Blender 2.63 Workshop - **Lesson 2**

Carl R. Knapp **CarlRKnapp@Gmail.com**

What we will cover in :

Section 1: Building a Mesh vertex by vertex
Section 2: Introduction to Modifiers

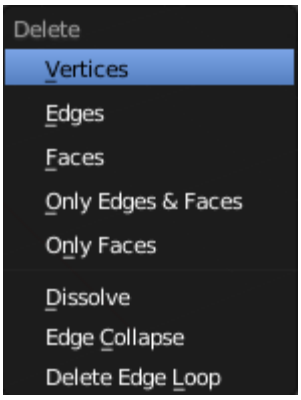
Section 1: Building a Mesh vertex by vertex

Section 1a - Mesh Primitives and Object Data

In our default scene we are provided a cube. If we break down the cube we find that it consists of a total of 8 **Vertices**, **12 Edges**, and **6 Faces**. The cube is the center is just one of the many available mesh **Primitives**. A **Mesh** is nothing more than an object which consists of **Vertices**, **Edges**, and **Faces**, and a **Primitive** is a basic shape which can be used as a starting point. In fact, any **Primitive** object could be built by hand by the careful placement of **Vertices**, **Edges**, and **Faces**.

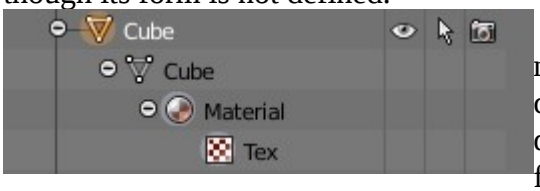
There are many instances where we find that starting from a **Primitive** object is not desired; perhaps we want to create something a little more free-form, or we may even want to trace a picture. Regardless of the reasoning, it is possible to create an object from "scratch".

In order to build a mesh piece by piece we do need a mesh object to start with. Since our default scene comes with a cube it makes sense to use the cube. If we **TAB** into **Edit Mode** with the cube selected(which it is by default) and press the **X** key it will bring up our **Delete Menu**.



If we choose the option **Vertices** we will now see that the cube is gone, however, note that the cube object still exists in the **Outliner** and we are still in the **Edit Mode** of the cube. How can this be true?

All objects in Blender are made up of what is referred to internally as **Object Data**. If we had selected the cube in **Object Mode** and deleted it, we would no longer have an object in our **Scene** named "Cube" because we would have deleted the object and all of its **Object Data.** By deleting all the **Vertices**, **Edges**, and **Faces** in **Edit Mode** all we did was destroy its form, but there still is an object called "Cube" in our scene and **Object Data** still exists for it even though its form is not defined.

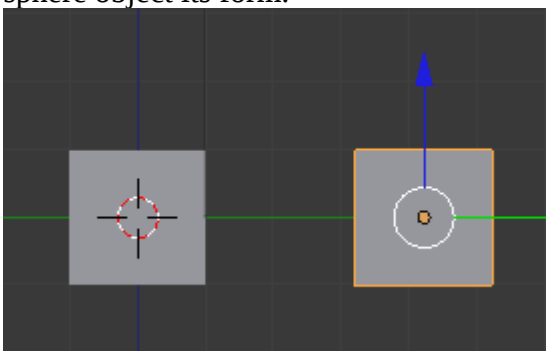


Object Data can be thought about as all the components that make up an object. In the picture to the left the very first item can be thought of as the object's name which can be changed quite easily. **Primitives** will always have a default name just for the sake of ease of use, if we were to add a second cube, the new cube would have the number **.001** appended to the end. The number will increment by 1 for each identical object added into the **Scene**.(We will talk more about adding objects in future lessons).

Blender 2.63 Workshop - **Lesson 2**

Carl R. Knapp **CarlRKnapp@Gmail.com**

The next item down is the actual **Object Data**. It is the shape and form of the object. In the picture below we have two cubes, however one of them started its life as a sphere. On the right we can see how this works. The sphere object actually has the **Object Data** from the cube, this object data told the sphere object its form.

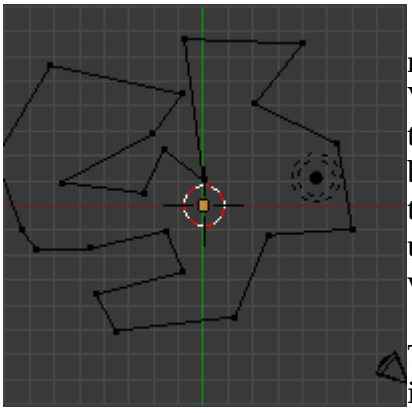


Section 1b – Building a mesh from scratch

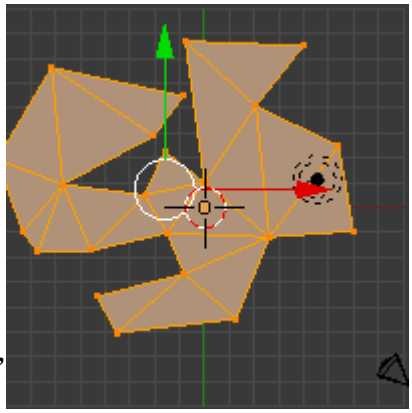
in **Edit Mode** holding down **CTRL(Control)** and clicking the **Left Mouse Button** will create a **Vertex** at the location of your mouse pointer. If you continue making **Vertices** they will be connected to each other by an **Edge**. Since space is three dimensional in Blender, but a mouse only moves in two dimensions the viewport angle is used to determine whichever dimension is not represented.

If we hold down the **SHIFT** key and select two **Vertices** with the **Right Mouse Button** and press the **F** key we will manually create an **Edge** between both **Vertices**.We can also create **Faces** in the exact same manner, in order to create a **Face** we must have at least 3 **Vertices** selected. With the **Vertices** that you want to make the **Face** with selected again press the **F** key.

There are several ways to “fill” an object created this way, and ultimately the method of choice will depend on a number of factors such as the complexity of texturing on the object, if it needs to move, and how much detail will be observed by the viewer. For static objects that will be far away from the vantage point of the viewer it is completely reasonable to use a lower quality object, ultimately however this is a decision that comes down to perspective and deadlines.

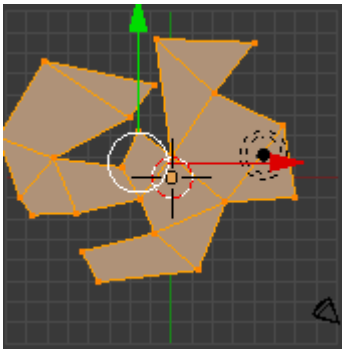


As an example, the object to the left of me is nothing more than a collection of **Vertices**, and **Edges**. I don't care about topology, it won't be animated, and will be far enough away that none of those things matter. In this instance we can use the **Fill** tool by selecting all points with **A** and pressing **ALT F**.



The **Fill** tool does not do anything fancy, it simply attempts to fill in the area selected with **Faces.** It is a quick and dirty tool, and will in no way

produce preferred results, but is nonetheless a pretty darn convenient tool to have.



One of the reasons why the **Fill** tool is not preferred is because it indiscriminately produces triangles which tend to make things like animation and texturing work rather poorly. Pressing **ALT J** will take the selection and convert as many triangles into quads as possible. Other tools are available to do similar functions such as the **Beautify Fill** tool or the **Remesh Modifier**.

In conjunction with **Extrusion** we now have the tools to take a 2 dimensional object created with this method and make it 3 dimensional. If I press **E** with this object I will extrude all the **Faces** in one direction.

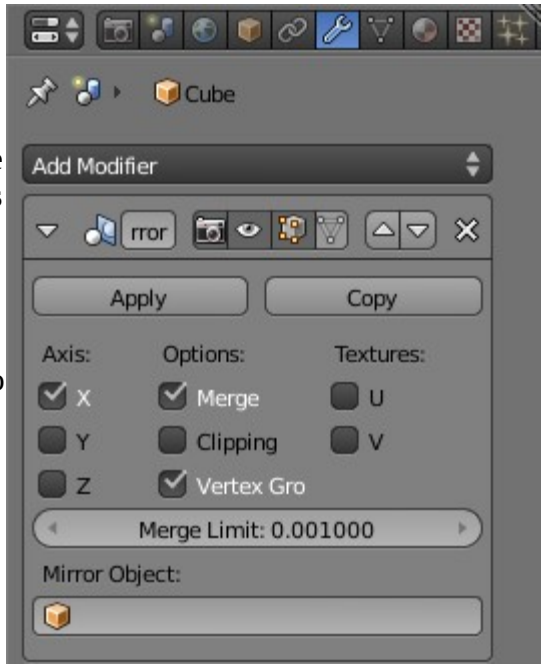
Section 2: Introduction to Modifiers

As you continue to work with Blender you will likely find **Modifiers** to be some of the most indispensable tools available. Put simply, **Modifiers** allow you to make changes to a mesh without actually altering the object. These changes are non-destructive, and can be put on and taken off with no permanent effect on the object it is applied to. **Modifiers** are located in the **Properties Editor Type**. We will be looking at the **Mirror** and **Subsurface Division Modifiers**.

All **Modifiers** have common options. **The Camera Icon** affects the renderability of the **Modifier**(If it shows up in the final render). **The Eye Icon** affects whether or not you can see the **Modifier** in the viewport but does not affect renderability. The cube with 4 smaller orange cubes affects whether or not the **Modifier** is viewable in **Edit Mode**, but does not affect it in **Object Mode**.**The Triangle Icon** affects whether or not it affects the **Editing Cage**, this option is not available with all **Modifiers**.

Modifiers can also be applied to the mesh making those modifications permanent, however there are very few times where I feel this would be necessary unless the object is going to be exported for 3D printing, or use in a game engine.

Section 2a – The Mirror Modifier

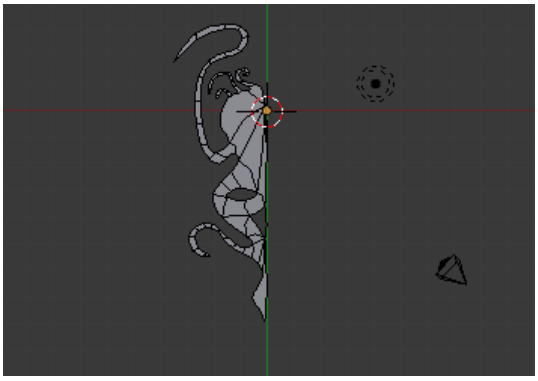


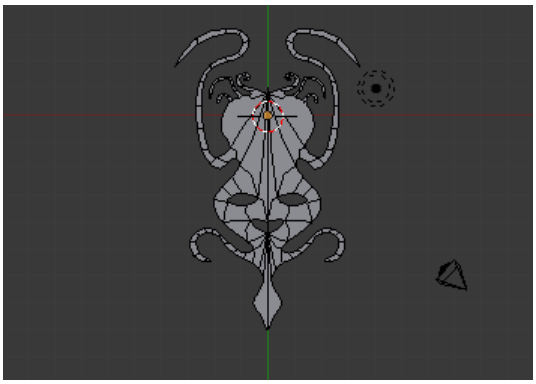
The **Mirror Modifier** allows you to mirror the mesh along a specific **Axis**, by default this is the **X Axis**(Split at the **Z Axis**). It is possible to mirror an object along multiple **Axes** as well.

By default the **Merge** option is ticked which means that any **Vertices** along the threshold of where the mirroring starts from will be treated as one **Vertex** or rather merged into one **Vertex**. Clipping prevents **Vertices** from traveling to the mirrored side and will lock them to the mirror threshold. The **Vertex Groups** option will mirror **Vertex Groups** on the

Carl R. Knapp **CarlRKnapp@Gmail.com**

mirrored side as a different set of **Vertex Groups** if the proper naming convention is used. To mirror **Vertex Groups** you need to make sure that group name has **.L** or **.R** appended to the end *i.e.* ***cheek.R*** *will turn into* ***cheek.L*** *if mirrored along the* ***X Axis***.





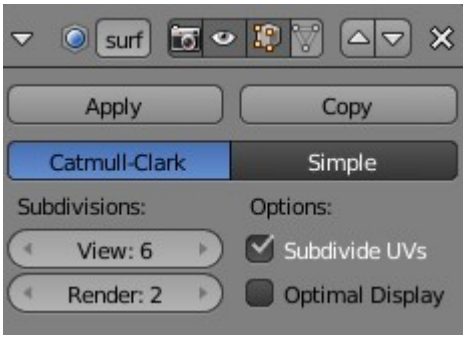
Here I have a mask that is actually just one half, however if I add the **Mirror Modifier** I now have a full mask.

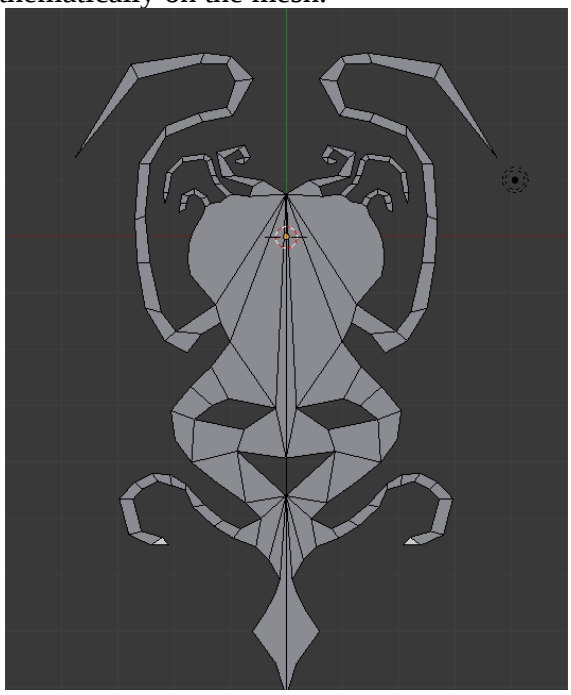
Section 2b – The Subsurface Division Modifier

One of the hardest things to do in 3D modelling is making round corners. Making objects appear more "organic" would be nearly impossible without the aid of the **Subsurface Division Modifier** (typically just called a **Subsurf Modifier**).

Put simply this Modifier makes rounded corners mathematically on the mesh.

To the right is the same mask as above in the **Mirror Modifier** example, but with the **Subsurface Division Modifier** removed, we can see the rounded **Edges** are not present.

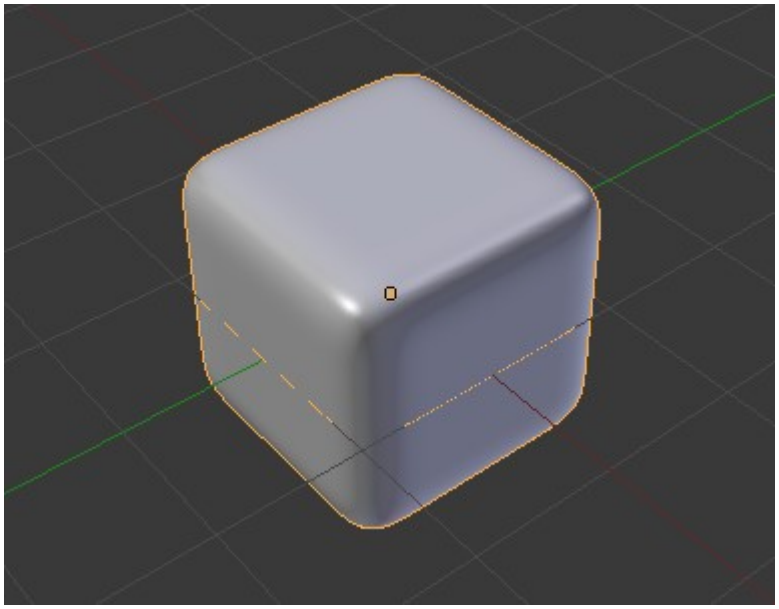




Carl R. Knapp **CarlRKnapp@Gmail.com**

Options for this **Modifier** are actually fairly simple. **Catmull-Clark** is the the name of the mathematical algorithm used by default, more info can be found here: (http://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface) .

Subdivisions can be set individually for both the viewport and for rendering. Although it is idea to use the same if possible, this modifier can be quite taxing on a CPU and should likely be capped at 2 for the viewport and 3 for rendering.



On the right is a heavily Subsurfed cube.

Since I expect that this **Modifier** will be covered extensively over the course of these lessons my only goal was to introduce the tool so far, we will work with it more over time.