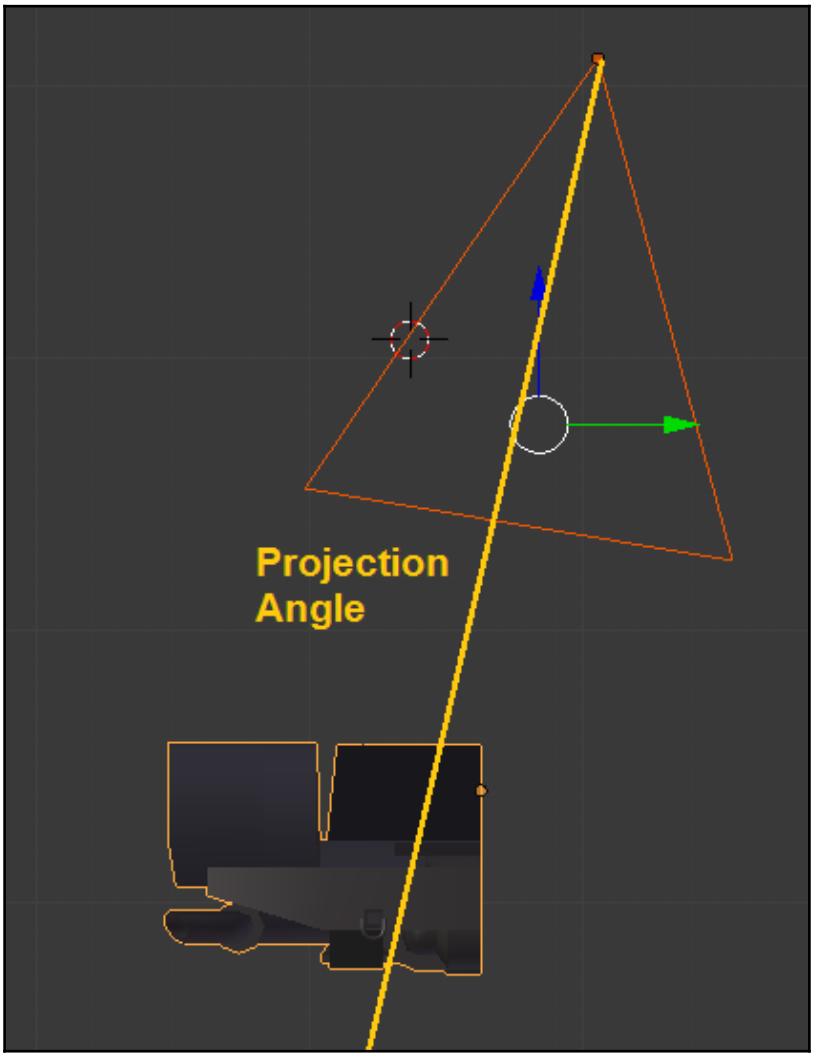


You can see that bottom of the truck is *almost* flat. In other words, it looks like it could mostly be unwrapped from the bottom view. That's very tempting, as it would save a lot of work. However, you can see that there are a few little parts that would need to be unwrapped from the side view.

Naturally, unwrapping a part from the wrong view will result in having no texture space available for it:



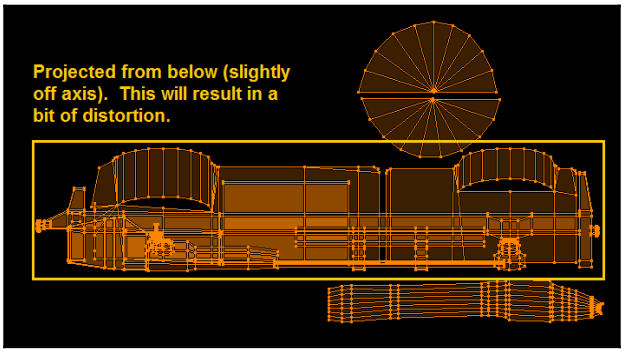
To avoid that problem, you can unwrap the model from almost the bottom view. If you unwrap it, say, 15 degrees off the axis, you should get an adequate result:



You will still have a bit of distortion:

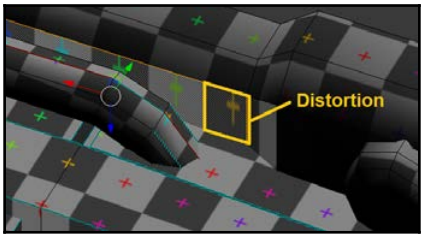


In this case, I only rotated slightly on the *X*-axis. That means that there is no texture space available for either the very front or very back polygons of the truck frame. To correct that, you'd want to be slightly off-axis on both the *X* and *Y* axes.



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.



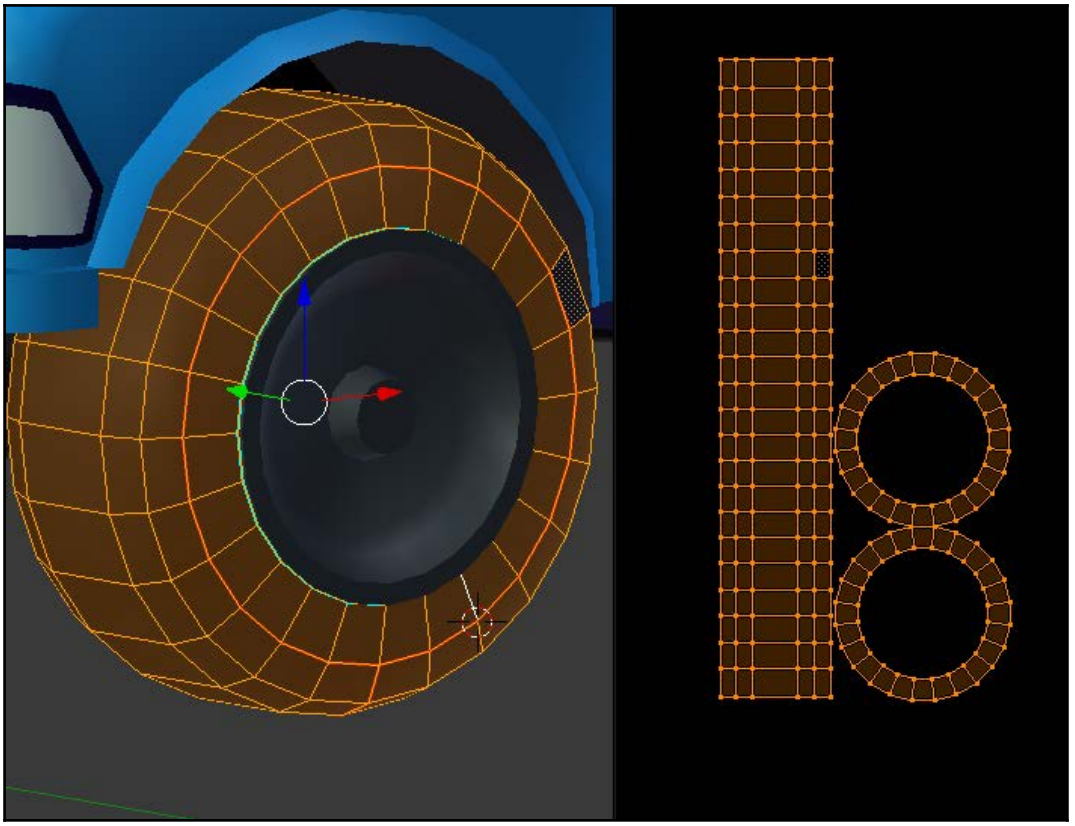
Next, we'll unwrap the tire:



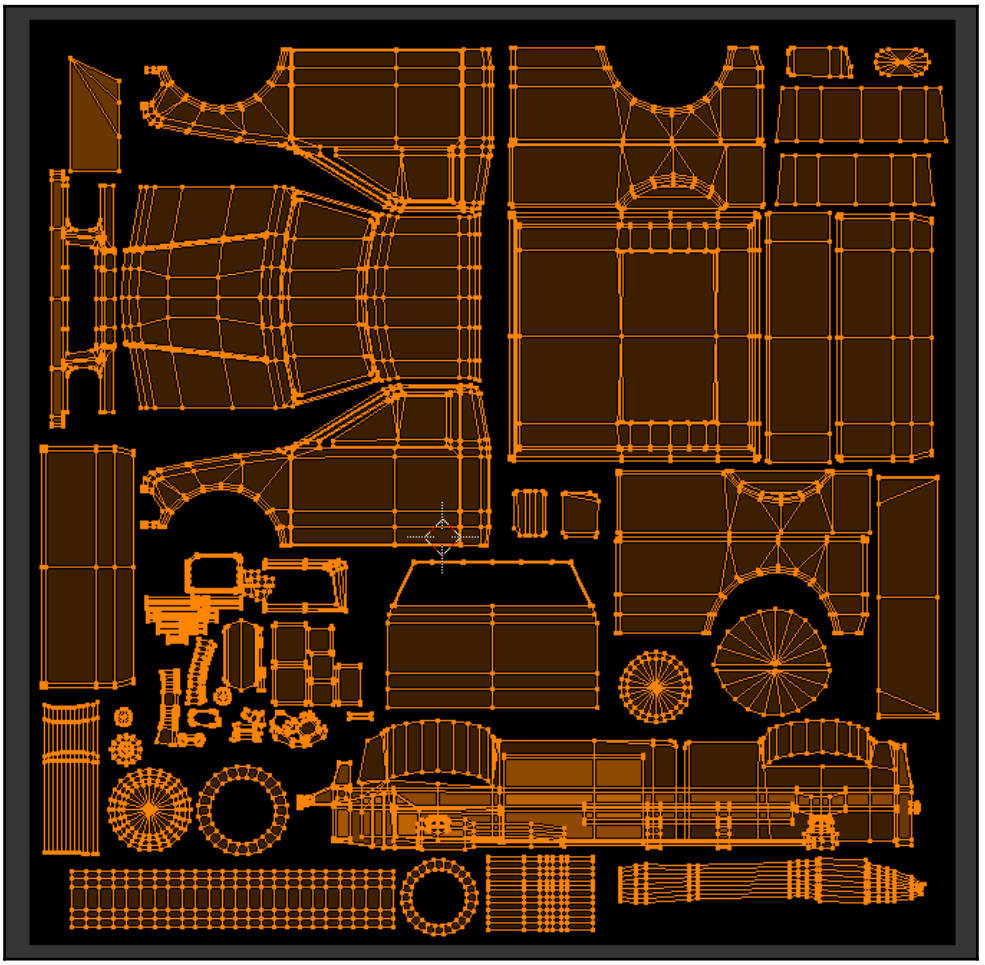
I've deleted all but one wheel/tire so that all four of them eventually use the same texture space. Once we've UV unwrapped one tire, we can just duplicate those polygons, and they'll be assigned to the same portion of the image. This is helpful for saving space.



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:



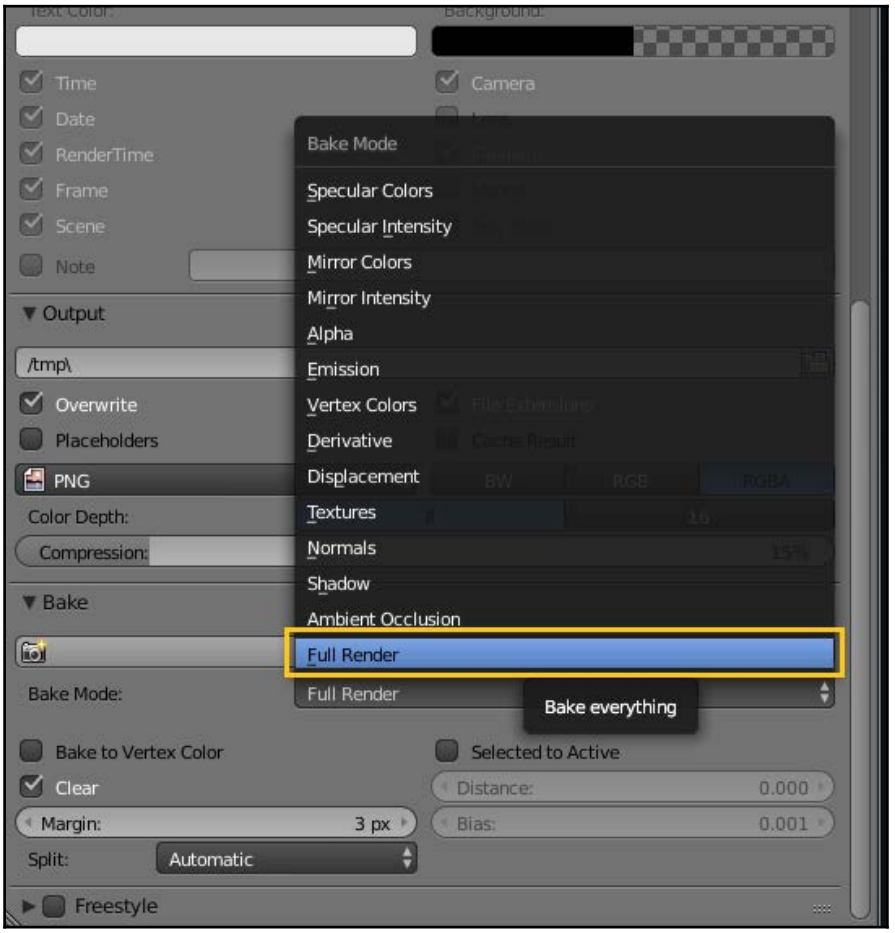
Using these same techniques, you can now go through and unwrap the various parts of the truck. Here's what my final UV map ended up looking like:



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!

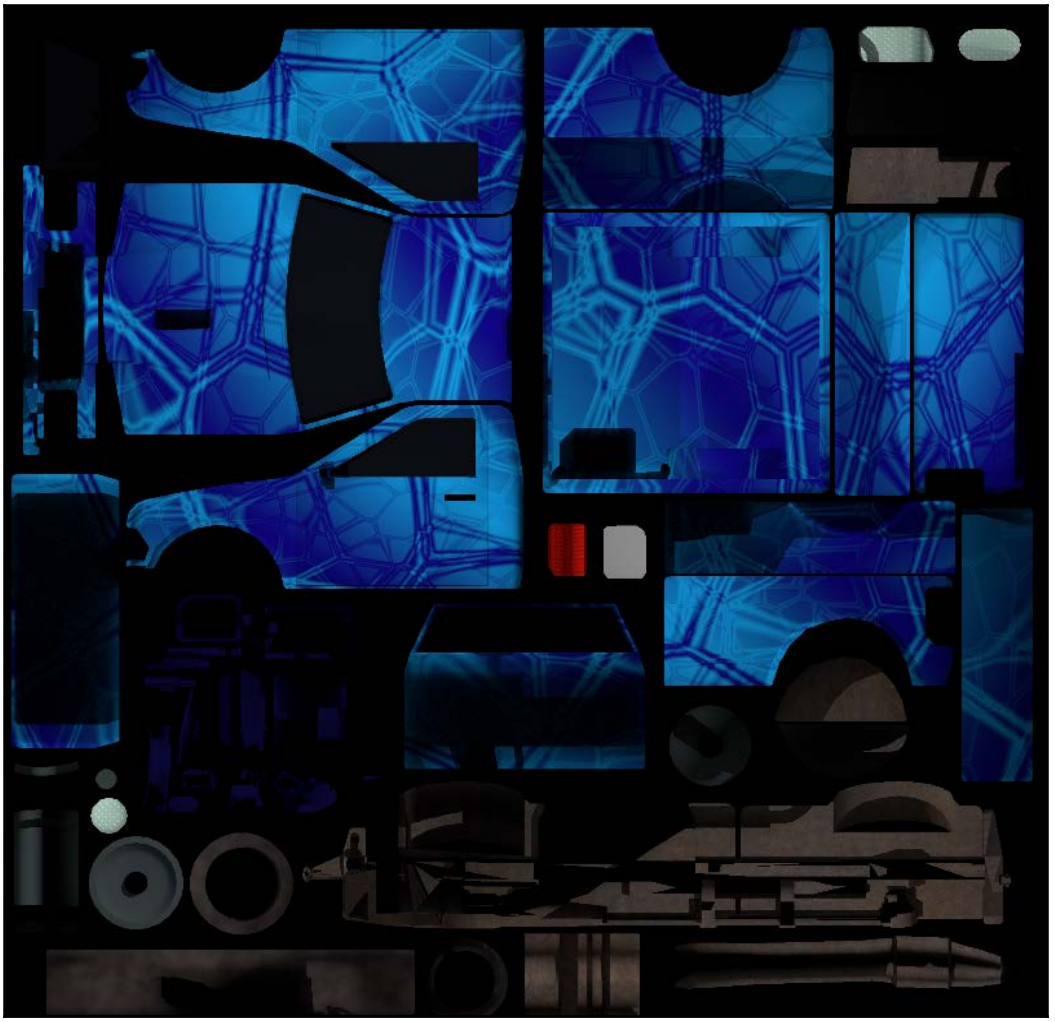
Now that we've got our UV map created, we can bake out our textures again. Last time, we just stuck with the **Texture** option for our **Bake Mode**. This time, just to experiment, we'll use the Full Render option.

When you use this option, Blender will bake not only the textures, but the light, shadow, and other aspects of your model as well:



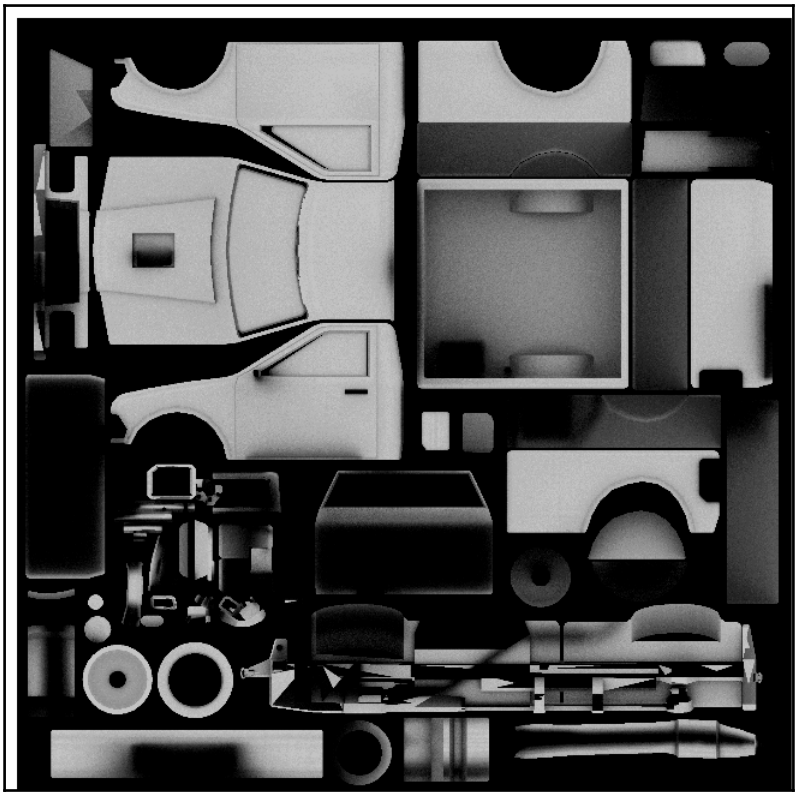
Make sure you add a few lights to your scene to test it out (otherwise the image will just be black). When you're ready, hit the **Bake** button.

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.

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.).

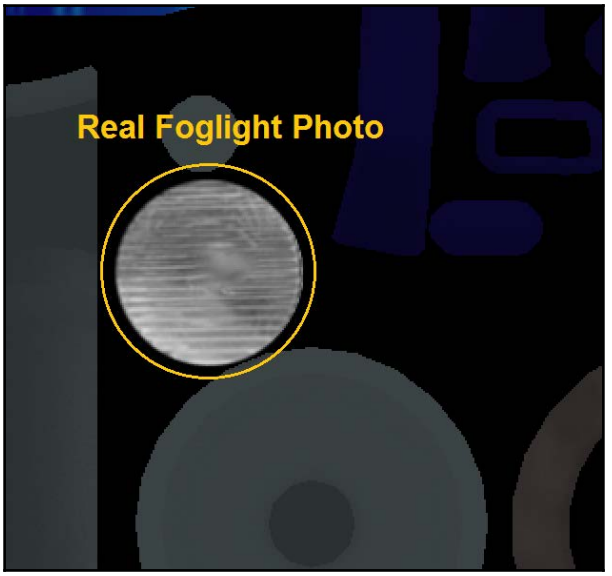
Once you've combined them together in a way that you like, you can also go through and manually paint highlights, shadows, and decals onto your texture map.

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:





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.

You can also take this opportunity to add any decals or logos that you'd like:

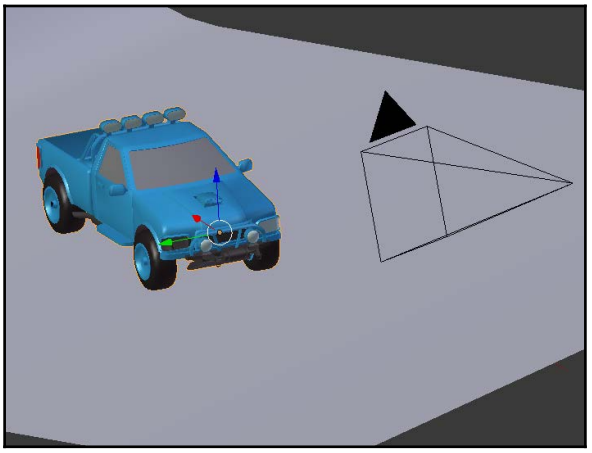


With a little time and practice, you can create some pretty nice texture maps with these techniques.

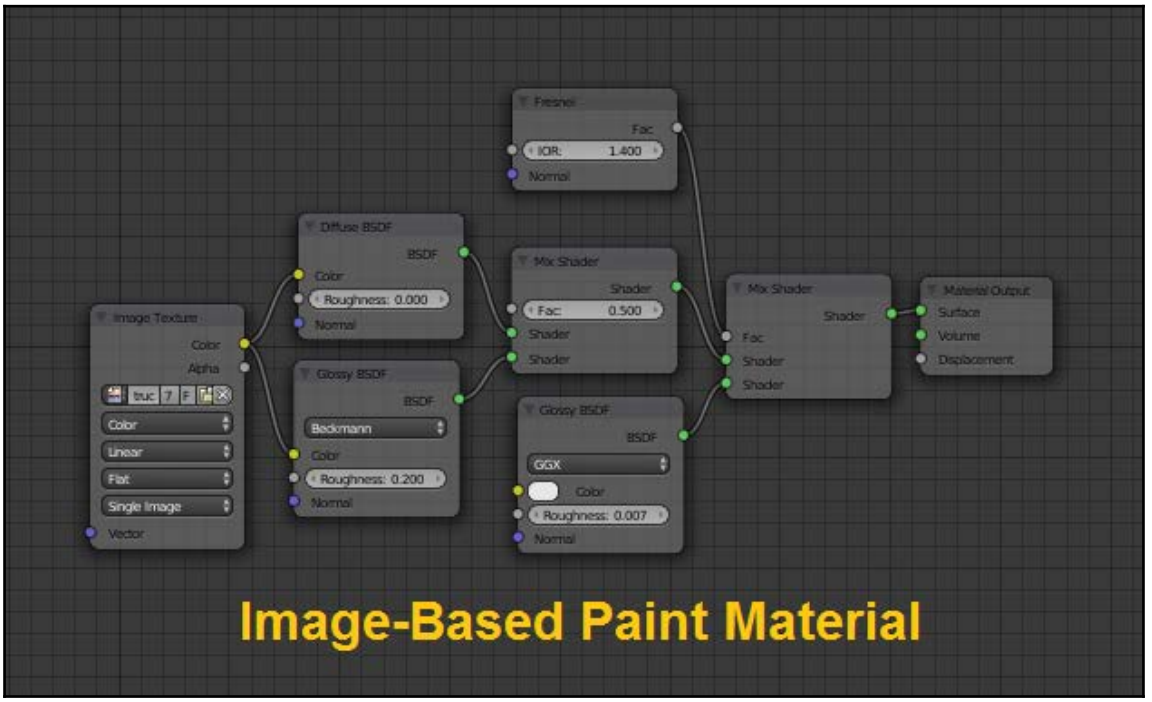
Checking the Texture Map

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.

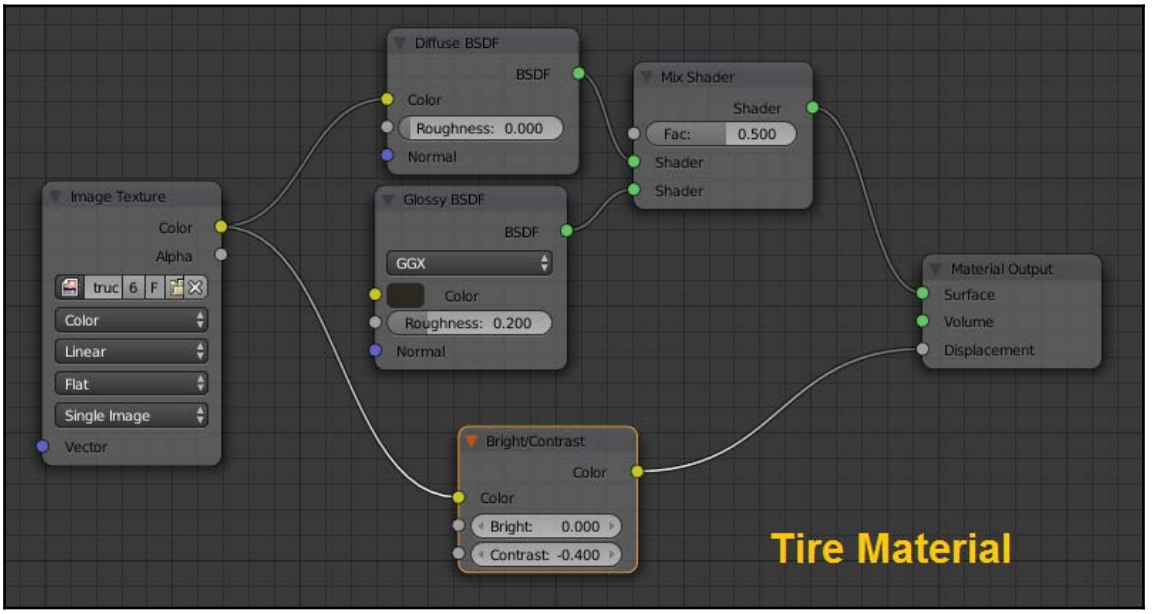
You can start by just adding a basic background, lighting setup, and camera:



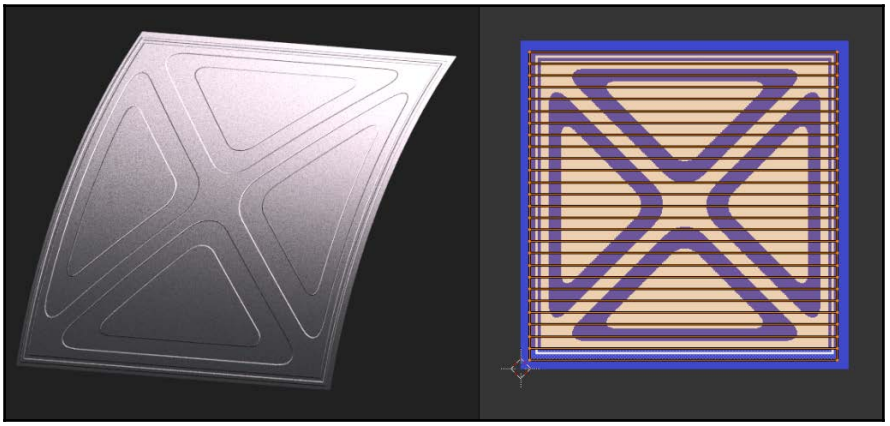
Then, we'll create our materials. For instance, here is a quick car paint material using the texture image:



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:



And here's a quick version of the truck that uses normal maps to add damage/detail to the body (Render Credit: *James Clayton*):



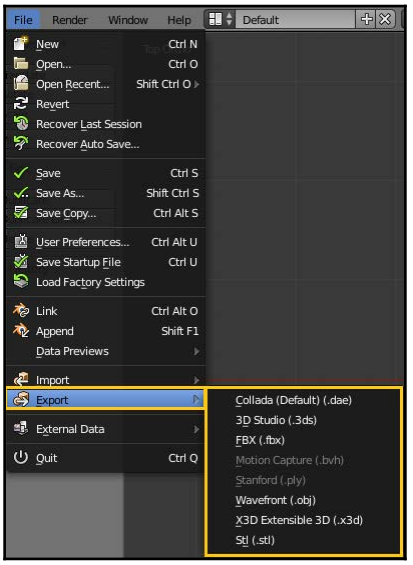
Feel free to experiment with separate Displacement/normal maps. However, keep in mind that every image you have to load will increase render time and place additional stress on the CPU/GPU. Only use a separate map when you're sure that your game or other 3D program can handle it.

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.

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.

Index

3

3D models
  building, for external applications 301, 302, 303, 304

A

accessories, spacecraft
  creating 133
angles
  adjusting 66
Array modifier 83
Ashikhmin-Shirley 101
Average Island Scale option 352

B

barrel, sci-fi pistol
  creating 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 23, 24
  Edge Split modifier, adding 19
  handgrip, modeling 25, 26, 27, 28, 29, 30, 31
  Mirror modifier, adding 10
Beckman 101
Bevel tool 21
Blender Cycles
  about 250
Blender Internal
  about 250
  materials, creating 284
Blender
  normals, correcting 15
Bridge Edge Loops 13
Buffer Shadow
  about 281

C

circular cut
  adding, for screw holes 65
  creating 47
circular shapes
  adding 66
cockpit, spacecraft
  adding 197, 199, 201
color-change effect
  creating 106
curved background plane
  adding 95
  Diffuse material, adding 96
  Subdivision surface, adding 95
Cycles Render engine
  about 88
  using 89

D

decals
  adding, with UV maps 242, 244
details
  adding, to spacecraft 138, 139, 141, 142, 143, 144, 145, 146, 147, 148, 149, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 164

E

Edge Split modifier 66
exhaust ports, spacecraft engine
  working on 170, 173, 175

F

final details
  adding, to sci-fi pistol 72

Freestyle
about 250
edges, marking 283
modeling 250
Full Render option 362

G

GCX 101
Global Illumination
about 364
GPU rendering
enabling 90
grappling gun
adding, to spacecraft engine 172

H

handgrip, barrel
about 33
customizing 34
modeling 25, 27, 28, 29

I

Image Editor
details, adding 365
Inset tool 55, 59, 127, 139
InsetExtrude tool 59

K

Knife tool 142
about 126
knife tool
activating 17

L

landing gear
building 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196
Low-Poly Racer
about 325
materials, adding 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336
truck model, creating 304, 305, 306, 307, 308, 309, 310, 311, 312
truck model, detailing 315, 316, 317, 318, 319, 320, 321, 322, 323
workflow, texturing 325
Lunatics!
URL 299

M

manifold meshes
versus non-manifold meshes 312, 313, 314
materials, spacecraft
creating 230, 231, 233, 234, 235, 236, 237, 238, 239, 240, 241
materials
adding, to sci-fi pistol 99
creating, for robot 284
creating, of spacecraft 226, 228, 229
mechanical details
creating, workflow 76
Mesh Display 14
Mirrored Objects
parts, adding 58
models, spacecraft
creating 224

N

N-Gons 15
Node Editor 102
non-manifold meshes
versus manifold meshes 312, 313, 314
normals 14

P

Path objects
working with 209, 210, 211, 212, 213, 214, 215, 216
pipe-style detailing, sci-fi
adding 77
major segments 78
minor segments 78
Torus, adding 77
pipes
adding, to spacecraft engine 166, 167, 168, 169
Project From View option 346
Proportional Editing 61

Q

quads 15

R

Ray Shadow
  about 281
robot
  arms, detailing 270
  body, creating with Subdivision Surfacing 263
  building 261
  material options 292, 298
  materials, creating 284
  rendering 292, 294, 296
  scene, preparing 277

S

scene, spacecraft
  camera, adding 98
  creating 224, 225, 226
  preparing 223
scene
  light, adding 279
  preparing, for robot 277
  setting up 89
sci-fi pistol
  barrel, creating 8
  detail, adding 72
  mterials, adding 99
  overview 6
  rendering 88
  shapes, cutting 47
  shine, adding 100
  texturing 88
Screw tool
  about 178
segments 21
shading problems
  fixing 54
Sharp 101
Shrink Wrap modifier 47
Shrinkwrap modifier
  adding 67
Solid Shading 94
spacecraft engine
  exhaust ports, working on 170, 171, 173, 174, 181
  grappling claw 178
  grappling gun, adding 172
  mirror modifier, adding 174
  missile launcher, working on 181
  pipes, adding 166, 168, 169
  Sensor Suite 183, 184
spacecraft
  accessories, creating 133
  body, building 118
  bottom of ship 183
  building 118
  cockpit, adding 197
  details 158
  details, adding 137, 139, 141, 142, 143, 144, 145, 146, 147, 148, 149, 151, 152, 153, 154, 155, 156, 157, 159, 160, 162, 163
  final view 247, 248
  finishing touches, adding 217, 218, 219, 220
  landing gear, building 185
  materials, creating 228
  models, preparing 223
  scene, preparing 223
  small parts, adding 204, 205, 207, 208
  small parts, creating 202
Split Angle 20
Subdivision Surface modifier 55
Subdivision Surfacing
  used, for creating body of robot 263

T

texture
  adding 99
  baking 341
  mapping, changing 110
  scale, adjusting 111
textures map
  checking 367
Torus object 166
torus
  adding 77
tris (triangles) 15

U

UV Map 337
UV mapping 242
UV maps
  adding, with decals 244, 245
  creating, with decals 245
  used, for adding decals 242
UV unwrapping 337
UV/Image Editor 340

V

Vertices 9

W

Wireframe mode 96

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 `www.packtpub.com`.

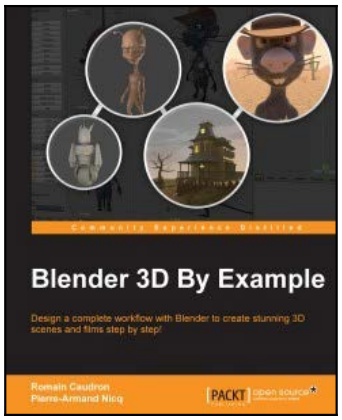
About Packt Open Source

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.

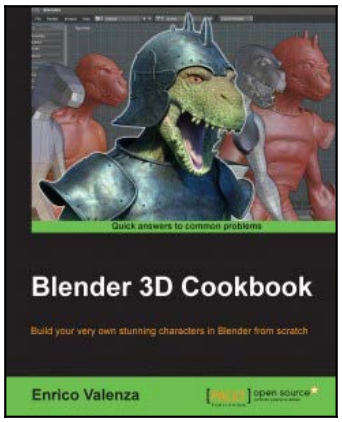


Blender 3D By Example

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

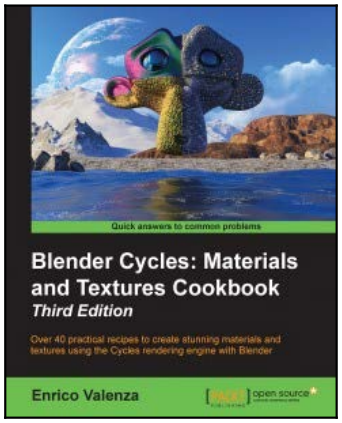


Blender 3D Cookbook

ISBN: 978-1-78398-488-6 Paperback: 608 pages

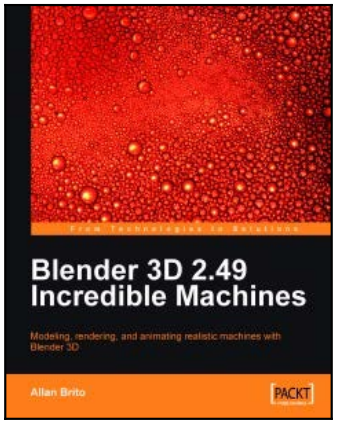
Build your very own stunning characters in Blender from scratch

- 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
• 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